CENTRO BRASILEIRO DE PESQUISAS FÍSICAS

COORDENAÇÃO DE FÍSICA DE ALTAS ENERGIAS - COHEP

Upgrade do detector do Projeto CRE4AT dedicado a medidas de raios cósmicos para estudos de física atmosférica

Diogo Ayres Rocha

Orientador:
Prof. Dr. André Massafferri Rodrigues

Rio de Janeiro
Abril, 2024

"UPGRADE DO DETECTOR DO PROJETO CRE4AT DEDICADO A MEDIDAS DE RAIOS CÓSMICOS PARA ESTUDOS DE FÍSICA ATMOSFÉRICA."

DIOGO AYRES ROCHA

Dissertação de Mestrado Profissional em Física com ênfase em Instrumentação Científica, apresentada no Centro Brasileiro de Pesquisas Físicas do Ministério da Ciência Tecnologia e Inovação. Fazendo parte da banca examinadora os seguintes professores:

André Massafferri Rodrigues – Orientador/CBPF

Antônio Pellegrino – NIKHEF

Documento assinado digitalmente
ALINE GESUALDI MANHAES
Data: 22/08/2024 08:31:07-0300
Verifique em https://validar.iti.gov.br

Aline Gesualdi Manhães – CEFET-RJ

Rio de Janeiro, 08 de maio de 2024.

DEDICATÓRIA

Dedico este trabalho a todos que me incentivaram e ajudaram nesta caminhada, em especial a Deus, à minha família, aos meus amigos e a todos os professores que, pela sabedoria e conhecimentos transmitidos, fizeram parte da minha história, desde a educação mais básica até o nível aqui alcançado.

AGRADECIMENTOS

Agradeço ao pesquisador, Dr. André Massafferri Rodrigues, pela orientação dada; Ao Centro Brasileiro de Pesquisas Físicas (CBPF), por toda a infraestrutura disponibilizada para a execução deste projeto; Ao laboratório de mecânica do CBPF (LITMEC), pela ajuda nos projetos e produções das peças necessárias; Ao programa antártico brasileiro (PROANTAR), por toda logística disponibilizada para o transporte do material necessário; E a todos aqueles que colaboraram, direta ou indiretamente, para a conclusão desse trabalho.

RESUMO

Ao logo das últimas décadas, alterações climáticas marcadas pelo aumento da temperatura terrestre têm atraído a atenção da comunidade científica e de autoridades governamentais, que têm buscado incentivar e desenvolver programas para conter o avanço dessas distorções, que podem impactar, negativa e significativamente, o meio ambiente.

Após a descoberta dos raios cósmicos em 1912, diversos estudos vêm sendo desenvolvidos e, por meio deles, numerosas descobertas ocorreram. Dentre elas, um trabalho chamou a atenção em 1997, quando Svensmark and Friis-Christensen publicaram um estudo, correlacionando a cobertura global de nuvens com a intensidade dos raios cósmicos medidos. Sabe-se que a atividade solar desempenha um papel relevante na intensidade com que os raios cósmicos chegam à Terra, pois os ventos solares exercem um papel fundamental na blindagem terrestre. Por intermédio desse estudo, uma possível correlação entre a atividade solar no processo de formação de núcleos de condensação (CCN) e, posteriormente, a formação de nuvens, foi estabelecida.

Objetivando confirmar essa relação, o projeto piloto CRE@AT (*Cosmic Ray Experiment at Antarctica*) foi criado, iniciando a aquisição de dados em 2015, com a instalação do primeiro detector no módulo avançado CRIOSFERA 1 e, agora, na próxima fase, chamado de CRE4AT (*Cosmic Rays Experiment for Atmosphere*). Nessa nova fase, o projeto ganhou expansão e outros detectores serão instalados em diversos locais, com diferentes características. Pensando nisso, houve a necessidade de desenvolver um novo detector, tema deste trabalho. Dois detectores foram montados, testados, e o primeiro foi instalado em fevereiro desde ano, 2024, no refúgio IPANEMA, próximo à Estação Antártica Comandante Ferraz. Por conta da baixa influência humana e características únicas, o continente Antártico é uma excelente localização para o desenvolvimento deste trabalho. A técnica de estudo realizada pelo experimento CRE4AT busca explicar a formação de nuvens por meio da medida de fluxo de raios cósmicos, nos períodos em que ocorrem variações abruptas. O detector apresentado nesse trabalho realiza essa medida em três grupos independentes e os dados adquiridos durante 1 mês aproximadamente, no o período de instalação, serão utilizados para avaliação do detector produzido.

Palavras-chave: Instrumentação eletrônica; Raios Cósmicos; Nuvens; SiPM; Antártica.

ABSTRACT

Over the last decades, the climate has been changing, characterized by the rise of the temperature on earth. It has been drawing attention to the scientific community and governmental authorities, which has been trying to foster and develop programs to take steps to restrain these phenomena, which may significantly and negatively impact on environment.

After the discovery of the cosmic rays in 1912, several studies have been taking place and, through them, many discoveries have arisen. Among them, a paper published by Svensmark and Friis-Christensen in 1997, correlating the global annual mean cloud cover with the measures of cosmic rays intensity drew attention to the scientists. It is known that the solar activity plays an important role in the intensity with which the cosmic rays come to earth, as the solar wind is of fundamental importance to Earth shield. By means of this study, it was established a possible correlation between the solar activity, the process of Cloud Condensation Nuclei's formation with the formation of clouds subsequently.

Aiming to confirm this relation, the pilot project CRE@AT (Cosmic Ray Experiment at Antarctica) was created, starting the acquisition of data in 2015, with the installation of the first detector in the advanced laboratory Cryosphere 1, called CRE4AT (Cosmic Rays Experiment for Atmosphere) today. In this phase, the project became bigger and other detectors will be installed in many places with different characteristcs. it was necessary to develop a new detector, that is the purpose of this study. Two detectors were assembled, evaluated and the first one was installed in Februery, 2024, in IPANEMA refuge, near the Comandante Ferraz Antarctic Station. Due to the low human influence and its unique characteristics, the Antarctic continent is an excellent place for developing this study. The technique employed by CRE4AT tries to explain the formation of clouds, based on the measurement of the flux of cosmic rays, during the periods of abrupt change. The detector presented in this study performs the flux measurement in three independent groups and the data acquired over approximately 1 month, during the installation period, will be used for evaluating the engineered detector.

Keywords: Electronic Instrumentation; Cosmic Rays; Clouds; SiPM; Antarctica;

Sumário

Lista de Figuras

Lista de Tabelas

Abreviações

ADC	Analog to Digital Converter
APD	Avalanche Photo-Diode
ATTO	Amazon Tall Tower Observatory
BGA	Ball Grid Array
CAE	Chuveiro Atmosférico Extenso
CBPF	Centro Brasileiro de Pesquisas Físicas
CCN	Cloud Condensation Nuclei
CERN	Centro Nuclear Europeu
CI	Circuito Integrado
CLOUD	Cosmics Leaving Outdoor Droplets
COHEP	Coordenação de Física de Altas Energias
CRC	Cyclic Redundancy Check
CRE4AT	Cosmic Rays Experiment for Atmosphere
CRE@AT/ CREAT1	Cosmic Rays Experiment at Antartica
CS	Chip Select
DC	Direct Current
EACF	Estação Antártica Comandante Ferraz
ECL	Emitter-Coupled Logic
EMC	Ejeção de Massa Coronal
FEE	Front-End Electronics
FPGA	Field Programmable Gate Array
HV	High Voltage
I2C	Inter-Integrated Circuit
IA	Inteligência Artificial
INCT	Instituto Nacional de Ciência e Tecnologia
LED	Light Emitting Diode
LHCb	Large Hadron Collider beauty
LIS	Sistema de Injeção de Luz (Light Injection System)
MAPMT	Multianode Photomultiplier Tubes
MEMS	MicroElectoMechanical System
NIM	Nuclear Instrumentation Module
NTP	Network Time Protocol
NVR	Network Video Recorder
PC	Personal Computer
PCB	Placa de Circuito Impresso
PID	Proporcional, Integral e Derivativo
PROANTAR	Programa Antártico Brasileiro
QDC	Charge to Amplitude Conversion
RCG	Raios Cósmicos Galáticos
RTC	Real Time Clock

SD	Secure Digital
SiPM	Silicon Photomultiplier
SPI	Interface Periférica Serial
SSH	Secure Socket Shell
TTL	Transistor–Transistor Logic
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
UTC	Tempo Universal Coordenado
UV	Ultravioleta
VHDL	VHSIC Hardware Description Language
VME	VERSAModule Eurocard
WSL	Wavelength Shifter
ZAF	Fator do Ângulo Zenital (Zenith Angle Factor)

Capítulo 1

Introdução

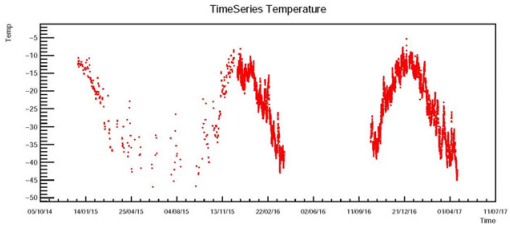
No século XVIII, o eletroscópio de folhas foi descoberto[1] e, no início do século XX, ele era o instrumento mais utilizado para o estudo da ionização causada pela radiação. Inicialmente, observou-se que algumas pedras emitiam radiação naturalmente. Naquele período, diversos pesquisadores realizaram medições de ionização em lagos, oceanos e também em montanhas, para verificar as alterações devido à altura. Experimentos com balões foram realizados e um físico chamado Victor Hess iniciou sua pesquisa nesse período[2]. Em 1912, Hess descobriu que partículas carregadas atravessavam a atmosfera em direção à Terra. O experimento consistiu de medidas de ionização da atmosfera por meio de um voo de balão, que atingiu 5300m de altitude. Pela leitura dessas medidas, constatou-se que, quanto maior a distância em relação ao nível do solo, após 1km de altura, maior a ionização, demonstrando-se, assim, que as partículas ionizantes vinham do espaço[3]. Essas partículas foram chamadas de raios cósmicos. Por conta dessa descoberta, o físico recebeu o prêmio Nobel de física em 1936.

Desde então, outros instrumentos de medição para realizar o estudo dos raios cósmicos vêm sendo desenvolvidos e são estudados até os dias atuais. Em 1911, outro instrumento importante, usado para descoberta do pósitron, foi criado por Charles T. R. Wilson, um físico escocês. Wilson observou que gotículas de água eram formadas por condensação de vapor em íons, formando o rastro de passagem das partículas. Por meio dessa observação, a câmara de nuvens foi desenvolvida.

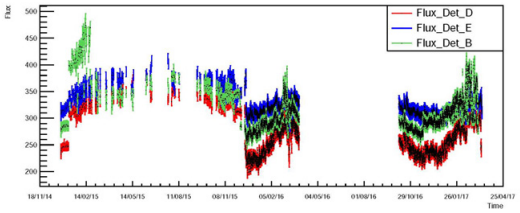
Estudos indicam que existe uma relação entre a cobertura global de nuvens e a variação do fluxo de raios cósmicos incidentes[4][5][6][7]. O entendimento dessa possível correlação com a formação de nuvens ajudaria a entender as implicações climáticas desse processo no planeta. Visando compreender essa correlação e, possivelmente, confirmar o mecanismo de formação de nuvens por chuveiros atmosféricos, foi criado, inicialmente, o projeto piloto CRE@AT, e posteriormente, o projeto CRE4AT.

Durante o primeiro período de aquisição de dados, o projeto foi chamado de *Cosmic Rays Experiment at Antarctica (CRE@AT)*, utilizando como local de instalação o módulo Avançado CRIOSFERA 1, pertencente ao Programa Antártico Brasileiro (PROANTAR), que teve início em 1982 com a primeira missão antártica (OPERANTAR 1), após a adesão do país ao Tratado Antártico em 1975[8]. A participação do Brasil foi concretizada com a instalação permanente da Estação Antártica Comandante Ferraz (EACF), na península Keller, localizada na ilha Rei George, em 1984[9]. O módulo CRIOSFERA 1 está situado em uma região próxima do polo sul magnético terrestre, que possui características importantes para o projeto. O experimento visa buscar essa correlação por meio da pesquisa de formação de nuvens em eventos intensos, onde é exequível separar características e processos climáticos de eventos específicos, causados pelos chuveiros atmosféricos. A procura por aumento de fluxo abrupto, onde o fluxo de partículas que chega à Terra e conseguem atravessar a atmosfera aumenta significativamente, principalmente nos polos, tornou-se a técnica principal para o desenvolvimento do trabalho.

A menor espessura atmosférica e reduzida intensidade do campo presente na magnetosfera terrestre nos polos magnéticos, causando menor deflexão das partículas carregadas, possibilitam uma maior radiação presente no solo. Além disso, o continente antártico também possui menor interferência humana, o que contribui para obtenção de resultados menos ruidosos. Durante os anos de 2015 a 2017, o detector de partículas carregadas, construído no CBPF, adquiriu dados nessa região de difícil acesso e reduzida infraestrutura. Devido a algumas limitações durante os invernos no continente antártico, o módulo sofreu interrupções de energia, limitando os dados adquiridos. Alguns deles podem ser vistos na figura 1.1. O detector utilizado neste projeto piloto retornou ao CBPF em 2023 e, com a produção de um novo sistema de aquisição de dados (DAQ), projetado como protótipo do sistema tema deste trabalho. Ele foi religado, coletando dados na instituição, com um sistema energeticamente eficiente em relação ao utilizado anteriormente.



(a) Gráfico temporal da variação térmica CRIOSFERA 1.



(b) Gráfico dos Fluxos de partículas adquirido por 3 grupos de detectores. No gráfico, foi retirado *burst* por meio do algoritmo de identificação.

Figura 1.1: Dados obtidos pelo experimento CRE@AT durando o primeiro período de aquisição.

Para a segunda fase de aquisição de dados, o refúgio IPANEMA, próximo à EACF, reativado entre o fim de 2021 e início de 2022 [10], foi escolhido para instalação do primeiro detector na Antártica, devido às limitações energéticas, dificuldade de acesso e de transporte de material existentes na localização anterior. Devido à expansão do projeto, que será denominado *Cosmic Rays Experiment for Atmosphere* (CRE4AT), programa-se a instalação de um detector na Amazônia, no próprio CBPF e no módulo avançado CRIOSFERA 1. Assim como na fase anterior, todo o detector foi projetado de forma a possuir um sistema de aquisição de dados autônomo, com algoritmos de recuperação em caso de problemas durante o período de aquisição.

1.1 Raios Cósmicos Galácticos e atividade solar

Os raios cósmicos galácticos são formados por partículas estáveis de alta energia, em geral prótons, núcleos de hidrogênio[11], provenientes basicamente de fontes extragalácticas. Ao passarem pela camada superior da atmosfera, essas partículas podem interagir com os átomos que a constituem, principalmente nitrogênio e oxigênio[12], formando novas partículas, que podem interagir pela segunda vez, gerando uma nova sequência de partículas. Esse processo constitui o chuveiro atmosférico extenso (CAE), cujo tamanho depende da energia da partícula que o originou e da seção de choque da interação. Um esquema desse processo pode ser visto na figura 1.2.

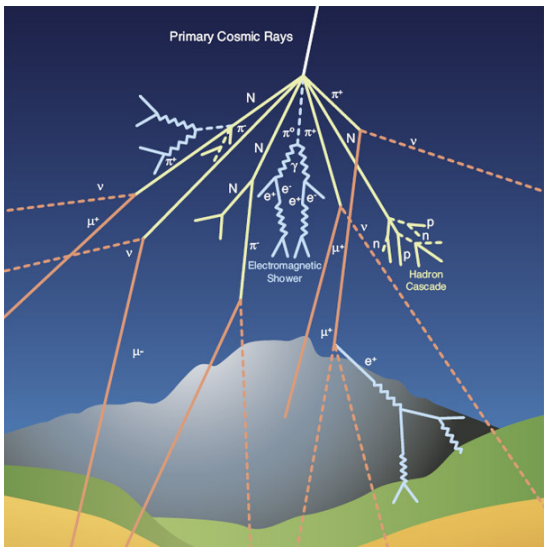


Figura 1.2: Interação do RCG com a atmosfera formando chuveiros atmosféricos [11]. Fonte: Imagem obtida no site `https://home.cern/science/physics/cosmic-rays-particles-outer-space`. Acessado em 15-11-2023.

Na primeira interação, são formados mésons (píons[1] e káons) em sua grande maioria, que, em geral, decaem, gerando múons, antes de interagirem com outras partículas[14]. Durante o trajeto de aproximadamente 15Km que os múons percorrem até a superfície terrestre, existe uma perda mínima de energia por ionização, de aproximadamente 2GeV[15], que lhes confere alto poder de penetração. No fim do chuveiro, a maior parte das partículas que podem ser encontradas são os múons, que facilmente chegam a 15% do total de partículas carregadas[16], a uma velocidade próxima à da luz[17]. A composição total de um chuveiro é, em geral, de 90% de elétrons, pósitrons e fótons, 9% de partículas alfa e 1% de partículas hadrônicas[18], sendo a maioria absorvida ao longo da trajetória pela atmosfera terrestre.

Diversos experimentos vêm estudando as relações dos raios cósmicos e suas interações com a atmosfera, buscando compreender essas relações em altas energias presentes nas partículas primárias que podem chegar com energia superior à 10^9GeV[19]. A energia presente nesses raios cósmicos ultraenergéticos supera a obtida nos maiores aceleradores de partículas até hoje construídos. Experimentos como o Laboratório Pierre Auger[20], o experimento IceCube no Polo Sul[21] e o experimento AMS-02 (Alpha Magnetic Spectrometer) na estação espacial[22] são exemplos que vêm, ao longo da última década, realizando medidas relacionadas aos RCG.

A quantidade dos raios cósmicos que atingem a superfície do planeta é dependente da atividade solar, que apresenta intensidade variável ao longo do seu ciclo e possui um semiperíodo de 11 anos[23]. O referido ciclo ocorre devido a um processo de dínamo, explicado pelo movimento de material condutor, causado pela rotação diferencial[24][25] do sol. No fim de cada semiciclo, a polaridade magnética solar é invertida e, com isso, após dois semiciclos, a polaridade original é retornada[26]. Essa teoria, apesar de proposta em 1919, ainda não é completamente compreendida[27]. Durante esse ciclo, a atividade solar, o número de manchas solares e a intensidade dos ventos solares sofrem alterações. Outros eventos também acontecem com maior ou menor intensidade durante o período do ciclo solar, como erupções solares, também chamadas de explosões solares, e ejeções de massa coronal (EMC)[28].

Os ventos solares são formados, principalmente, por prótons e elétrons, da ordem de 1keV e 1eV respectivamente[29], provenientes do plasma da coroa solar, camada superior da atmosfera

[1] A partícula subatômica Píon foi descoberta por um importante físico brasileiro, Cesare Mansueto Giulio Lattes (mais conhecido como César Lattes), em um laboratório a 5km de altura nos Andes Bolivianos, utilizando chapas fotográficas em uma técnica de emulsão nuclear, entre 1947 e 1948[13]

da estrela, que é constantemente emitido pelo sol. A Terra é parcialmente blindada contra os raios cósmicos galácticos, tanto pela magnetosfera terrestre quanto pelos ventos solares, cuja interação das partículas carregadas com a magnetosfera cria um campo capaz de bloquear parte dos RCG. Como é possível ver na figura 1.3, o período de maior intensidade dos raios cósmicos coincide com o menor número de manchas solares, logo, menor atividade solar e, consequentemente, menor intensidade dos ventos solares. Apesar das variações ao longo do ciclo, existem registros de que a intensidade dos ventos solares tem sofrido alterações ao longo do tempo, em especial ao longo dos últimos 100 anos, durante os quais essa intensidade mais que dobrou[30].

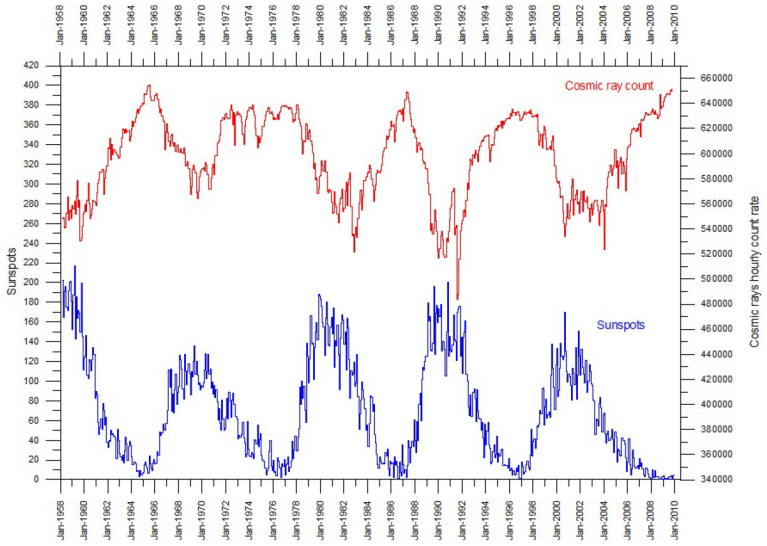


Figura 1.3: Gráfico comparando o número de manchas solares com a quantidade de raios cósmicos medidos[31].
Fonte: Retirado do site `http://www.climate4you.com/Sun.htm`. Acessado em 10-03-2024.

No pico de atividade do ciclo solar, as manchas solares são encontradas em maior quantidade. Essas regiões de intensa atividade magnética possuem menor temperatura superficial, já que o campo acentuado impede a convecção[32]. Quando esses campos se rompem em um evento de erupção solar, parte da massa coronal solar é ejetada em maior quantidade no espaço, podendo criar intensas tempestades geomagnéticas na magnetosfera e ionosfera terrestres, afetando sistemas de comunicação e de navegação[28]. Esses eventos podem ser acompanhados por uma diminuição dos raios cósmicos que conseguem atingir a atmosfera terrestre e, logo, uma queda no número de CAE e no fluxo de partículas que atingem a superfície. Esse evento de queda abrupta no fluxo de partículas é conhecido como decréscimo de *Forbush*[33]. Durante eventos de grandes erupções solares porém, uma quantidade significativa de partículas energéticas pode atingir a Terra. Nesses casos, principalmente na região dos polos magnéticos, pode ocorrer um aumento significativo de chuveiros atmosféricos e detecções de raios-x. Esses eventos abruptos de aumento no fluxo de partículas será denominado no contexto deste trabalho de *burst*[34].

Há mais de 200 anos, o astrônomo William Herschel notou que climas mais frios coincidiam com períodos de menor atividade solar[35]. Essa observação pode ser confirmada com os resultados apresentados por E. Friis-Christensen and K. Lassen em 1991, como mostrado na figura 1.4. Apesar de a temperatura das manchas solares ser mais baixa, a irradiação é compensada por regiões de intensa emissão, chamadas de *plages* e *faculae*, ao redor[36][37]. Chama a atenção a queda da anomalia de temperatura, entre 1945 e 1970, durante a qual uma subida na concentração de gases de efeito estufa, teoria usada para explicar o aumento da temperatura terrestre, não poderia correlacionar essa diminuição. Contudo, claramente, esse período coincide com o decréscimo da atividade solar.

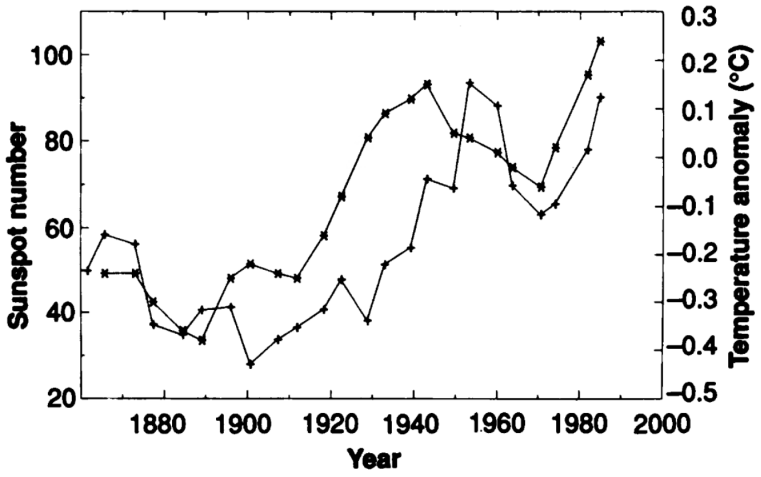


Figura 1.4: Gráfico apresentando a variação da anomalia de temperatura com relação ao número de manchas solares[38].
Fonte: Retirado do artigo "Length of the Solar Cycle: An Indicator of Solar Activity Closely Associated with Climate".

Como é possível ver na figura 1.5, a variação da irradiação solar, apesar de ter uma pequena flutuação coincidente com o número de manchas solares, também não explica completamente o aumento da temperatura terrestre.

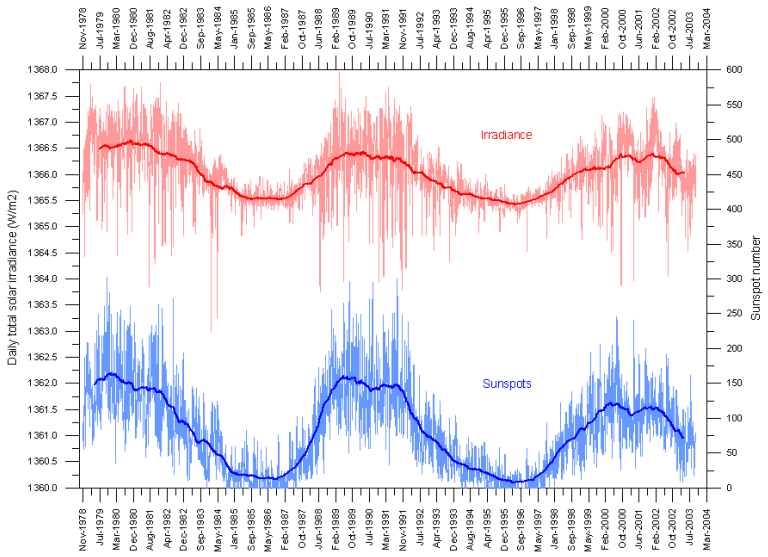


Figura 1.5: Gráfico comparando a irradiação solar com a atividade solar[31].
Fonte: Retirado do site `http://www.climate4you.com/Sun.htm`. Acessado em 10-03-2024.

Sabe-se que a maior parte do controle da temperatura terrestre é realizada a partir da relação entre a reflexão da radiação emitida pelo Sol e da radiação absorvida[39][40]. Da radiação refletida, 30% é realizada pelas nuvens, pela superfície e atmosfera, principalmente na faixa do infravermelho. Os outros 70% são absorvidos pelos gases atmosféricos, pelas nuvens, por partículas suspensas no ar e pela superfície terrestre. Apesar de as nuvens reterem parte do calor presente na Terra, a irradiação refletida pela maioria das nuvens supera essa retenção. Com isso, no geral, o aumento da cobertura global de nuvens indica uma diminuição da temperatura terrestre.

Por meio do estudo apresentado por Svensmark and Friis-Christensen em 1997 [4], foi sugerido que a cobertura global de nuvens possui um ligação com a variação dos RCG, como pode

ser visto na figura 1.6. Com o aumento dos raios cósmicos, foi observado também um aumento da cobertura global de nuvens.

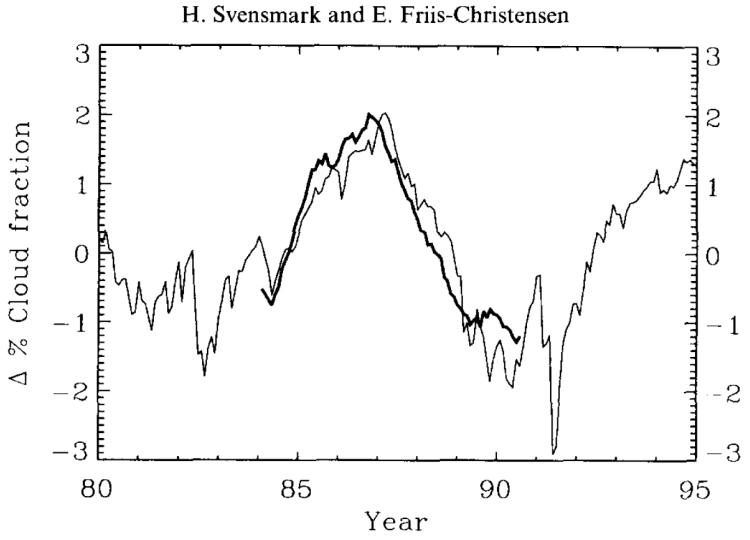


Figura 1.6: Gráfico mostrando a variação percentual média anual da cobertura global de nuvens, pela linha mais grossa, e a medida de raios cósmicos média normalizada pelo Climax, Colorado[4].
Fonte: Retirado do artigo "Variation of cosmic ray flux and global cloud coverage—a missing link in solar-climate relationships".

Para ocorrer a formação de nuvens, duas condições são necessárias:

1. O ponto de orvalho precisa ser atingido, que é o momento em que ar está saturado pela presença de vapor d'água;

2. Uma superfície para que o vapor d'água se condense, aglomerando acima do solo, que ocorre em núcleos de condensação[41].

Assim, foi proposto que a passagem dos raios cósmicos pode criar íons que se aglomeram, crescendo, e, com isso, gerando partículas aerossóis, que, posteriormente, podem formar núcleos de condensação de nuvens (CCN) e, assim, nuvens. A representação desse processo pode ser visto na figura 1.7.

Seguindo a lógica do processo de formação de nuvens por RCG, o aumento da intensidade dos ventos solares, a redução dos RCG que chegam à Terra e a diminuição da cobertura global de nuvens, observados ao longo dos anos, poderiam explicar o perceptível aumento da temperatura global. Dessa cadeia, o fenômeno físico que ainda precisa ser melhor compreendido está na correlação entre raios cósmicos e o processo de criação de nuvens. O experimento CLOUD (*Cosmics Leaving Outdoor Droplets*), desenvolvido no Centro Nuclear Europeu (CERN), é projeto mais avançado, buscando, atualmente, o entendimento dos processos físicos desencadeados pela passagem de raios cósmicos que originam a formação de nuvens.

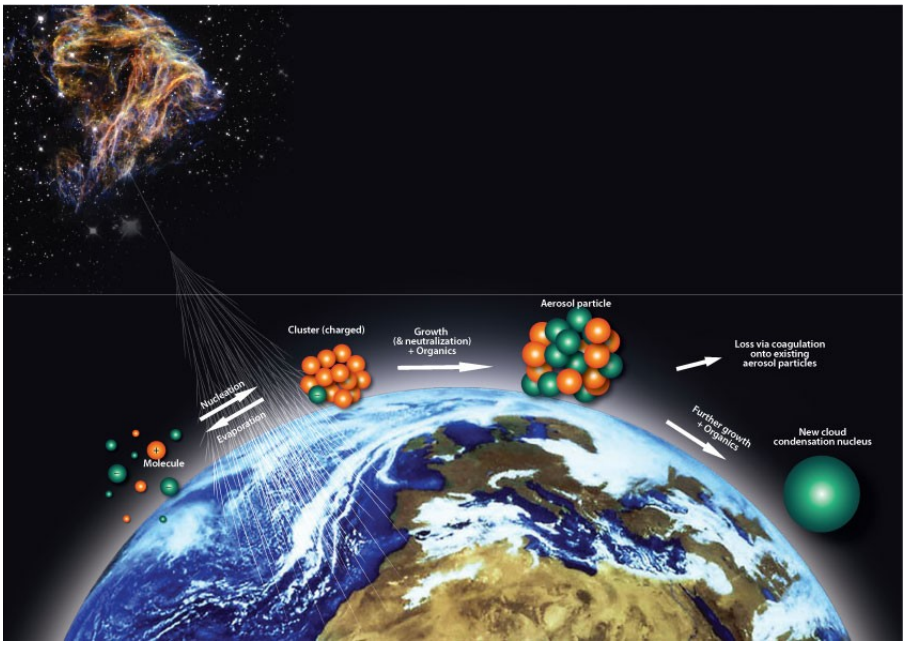


Figura 1.7: Representação da formação de nuvens pela passagem dos raios cósmicos galácticos[42].

Fonte: Retirado do site `https://phys.org/news/2011-08-cern-cloud-team-pieces-puzzle.html`. Acessado em 10-03-2024.

1.2 Projeto CRE4AT

O experimento CRE4AT nasceu do desenvolvimento do projeto piloto, instalado no módulo avançado antártico CRIOSFERA 1 por um antigo estudante do grupo, e, agora, iniciando uma nova fase, com instalação de novos detectores em outros locais.

O projeto tem por objetivo estudar os raios cósmicos galácticos e, por meio da coleta e análise dos dados obtidos, estabelecer uma ligação coerente com os mecanismos propostos por H. Svensmark and E. Friis-Christensen[4]. O estudo proposto no experimento CRE4AT será realizado por meio de um novo método, durante eventos abruptos, os *bursts*. Ele tem por escopo obter comprovações em um período menor de tempo, visto que as pesquisas realizadas até o momento fazem uso dos dados de cobertura global de nuvens para correlacionar com fluxo de partículas, o que demanda longos anos de coleta de dados. Para a nova fase do projeto, foi necessário desenvolver um detector e toda a eletrônica, mecânica e óptica associadas, de forma flexível, já que ele será instalado em locais com diferentes características topográficas e climáticas.

Dois detectores foram montados, comissionados no laboratório multiusuário da coordenação de Física de Altas Energias (COHEP/CBPF) e enviados para os locais de instalação. Durante a missão OPERANTAR 42, iniciada em janeiro deste ano, 2024, o primeiro detector foi instalado com sucesso, sendo tema deste trabalho. O segundo está na Amazônia, aos cuidados do projeto *Amazon Tall Tower Observatory* (ATTO), onde será instalado, esperando a missão que está prevista para junho deste ano. O terceiro será montado neste ano, para ser enviado à Antártica, ao módulo CRIOSFERA 1, provavelmente em agosto.

1.3 Refúgio IPANEMA

IPANEMA é um refúgio localizado na península Keller, na ilha Rei George, no continente Antártico, aproximadamente a 2km da Estação Antártica Comandante Ferraz, considerando a geografia da costa da península, como mostrado na figura 1.8. Atualmente, ele funciona como um laboratório adicional do PROANTAR, possuindo pesquisas nas áreas de biologia, meteorologia e física atmosférica e de partículas[10]. O primeiro detector e um modem com tecnologia 4G foram instalados em 2022. O modem possibilita a transmissão dos dados coletados no refúgio para o laboratório multiusuário da COHEP/CBPF, onde estão localizados dois servidores responsáveis pelo armazenamento e processamento dos dados diariamente.

Figura 1.8: Distância do refúgio IPANEMA em relação a EACF.
Fonte: Imagem extraída do Google Maps em 16-02-2024.

Atualmente, três vertentes do projeto encontram-se instalados neste local:

1. Detector de partículas carregadas e nêutrons, instalado em 2022;
2. Sistema de imageamento do céu, instalado em 2023;
3. Detector de partículas carregadas, instalado em 2024.

Além dessas vertentes do projeto instaladas no refúgio, uma análise de procura de nuvens embrionárias é realizada, via dados de satélites locais, por um estudante do grupo. Por conta da região ser próxima do polo sul, o local não possui satélites geoestacionários capturando dados. A solução encontrada foi a utilização de satélites que atravessam a região periodicamente, buscando encontrar nuvens que estejam em formação sobre o refúgio.

O sistema de imageamento, trabalho de outro estudante do grupo, é formado por uma câmera com lente grande angular, juntamente com um *Network Video Recorder* (NVR), ambos produzidos pela empresa *Dahua Technology*. O gravador é responsável por capturar uma imagem do céu sobre a estação a cada minuto e gravar pequenos vídeos ao longo do dia. Por meio de um sistema de inteligência artificial (IA), será possível buscar nas imagens, nuvens que estejam em processo de formação dentro do campo de visão da câmera. Nuvens passantes pelas imagens são ignoradas pelo processo de procura. Um sistema de aquisição das imagens pelo computador por meio de um programa utilizando a linguagem *python* também foi desenvolvido e as imagens são transferidas diariamente para os servidores do CBPF.

Os detectores de partículas utilizam dois cintiladores plásticos diferentes, assim como duas tecnologias de conversão fotoelétrica distintas. O detector, instalado na primeira missão, é constituído por dois cintiladores mais espessos, que possuem maior eficiência em detectar nêutrons, além dos múons. Ao passar pelo cintilador, a partícula pode depositar energia no material, que irá cintilar, gerando fótons. Essa luz é convertida pelo tubo fotomultiplicador, que terá, na saída, um sinal elétrico correspondente.

O segundo detector, tema deste trabalho, foi desenvolvido com cintiladores mais finos, capazes de medir o fluxo em 3 grupos independentes e utiliza um transdutor à base de silício, o fotomultiplicador de silício (SiPM). Esse detector possui maior eficiência em partículas carregadas, em particular os múons, que estão presentes em maior quantidade. O processo de desenvolvimento e construção desse detector será explicado no próximo capítulo.

Capítulo 2

Desenvolvimento do Experimento CRE4AT

2.1 Cintiladores Plásticos e Wavelength Shifters

Os cintiladores são materiais que fornecem fótons detectáveis na parte visível do espectro da luz, após a passagem de partículas carregadas ou fótons. O plástico puro não é um bom cintilador, uma vez que ele não é transparente aos fótons produzidos pela excitação e eles são absorvidos no próprio material. Alguns compostos são, então, adicionados, para torná-los transparentes à própria emissão, tornando-os eficientes[43]. Outra característica importante precisa ser levada em conta no caso de cintiladores utilizados para contagem de partículas passantes: o processo de emissão deve ter curta duração, na ordem de nanosegundos, para que seja possível utilizar o cintilador para contagem com o menor tempo morto possível. O modelo escolhido para o projeto é produzido pelo Fermilab (*Fermi National Accelerator Laboratory*) de um material fluorescente, cuja luz emitida se dá na faixa do azul. Esse modelo possui 1 cm de espessura com 5cm de largura e 1,6m de comprimento. Para montagem do experimento, o comprimento dos cintiladores é reduzido, mantendo as outras medidas.

No processo de interação de partículas carregadas com a matéria, alguns fenômenos podem ocorrer, sendo os principais: colisões inelásticas, caracterizadas pelo processo de excitação e ionização do átomo do meio, e espalhamento elástico. Além desses dois processos, também podem ocorrer emissão de radiação Cherenkov, reações nucleares e bremsstrahlung[44]. No caso dos cintiladores, o primeiro processo citado será o utilizado para detecção, visto que ele causará a produção de fótons, cintilando o meio detector[45][46].

Na excitação, os elétrons do átomo ganham energia suficiente para permitir uma troca de camada eletrônica. Ela pode ser igual ou maior que a energia entre o estado fundamental e o primeiro estado excitado, porém menor que a energia de ionização e, nesse processo, o átomo se torna metaestável. Para voltar à estabilidade, o elétron libera a energia recebida na forma de um fóton. Esse processo de fluorescência ocorre em moléculas orgânicas, independentemente do estado físico em que se encontra, diferentemente de cintiladores cristalinos inorgânicos que dependem de uma regularidade da rede cristalina como base para o processo de cintilação. Na figura 2.1 são mostrados os estados de uma molécula orgânica em um sistema π-elétron, com as possíveis transições de estados para ocorrer o processo de fluorescência.

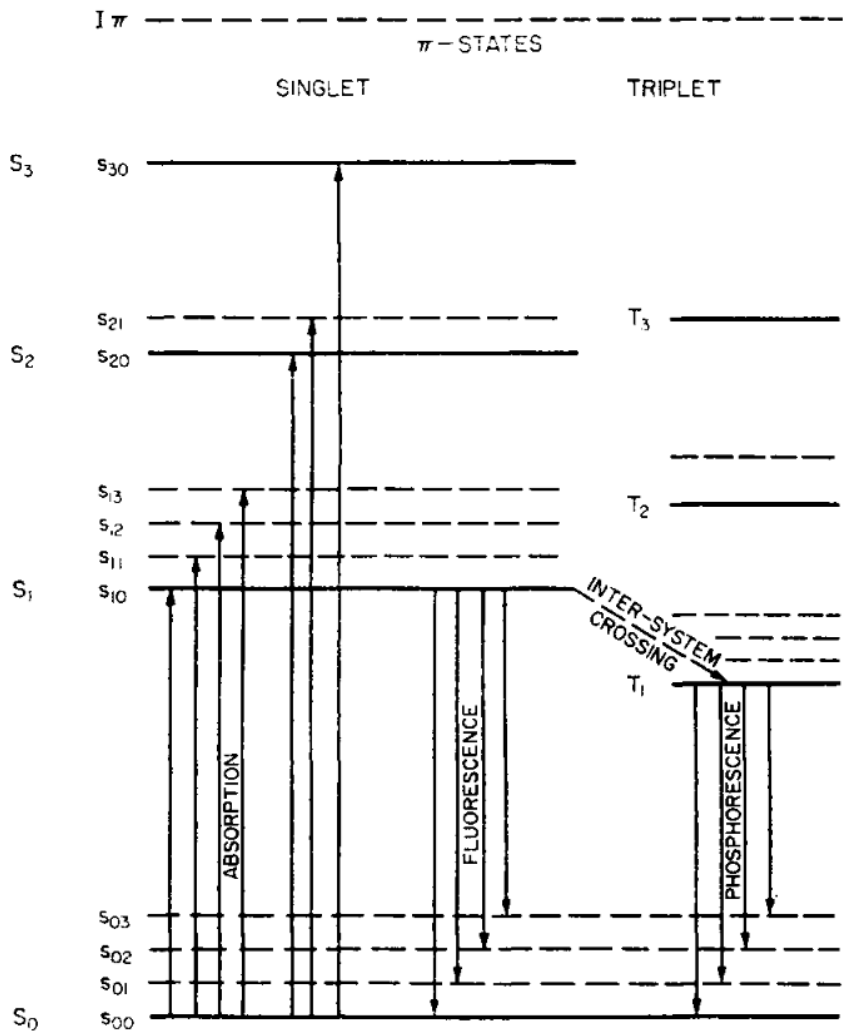


Figura 2.1: Modelo de níveis de energia em uma molécula orgânica evidenciando as trocas de energia para o processo de excitação e fluorescência em moléculas contendo um sistema π-elétron[45].

Fonte: Retirado do livro "The Theory and Practice of Scintillation Counting".

A molécula orgânica possui uma série de estados de energia que podem ser chamados S_0, S_1, S_2 e S_3, como pode ser visto na Figura acima, com seus suborbitais. Outros estados de excitação que estão correlacionados com spin 1 (estado tripleto) também são mostrados na figura como T_1, T_2 e T_3[45]. Para uma molécula orgânica ser interessante para a produção de cintiladores, a diferença de energia entre os níveis eletrônicos é da ordem de eV, visto que os fótons gerados estarão na faixa do espectro visível. O processo de estabilização atômica ocorre com os elétrons inicialmente decaindo para o primeiro estado excitado, S1, em um processo de conversão interna que ocorre sem emissão de fótons, com uma reorganização eletrônica, conservando a energia total. Esse processo conhecido como 'degradação interna' acontece em alguns picossegundos[45]. Posteriormente, o elétron excitado retorna ao estado fundamental, liberando um fóton.

O fato de o fóton ser liberado somente na transição do primeiro estado excitado para o estado fundamental é de suma importância para cintiladores orgânicos e explica por que ele é transparente à própria luz gerada pela excitação. Por conta desse fóton ser de baixa energia, ordem de eV, ele possui energia no limiar necessário para excitação de outras moléculas orgânicas do material. Sendo assim, existe uma pequena quantidade de fótons que são absorvidos dentro do próprio cintilador. Contudo, a maioria passa pelo material sem interação.

Sendo τ o tempo para o processo da troca de camada S_0 para S_1, a equação que descreve a intensidade da fluorescência ao longo do tempo pode ser dada pela equação:

$$I = I_0 e^{\frac{t}{\tau}} \tag{2.1}$$

em que I_0 é a intensidade máxima inicial. Na maioria das moléculas orgânicas utilizadas em cintiladores plásticos, τ é da ordem de nanosegundos.

No processo de ionização, a energia dada ao elétron de valência é igual ou superior à energia de ligação dele ao átomo, formando, assim, um par elétron-íon. Assim como no processo de excitação, uma reorganização eletrônica ocorre posteriormente, liberando um fóton.

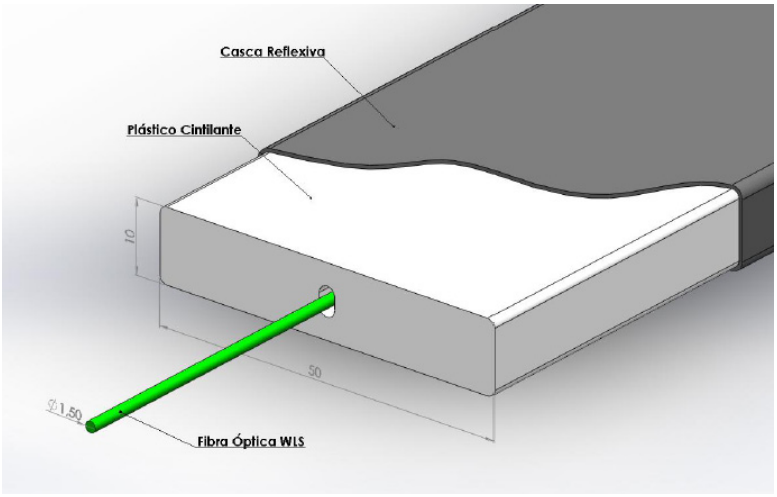


Figura 2.2: Esquema da *Wavelength Shifter* inserida no interior do cintilador plástico, como é feito na montagem do experimento[47].
Fonte: Retirado do projeto final de curso "DESENVOLVIMENTO DO EXPERIMENTO ANTÁRTICO DE MONITORAÇÃO DE RAIOS CÓSMICOS PARA O MÓDULO CRIOSFERA I".

A *Wavelength Shifter* (WSL) é uma fibra que possui a função de absorver esses fótons gerados pelo cintilador e emiti-los novamente em uma frequência diferente. No modelo escolhido, Y11(175)MSJ, produzido pela *KURARAY*, os fótons de espectro azul liberados pelo cintilador, com espectro de absorção adequado para WLS, são reemitidos no espectro verde, de menor energia, em maior quantidade[48]. Apesar de possuírem uma concentração diferente do dopante Y11, a caracterização da WSL Y11(200)MSJ, também produzida pela *KURARAY*, pode ser encontrada na referência [49]. Essa característica é importante devido à possibilidade de perda de alguns fótons no trajeto até o transdutor fotoelétrico. Com o aumento do número de fótons guiados ao transdutor ocorre o aumento da eficiência da detecção. Esses fótons atingem a fibra ortogonalmente, que fica inserida no interior dos cintiladores, como é possível observar na figura 2.2, e, os fótons reemitidos são guiados até as extremidades da WSL. Uma foto do modelo utilizado pode ser visto na figura 2.3.

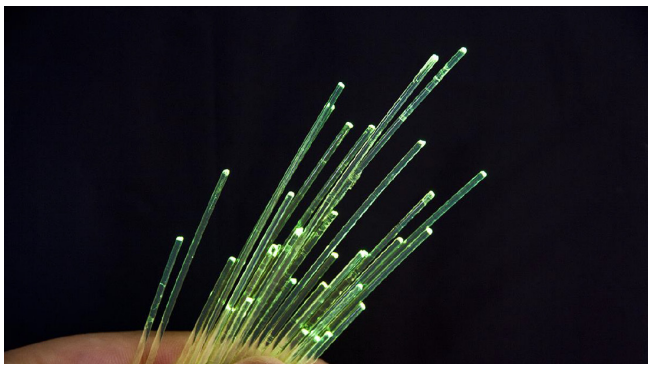
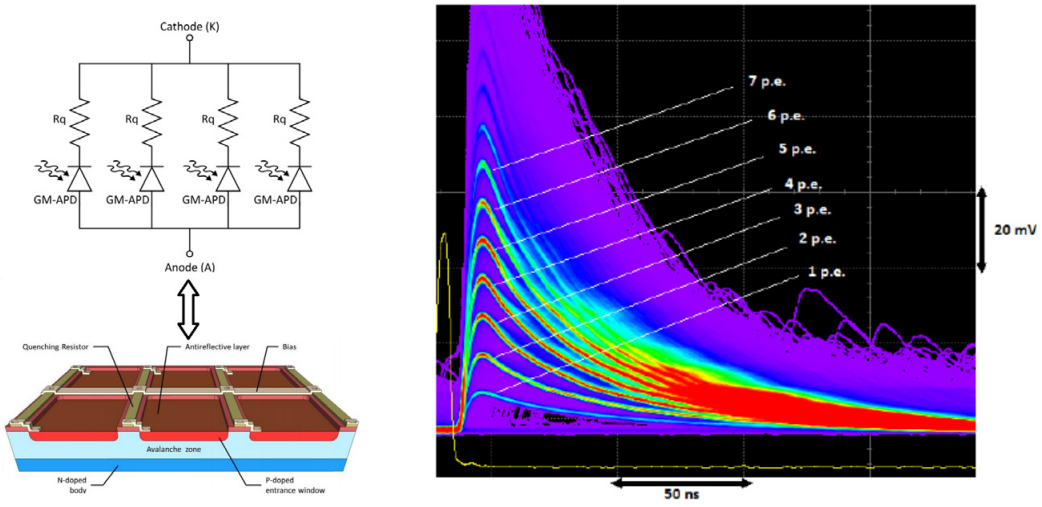


Figura 2.3: Foto da WSL utilizada no projeto[47].
Fonte: Retirado do projeto final de curso "DESENVOLVIMENTO DO EXPERIMENTO ANTÁRTICO DE MONITORAÇÃO DE RAIOS CÓSMICOS PARA O MÓDULO CRIOSFERA I".

2.2 Fotomultiplicador de Silício (SiPM)

Os fotomultiplicadores de silício são dispositivos semicondutores com dopagem N e P, assim como diodos, porém, eles são dispositivos que operam em um modo chamado Geiger-Müller[50], formados por um conjunto de diodos detectores de fótons por avalanche (APD's - *Avalanche Photo-Diodes*). Esse modo funciona com efeito avalanche e um ganho entre 10^5 e 10^6[51]. Um valor de alta tensão é aplicado reversamente, na ordem de dezenas de volts, calibrado individualmente por dispositivo, para colocá-lo no ponto de operação. Com o ganho alto, quando um fóton chega a um *pixel* do SiPM, o efeito avalanche iniciado o satura e, assim, é possível contar aproximadamente os fótons que chegaram, contando a carga na saída, já que ela é um somatório de todos os *pixels* saturados. Uma observação a ser mencionada é que, independentemente de o pixel ser atingido por um ou por milhares de fótons, a carga na saída será a mesma. Características importantes desse fotodetector podem ser encontradas nas referências [52][53][54][55][56][57][58].



(a) Modelo interno do funcionamento do SiPM.

(b) Sinal característico do SiPM, com osciloscópio em modo de persistência.

Figura 2.4: Modelo do fotomultiplicador de silício e sinal característico[59].
Fonte: Retirado da dissertação de mestrado "The Semiconductor Multiplication System for Photoelectrons in a Vacuum Silicon Photomultiplier Tube and Related Front End Electronics".

O princípio físico para detecção de um fóton utilizado pelo SiPM se baseia no processo de conversão elétrica da interação dele com a matéria, o efeito fotoelétrico.

Por meio de uma junção P-N, com a polarização reversa ocorre o alargamento da região de depleção, que é uma área com baixa concentração de portadores de carga, seja elétron livre ou 'buracos', que é a falta de elétron. Essas cargas geram um campo elétrico nessa região. Como é possível ver na figura 2.5, essa região possui as condições ideais para ocorrer o efeito fotoelétrico, onde os fótons que possuem energia acima da energia de ligação, assim como os fótons que o WSL emite, serão absorvidos, criando o par elétron-buraco.

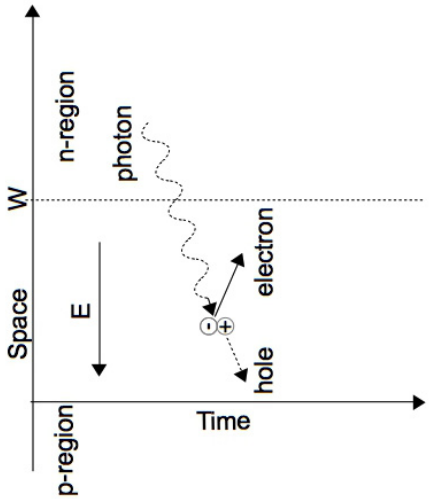


Figura 2.5: Modelo do processo fotoelétrico, ocorrendo na região de depleção[51].
Fonte: Retirado do artigo "Analysis of photon statistics with Silicon Photomultiplier".

Em consequência do campo elétrico, esses portadores são acelerados e, no trajeto, conseguem atingir energia superior à energia de ionização dos átomos semicondutores da região de depleção. Então, nesse processo, colisões inelásticas ocorrem e novos pares elétron-buraco são formados, e, consequentemente, ocorre o aumento do campo elétrico na região, fornecendo mais energia aos portadores criados em seguida, processo conhecido como avalanche.

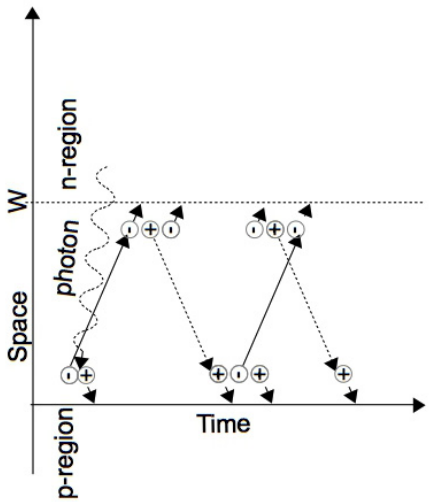


Figura 2.6: Esquema do processo avalanche iniciado pelo efeito fotoelétrico na região de depleção do SiPM[51].
Fonte: Retirado do artigo "Analysis of photon statistics with Silicon Photomultiplier".

Na figura 2.6 é mostrado o efeito avalanche que pode ocorrer. Devido ao 'buraco' ser

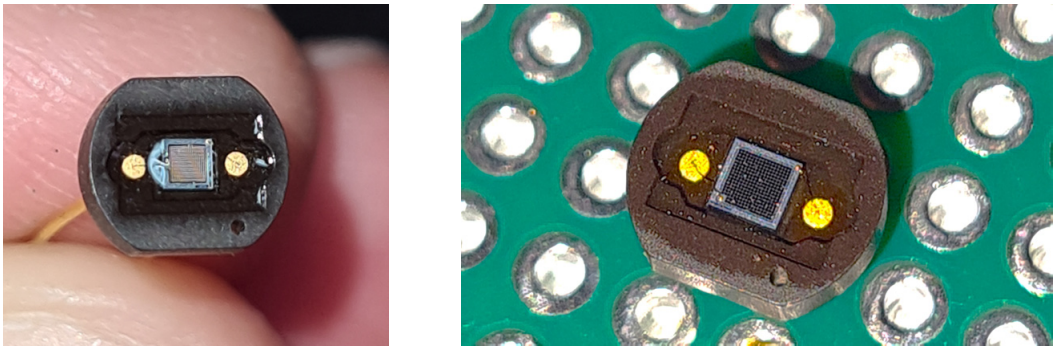
constituído pelo íon, um campo de maior intensidade é necessário para que ele ganhe energia suficiente para ionizar outros átomos no trajeto, campo esse maior que 10^4 V/m[51]. Esse processo demonstrado na imagem ocorre nos SiPMs, onde o efeito avalanche iniciado não é encerrado automaticamente e, para encerrar o processo, é usado um resistor (*quenching resistor*), como mostrado na figura 2.4. A saída do SiPM é uma carga proporcional ao número de *pixels* saturados. A carga de saída de cada *pixel* segue a equação:

$$Q = G \times e \tag{2.2}$$

sendo "Q"a carga final, "G"o ganho do SiPM, "e"a carga elementar do elétron.

Deve-se destacar que a energia térmica pode iniciar um efeito avalanche da mesma forma que um fóton. Esse processo é conhecido como corrente de escuro (*dark current*). Por isso, em geral, usam-se esses transdutores em baixas temperaturas para evitar esse processo. Outra técnica também usada para eliminação desses sinais não desejados se dá pelo uso de coincidência entre planos de cintiladores, já que a probabilidade de dois ou mais planos serem excitados simultaneamente devido à corrente de escuro é muito baixa. Este último foi o método escolhido pelo projeto, já que não se faz necessário o resfriamento do sistema, algo que demandaria grande consumo de energia, aspecto crítico em algumas localizações onde o projeto deverá ser instalado futuramente.

O modelo escolhido de SiPM para o experimento é o S13360-1375CS, produzido pela *Hamamatsu Photonics*. O fotodetector possui 285 pixels com *pitch* de $75\mu m$, em uma área de $1.3mm \times 1.3mm$. Uma foto do modelo escolhido pode ser vista na figura 2.7.

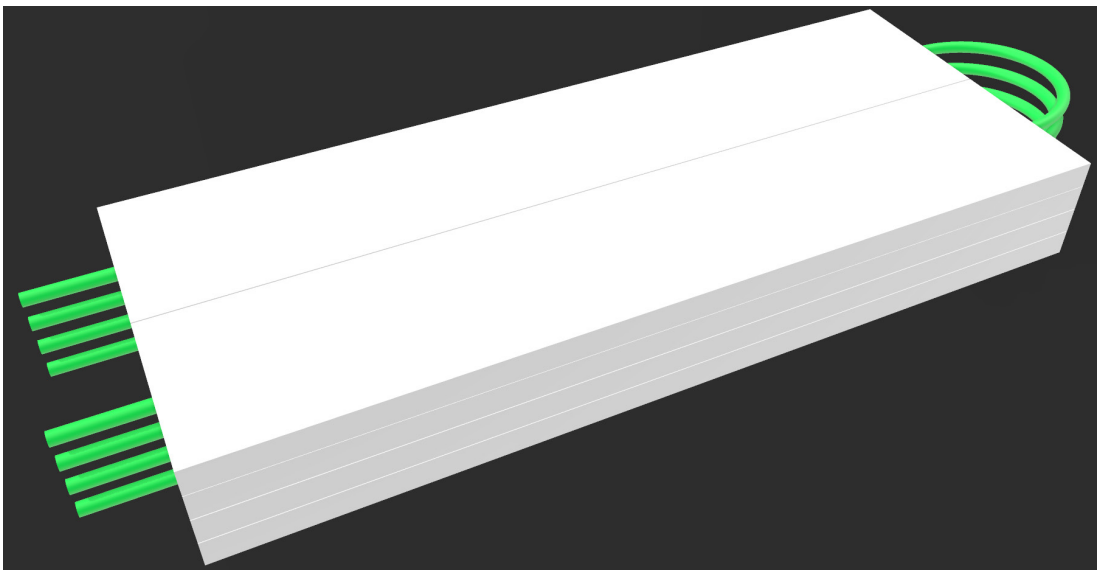


(a) (b) Foco na área sensivel do SiPM.

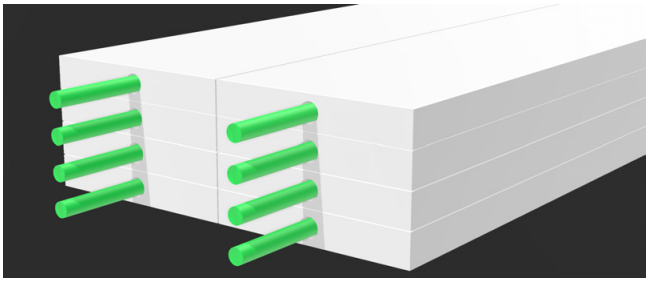
Figura 2.7: SiPM S13360-1375CS da *Hamamatsu Photonics*.

2.3 Grupos de detecção

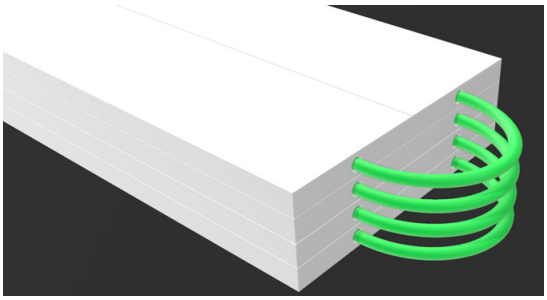
O Experimento CRE4AT foi projetado para ser construído de forma flexível quanto ao número de detectores que serão utilizados. Contudo, um conjunto mínimo de planos de detecção se faz necessário devido à coincidência entre canais. Esse conjunto será chamado de grupo, que é formado por 4 planos de cintiladores um sobre o outro, com dois cintiladores lado a lado em cada plano.



(a) Vista Superior.



(b) Vista Frontal.



(c) Vista Traseira.

Figura 2.8: Grupo de 8 Cintiladores com WSL.

Uma única fibra *Wavelength Shifter* é utilizada por plano, sendo inserida em um cintilador, fazendo a curva no fim dele e entrando no cintilador ao lado até chegar à região frontal, como pode ser visto na figura 2.8. Essa técnica foi utilizada para aumentar a área do plano de detecção, sem a necessidade de aumentar o número de SiPMs por plano. No contexto do experimento, os grupos são chamados por letras e os canais dentro do grupo são numerados.

2.4 Óptica do Experimento e LED UV

Para o projeto funcionar de forma adequada e confiável, uma parte importante do experimento está na mecânica óptica de cada grupo, pois é necessário eliminar a possibilidade de entrada de luz. Para isso, juntamente com a equipe da mecânica do CBPF (LITMEC), foi projetado um conjunto de peças construídas em polipropileno, que são produzidas na fresa da instituição. As peças projetadas possuem uma série de ângulos de 90°, impedindo a entrada de luz e, ao mesmo tempo, mantêm o conjunto de 8 cintiladores unidos, como é possível observar na figura 2.9. Essas peças no contexto do projeto são chamadas de extensores frontal e traseiro. Inicialmente, o projeto foi pensado somente com uma estrutura simples que abrigava o SiPM e era aparafusada no próprio cintilador, Dessa forma, era necessário produzir roscas no material orgânico do cintilador, retirando parte do material.

Esse método se mostrou problemático em virtude de três questões:

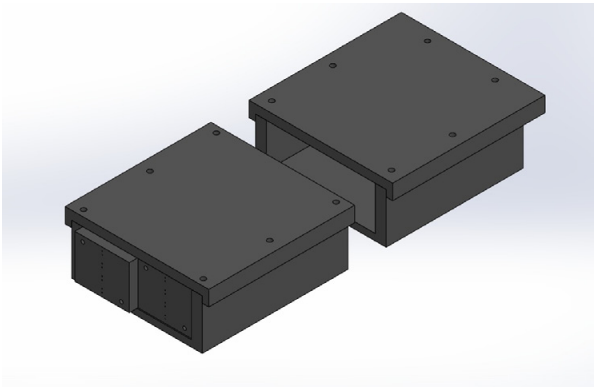
1. Com os parafusos inseridos no material cintilante, parte dos fótons poderia ser perdida devido ao metal, criando uma barreira interna no material;

2. Pelo fato de se considerar a área do plano de material cintilante no cálculo do fluxo de partículas passantes pelo detector, seria necessário levar em conta a porção de material retirada para inserção dos parafusos, para se alcançar precisão na medida;

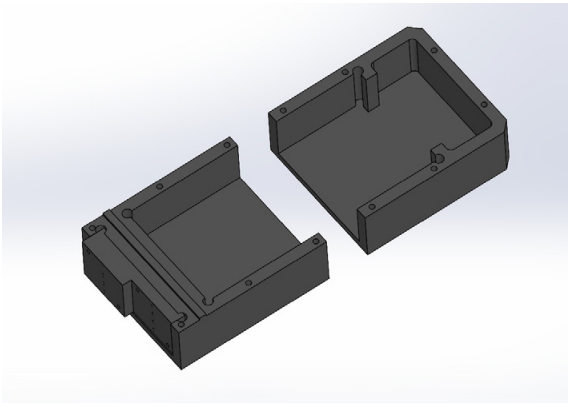
3. A estrutura era frágil. A opção unindo os 4 canais do grupo em uma única estrutura possui melhor resistência mecânica para o grupo.

(a) Primeira versão da óptica aberta.

(b) Primeira versão da óptica montada em um cintilador.



(c) Conjunto óptico atualizado com tampa fechada.



(d) Conjunto óptico atualizado sem tampa.

Figura 2.9: Conjunto óptico desenvolvido para o experimento CRE4AT.

Além dessas funções, na parte frontal, uma peça aparafusada chamada de *holder* no contexto do experimento, abriga tanto os 4 SiPMs, um para cada um dos planos, de um lado, quanto 4 LEDs (*Light Emission Diodes*) ultravioleta de outro (LED UV), como pode ser visto na figura 2.10.

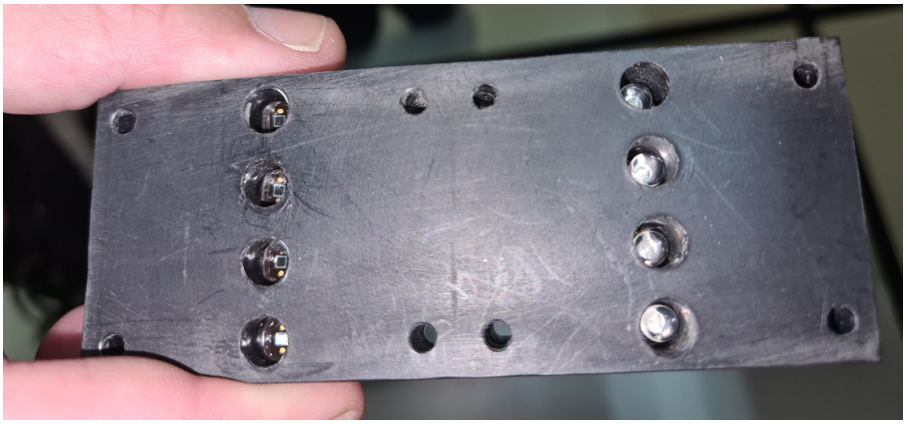


Figura 2.10: Holder do experimento.

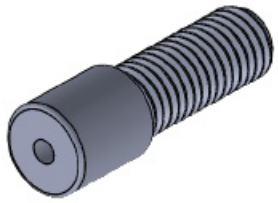
O LED utilizado é o modelo VAOL-5GUV8T4 da *VCC Optoelectronics*. Esse modelo foi escolhido devido à baixa intensidade de luz emitida, 80 mcd (milicandelas), e, ao mesmo tempo, possuir fótons de alta energia, que, quando absorvidos pela WSL, serão reemitidos em maior quantidade. O comprimento de onda emitido por este modelo é de 385nm, na faixa do UVA.

O LED no contexto do projeto é utilizado somente para calibração do sistema, verificando se a quantidade de emissões disparadas é recebida corretamente, assim como *crosstalk* entre canais. Esse protocolo será, posteriormente, discutido mais detalhadamente.

Para extrair o sinal dos SiPMs e enviar os pulsos aos LEDs que estão contidos dentro do *holder*, foi projetada uma placa simples, que é aparafusada junto com o *holder* ao extensor. A figura 2.11 mostra uma foto da placa. Como é possível observar, uma região entre a solda do SiPM e o parafuso foi deixada livre no projeto da placa para identificar o SiPM que está naquela localização. Com essa identificação, é possível rastrear todas as informações do fabricante relacionadas àquele componente que foi utilizado, como a tensão de operação, corrente de escuro, entre outras informações que o fabricante passa ao comprador após os testes individuais. O projeto completo dessa placa pode ser encontrado no apêndice B.

Figura 2.11: Placa utilizada para envio de sinal aos LEDs e extrair sinal dos SiPMs.

Para realizar o isolamento dos canais à entrada de luz externa e, ao mesmo tempo, garantir que a fibra esteja posicionada de forma correta na área sensível do SiPM, mantendo o mesmo distanciamento em todos os planos, o parafuso óptico foi desenvolvido. Na figura 2.12 é possível ver tanto o desenho quanto a versão produzida na fresa.



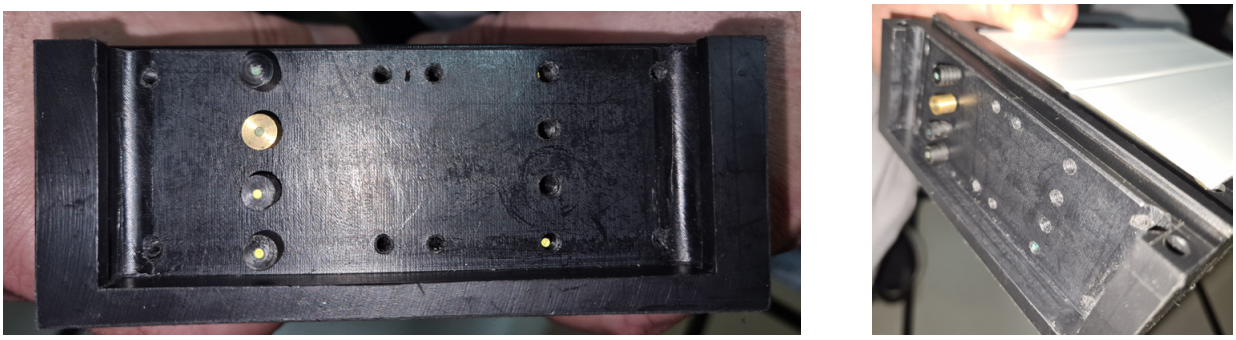
(a) Modelo 3D do projeto.

(b) Parafuso óptico produzido.

Figura 2.12: Parafuso óptico projetado.

Para fixar a fibra no parafuso óptico, impedindo que haja movimentação da WSL ao aparafusar, uma cola óptica é usada. O apêndice R mostra como todo o processo é realizado. A cola escolhida é a cola óptica 601 A+B, produzida pela *POLYTEC PT*. A cola é um adesivo epóxi desenvolvido especialmente para trabalhos com fibra por ser transparente à passagem de fótons, não interferindo nas medições que detector realizar. Após o processo de colagem, o parafuso é usinado para planificar, removendo o excesso de cola. Imagens da fibra colada ao

parafuso óptico presos ao *holder* são mostradas na figura 2.13. Algumas versões do parafuso foram produzidas em polipropileno (PP) e outra em latão, para avaliar a dilatação dos materiais. Para montagem dos experimentos, foi decidido produzir em PP.

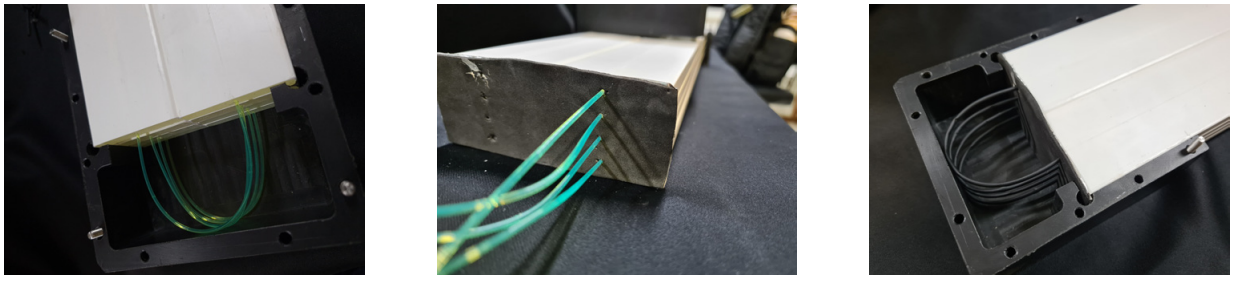


(a) Vista Frontal.

(b) Vista Laterial.

Figura 2.13: Extensor com os parafusos óticos e WSL coladas.

Para evitar escapamento da luz de um canal para os adjacentes, foi acrescentada, na região traseira de cada grupo, uma camada preta com uma fita adesiva de alumínio. As WSLs também foram envolvidas com espaguetes termorretráteis, como é visto na figura 2.14.



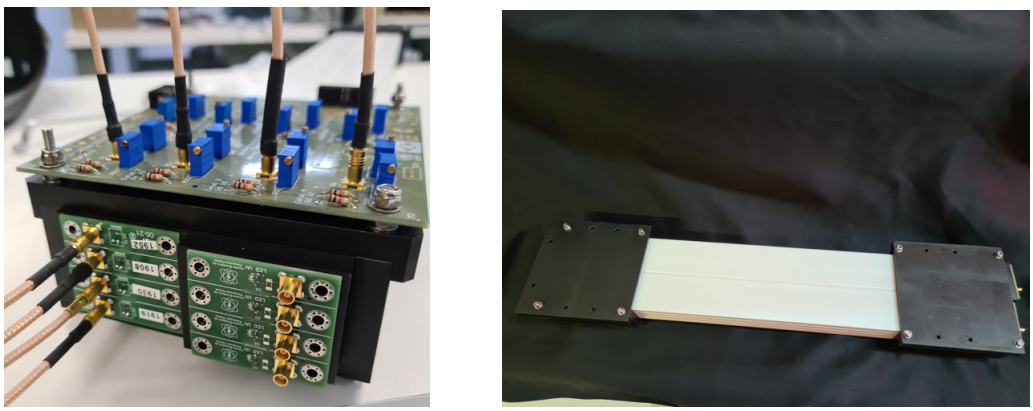
(a) grupo montado sem proteção.

(b) Inclusão da proteção na parte traseira dos cintiladores.

(c) Grupo com proteção traseira.

Figura 2.14: Vista superior da parte traseira do grupo montado com inclusão de proteção.

O projeto completo das peças aqui citadas pode ser encontrado no apêndice C e o grupo montado na figura 2.15.



(a) Vista Frontal do grupo montado.

(b) Vista Superior de grupo montado.

Figura 2.15: Grupo montado.

2.5 Eletrônica de Front-end (FEE)

Seguindo a linha de princípio dos grupos de detecção, a eletrônica de *front-end* desenvolvida nesse trabalho foi projetada com 4 canais no *software Altium Designer* e o projeto completo pode ser encontrado no apêndice D. O layout foi elaborado pelo técnico do CBPF Fernando Souza. Como é possível observar no diagrama de blocos da figura 2.16, cada canal pode ser dividido em 3 partes e a placa possui uma parte opcional, que é comum a todos os canais:

1. O integrador, que, no contexto do projeto, faz o papel de *shaper*;
2. O amplificador;
3. O discriminador;
4. Opcional: conversores DC-DC de alimentação.

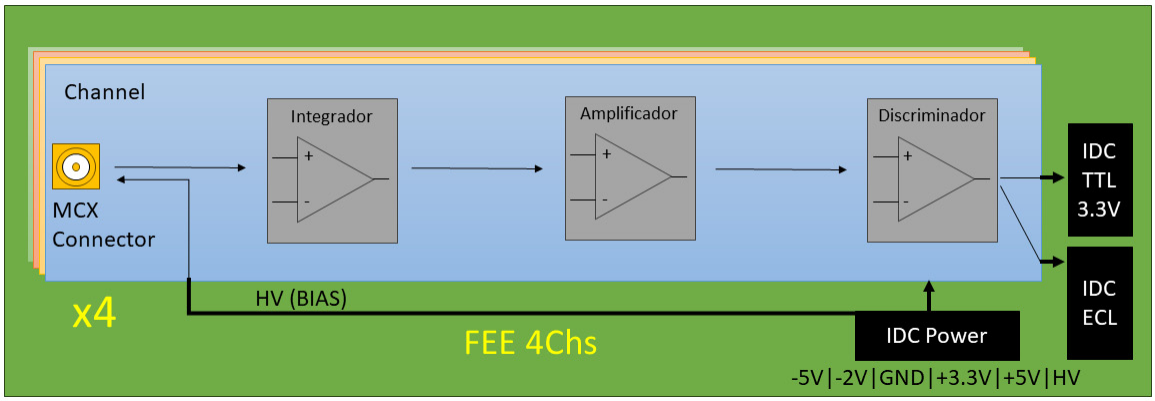


Figura 2.16: Diagrama de blocos da Eletrônica analógica de 4 canais.

Além das partes de cada canal que serão detalhadas, foi decidido utilizar um conector MCX para realizar a ligação de cada SiPM à placa, um conector IDC de 16 vias para receber toda a alimentação necessária e dois conectores IDC de 10 vias para saída de sinal, sendo um para sinal TTL de 3.3V e outro para sinal ECL. Uma foto da eletrônica produzida pode ser vista na figura 2.17.

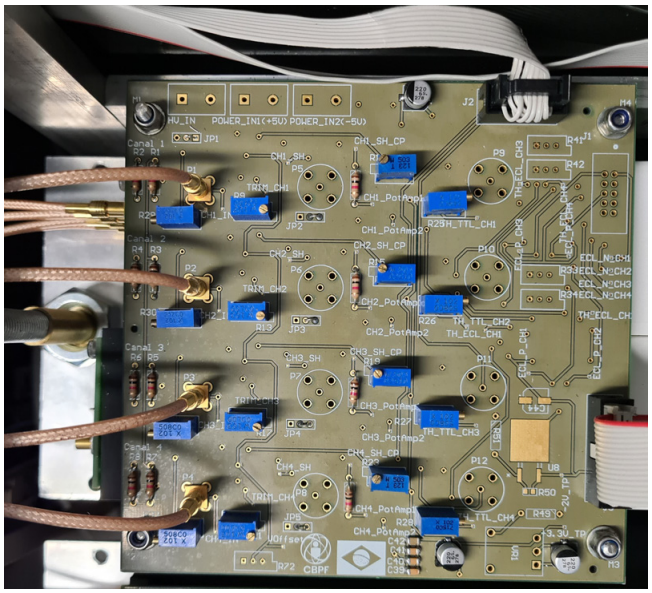


Figura 2.17: Foto da eletrônica de *front-end*.

2.5.1 Shaper

O circuito integrador é o primeiro estágio onde o sinal do SiPM é tratado. Esse circuito é responsável por reduzir o tempo do sinal de saída do fotomultiplicador de silício, que tem um decaimento exponencial e dificulta o uso para contagem de múons cósmicos, reduzindo, desta forma, o tempo morto do detector. O tempo morto do detector é caracterizado pelo período em que o sinal gerado por uma partícula passante é tratado e, logo, novos sinais podem ser perdidos. O projeto foi realizado utilizando o circuito integrado (CI) AD8039 da fabricante *Analog Devices*, que possui 2 amplificadores operacionais (AmpOp) por CI, facilitando a montagem já que tanto o integrador quanto o amplificador de cada canal utilizam esse componente.

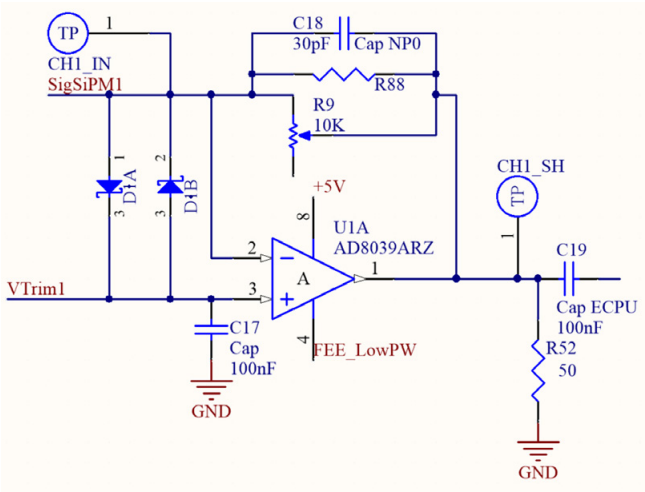


Figura 2.18: Circuito Integrador.

O tempo de duração do pulso de saída pode ser controlado por intermédio do potenciômetro na realimentação do circuito, que, nesse canal, é o R9, como mostrado na figura 2.18. O controle é realizado por meio da descarga do capacitor, que se dá pela constante de tempo $R \times C$. Durante os testes, o valor escolhido para esse potenciômetro, em todos os canais, foi de 900Ω, visto que a maior parte dos pulsos de saída estão dentro de 400ns, alguns chegando aos 500ns. Um resistor também foi inserido na realimentação, caso se decida, futuramente, fixar o valor resistivo e, assim, tornar o potenciômetro desnecessário.

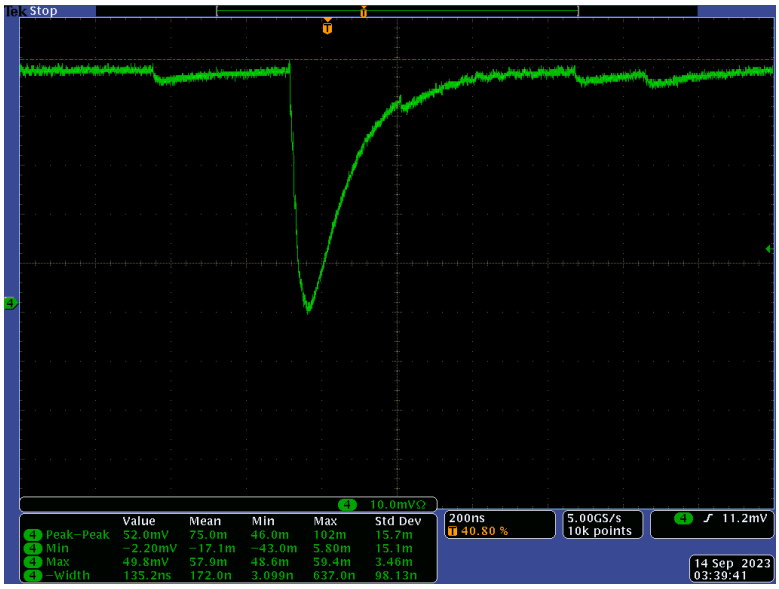
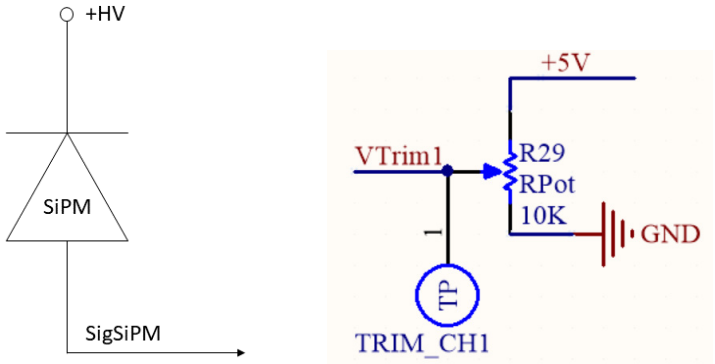


Figura 2.19: Sinal característico do SiPM na saído do estágio de *shaper*.

A entrada de sinal, SigSiPM1 na figura 2.18, é feita por meio da entrada inversora do amplificador. Logo, o sinal resultante na saída possui uma defasagem de 180° em relação à entrada, como é possível observar na figura 2.19. Por conta da realimentação negativa, o circuito possui uma característica conhecida, o curto-circuito virtual, onde a tensão que é aplicada em uma das entradas aparece na outra do AmpOp. Como cada SiPM possui uma tensão de operação, essa característica foi usada para ajustá-la individualmente, uma vez que uma única fonte é utilizada para alimentar todos os SiPMs. Com isso, na entrada não inversora é aplicada uma tensão (*Trim*) por meio de um potenciômetro, que é repetida na entrada inversora, provocando o decremento da alta tensão que é aplicada no catodo do SiPM.



(a) Esquema de alimentação do SiPM.

(b) Potenciômetro onde a tensão de ajuste canal a canal é realizada.

Figura 2.20: Esquema de alimentação do SiPM com Trim.

Para evitar variações (*ripple*) na alimentação da tensão de operação (BIAS) do SiPM, foi inserido um filtro de 2ª ordem em cada canal. O *jumper* JP1 colocado entre a entrada da alta tensão e o sinal de *ground* serve para aplicações específicas de teste, quando um sinal de um gerador de função externo é aplicado à entrada do circuito.

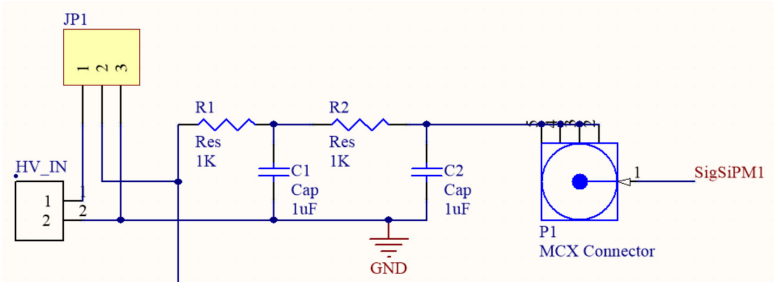


Figura 2.21: Filtro na linha de alimentação HV.

A eletrônica, com o objetivo de alcançar flexibilização, foi projetada para trabalhar com dois tipos de alimentação: simétrica e assimétrica. No terminal 4 do amplificador operacional, onde a tensão negativa é aplicada, um *label* "FEE_LowPW"pode ser visto. A tensão aplicada é selecionada por intermédio de uma chave (JP6), vista na figura 2.22, que irá alimentar o circuito ou com -5V, ou com *ground* (GND).

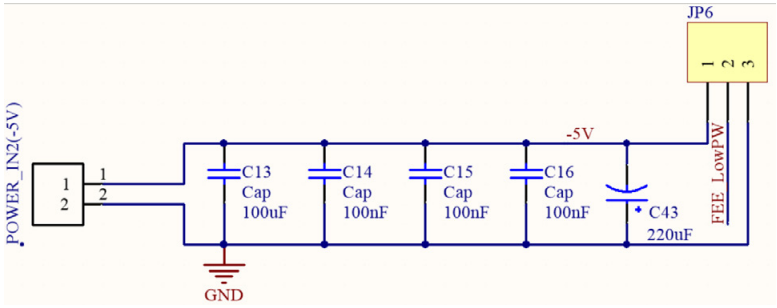


Figura 2.22: Circuito de alimentação -5V com chave de seleção para GND.

Os símbolos TP no esquemático são pontos de teste colocados na placa. Para realizar o ajuste do potenciômetro de realimentação, mede-se a resistência dele entre o TP "CH1_IN"e o "CH1_SH". O sinal de saída do *shaper* também pode ser monitorado no ponto de teste "CH1_SH"e uma resistência de 50Ω foi colocada para casamento de impedância. O sinal é desacoplado para o segundo estágio (amplificador), por meio do capacitor C19, no canal 1.

2.5.2 Amplificador

Após o ajuste de tempo do sinal, por intermédio do integrador, o sinal é amplificado e a sua fase alterada novamente em 180°, pelo circuito amplificador inversor, mostrado na figura 2.23.

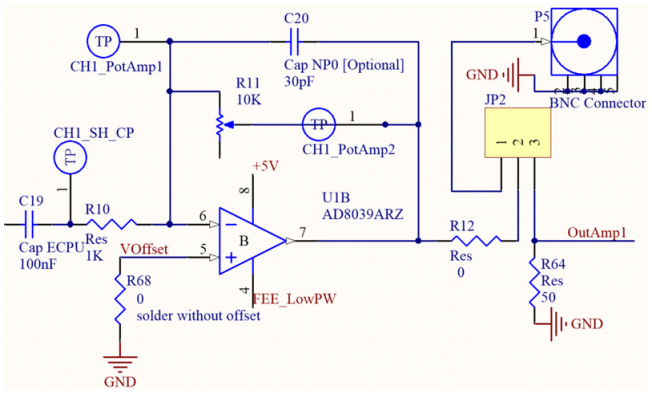


Figura 2.23: Circuito Amplificador da FEE.

Nesse estágio, o controle de ganho é ajustado pelo potenciômetro R11 e seu valor é medido nos pontos de teste CH_PotAmp1 e CH_PotAmp2. O capacitor C20 foi adicionado, caso algum ajuste de largura fosse necessário, o que não ocorreu. O ganho do circuito é calculado pela relação $\frac{R11}{R10}$. O valor escolhido para esse potenciômetro foi de 4kΩ, logo, um ganho de 4 vezes a saída do *shaper*, deixando o sinal com um nível de tensão suficientemente longe do ruído eletrônico e, ao mesmo tempo, sem saturar o amplificador, quando pulsos maiores são obtidos. A figura 2.24 mostra a curva característica na saída do amplificador.

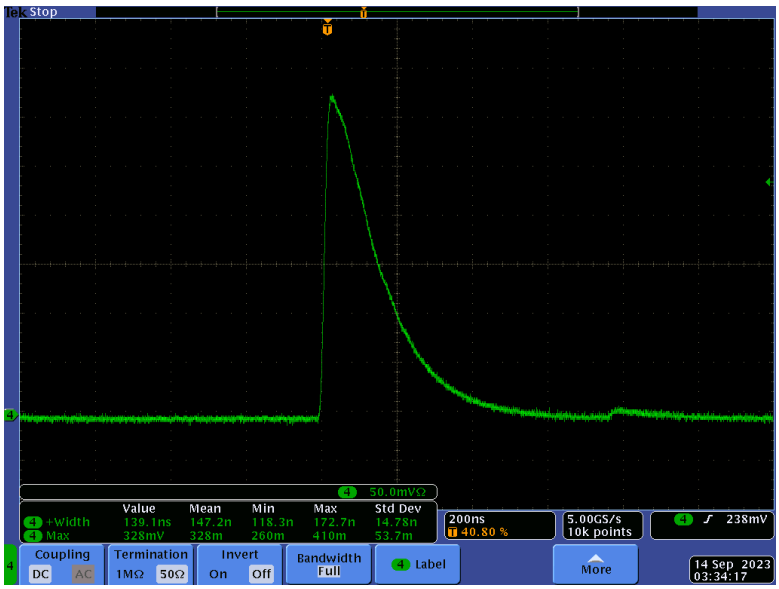


Figura 2.24: Sinal na saída do estágio amplificador.

Caso o circuito seja alimentado de forma assimétrica, um valor de tensão diferente de 0v é necessário na linha de sinal, tendo em vista que, no primeiro estágio, o pulso positivo proveniente do SiPM é invertido. Essa tensão (*offset*) é colocada por meio de um potenciômetro, mostrado na figura 2.25, na entrada não inversora do amplificador operacional e, neste caso, o resistor de 0Ω não deve ser soldado.

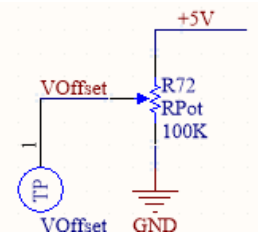


Figura 2.25: Circuito de offset opcional para o amplificador.

Uma saída do sinal analógico foi inserida por meio da chave JP2 e um resistor em série de 50Ω pode ser colocado para casamento de impedância em R12.

2.5.3 Discriminador

Com o escopo de medir a quantidade de partículas que passam pelos cintiladores e eliminar ruídos eletrônicos do sinal analógico, gerando um pulso digital para ser enviado à eletrônica de aquisição, um circuito discriminador foi projetado com o comparador ADCMP600BRJZ, produzido pela *Analog Devices*. O comparador foi escolhido por ser de alta velocidade, compatível com a lógica TTL/CMOS de 3.3V e já possuir histerese integrada.

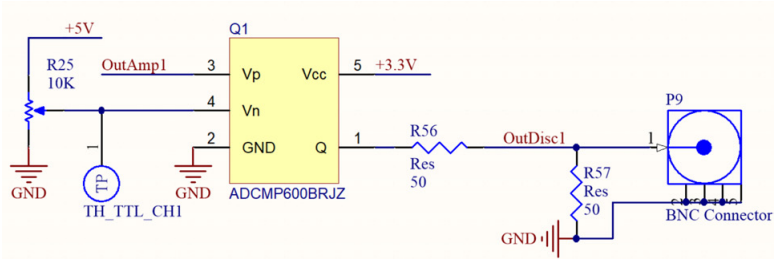


Figura 2.26: Circuito discriminador da FEE.

Um ponto de teste foi adicionado para medir o valor de limiar (*threshold*) utilizado para comparar ao sinal e gerar o pulso TTL na saída. Um resistor de 50Ω (R56) foi adicionado para casamento de impedância com o osciloscópio e para uso com contadores NIM. A sua utilização não é obrigatória com o sistema de aquisição desenvolvido, e, neste caso, um resistor de 0Ω pode ser usado. Caso a saída precise ser de tensão menor, um segundo resistor (R57) foi acrescentado, possibilitando dividir a tensão de saída do comparador. O pulso de saída OutDisc1 está ligado aos conectores BNC P9 e IDC J2.

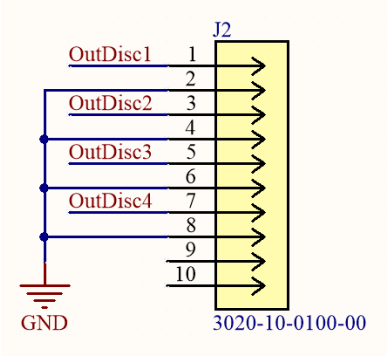


Figura 2.27: Conector IDC saída TTL 3.3V.

O pulso de saída característico pode ser visto na figura 2.28.

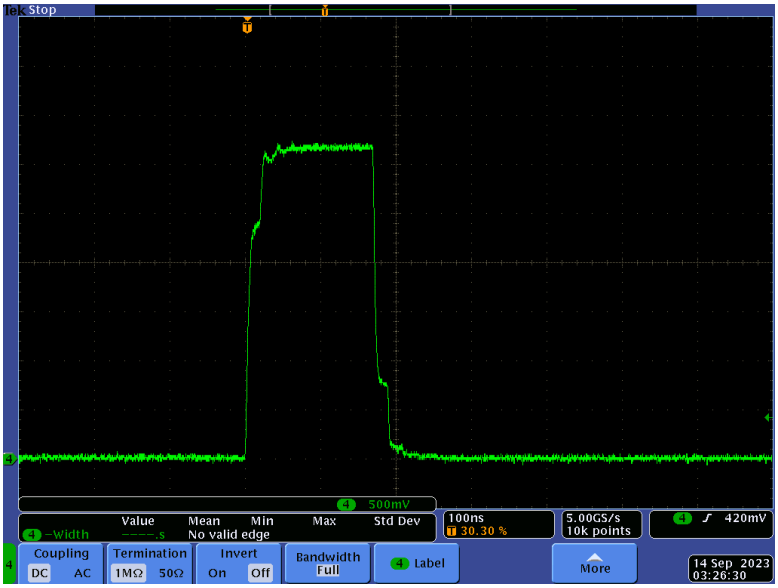
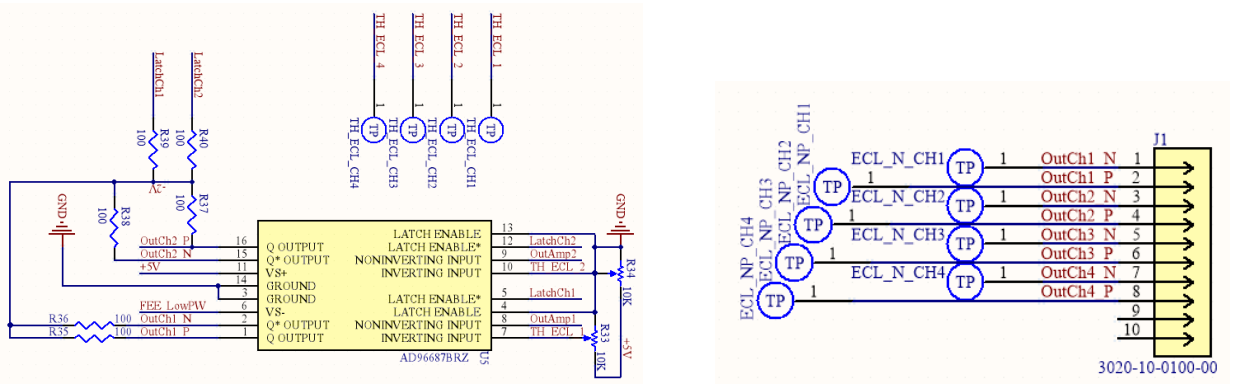


Figura 2.28: Sinal de saída discriminado.

Outra possibilidade que a placa possui é a saída na lógica ECL(*emmiter-coupled logic*). Neste caso, o comparador utilizado é o modelo AD96687BR, da fabricante *Analog Devices*, de

alta velocidade e baixa tensão. Este modelo foi escolhido por possuir dois comparadores no mesmo CI e ser compatível com outra eletrônica já produzida no laboratório, utilizada com uma MAPMT (*multianode photomultiplier*).



(a) Discriminador ECL. (b) Conector IDC da saída ECL.

Figura 2.29: Circuito do comparador ECL.

2.5.4 Conversores DC-DC de alimentação

A alimentação da FEE no experimento é provida pelo sistema e aquisição. Entretanto, durante o projeto, ela foi projetada também para uso sem necessidade de um sistema externo (*standalone*). Assim, foram inseridos conversores DC-DC para que que, por meio da alimentação de +5V e -5V, gerassem as tensões de +3.3V usada pelo comparador TTL e de -2V usado pelo comparador ECL. A tensão de polarização do SiPM, nesse caso, teria que ser alimentada externamente, assim como a alimentação simétrica por 3 terminais.

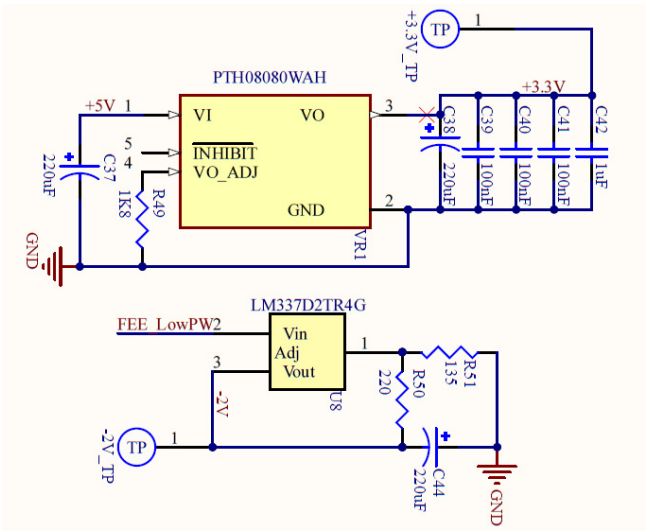


Figura 2.30: Conversores DC-DC para uso da placa no modo *standalone*.

O conversor escolhido para realizar a redução da tensão para +3.3V foi o PTH08080WAH, produzido pela *Texas Instuments*. Esse DC-DC possui saída ajustável, não isolada, com corrente máxima de 2.25A. O ajuste da tensão é feito por um resistor, ligando o *ground* ao terminal V_o Adjust. O resistor escolhido foi de 1,8kΩ, que resultou em uma tensão de saída de aproximadamente 3,32V.

Para alimentação de -2V, o conversor LM337D2TR4G, também produzido pela *Texas Instuments*, com corrente de saída de 100mA, foi escolhido por já ser usado em outros projetos do

laboratório, que utilizam comunicação ECL. Essa alimentação só é necessária caso o comparador com saída ECL seja utilizado na eletrônica de *front-end*.

2.6 Sistema de Aquisição de Dados (DAQ)

A placa do sistema de aquisição de dados foi desenvolvida no *software EasyEDA*, com o intuito de tornar o projeto autossuficiente e com baixo consumo de energia. A placa do DAQ, além de prover a aquisição e tratamento dos dados vindos das FEEs, também provê ao sistema toda a alimentação necessária, realiza aquisição de dados dos sensores ambientais, controla o sistema de calibração com LEDs UV, realiza a medida de tempo e salva os dados. Um diagrama de blocos mostrando como o experimento CRE4AT é interligado pode ser visto na figura 2.31. Como os experimentos já produzidos utilizaram 3 grupos, o diagrama mostra somente os grupos A até C.

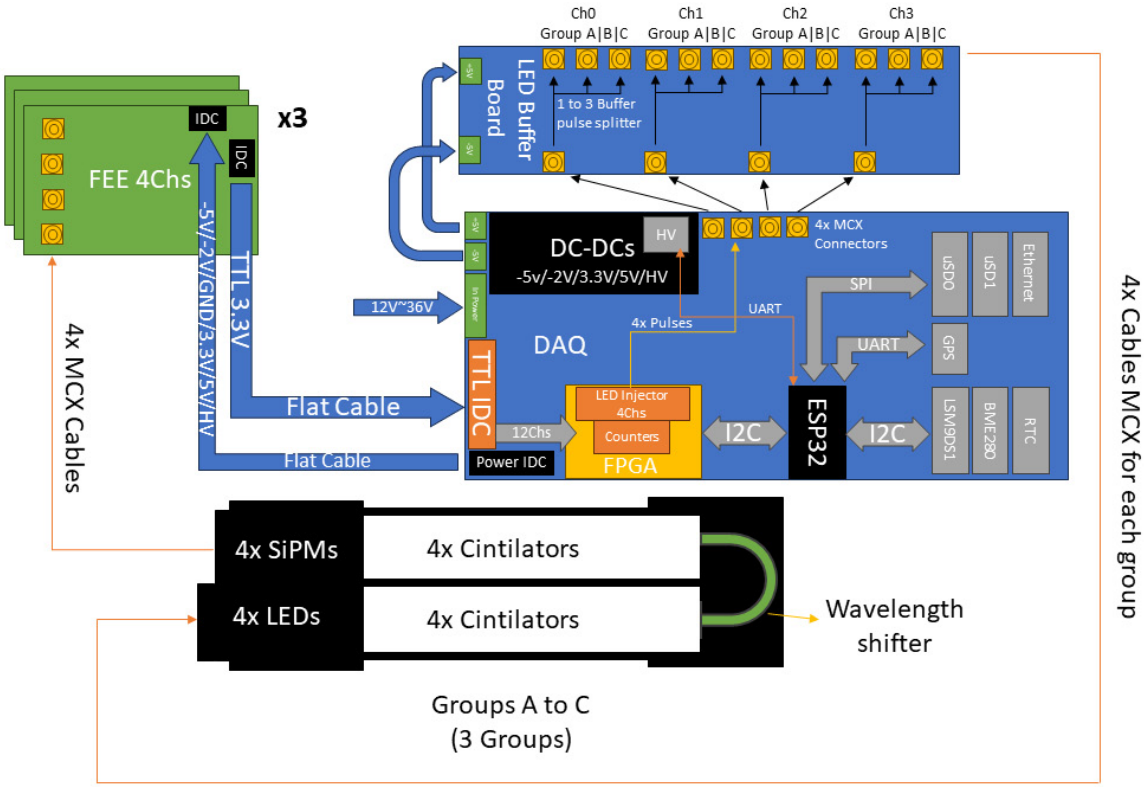
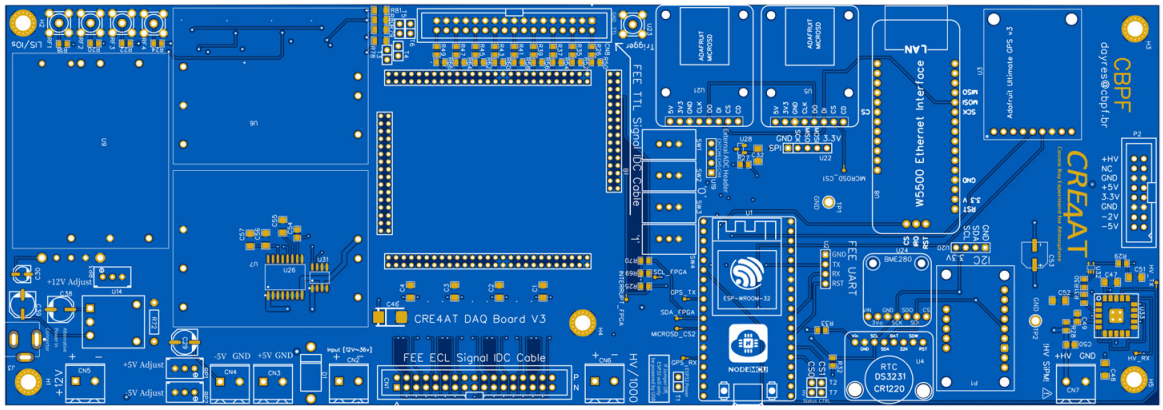
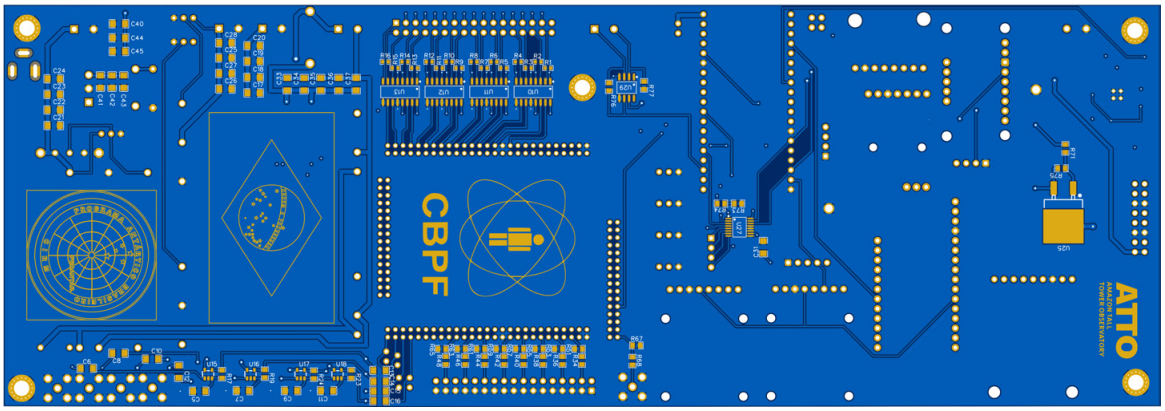


Figura 2.31: Diagrama de blocos do sistema de aquisição de dados inserido no experimento.

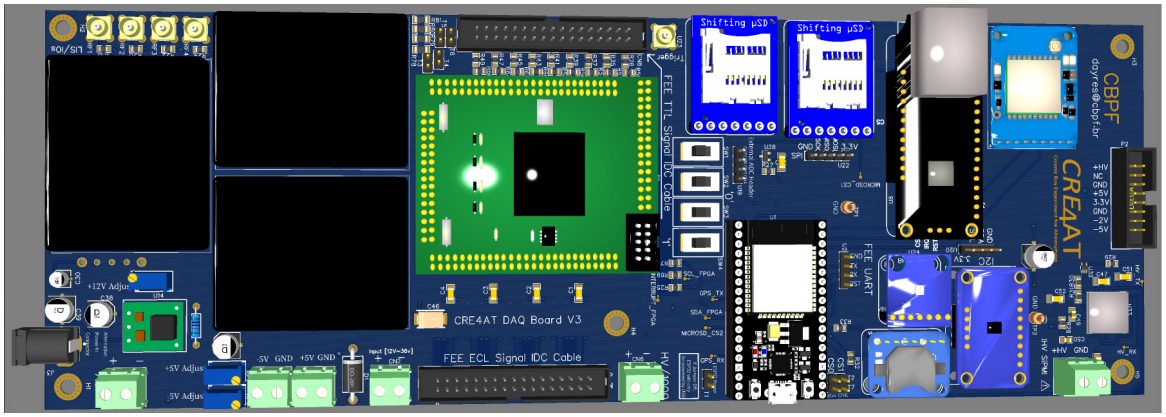
Imagens 2D e 3D do projeto da placa, que possui 6 camadas, pode ser visto na figura 2.32. O projeto esquemático e layout da placa estão disponíveis no apêndice E.



(a) Visão superior da placa.



(b) Visão inferior da placa.



(c) Visão 3D da placa com componentes.

Figura 2.32: Visão da placa do sistema de aquisição de dados.

O DAQ possui 16 canais ECL e 17 canais de entrada TTL 3.3V, com possibilidade de soldar um divisor de tensão, caso o sinal seja de 5V. Além dessas entradas, possui uma porta de entrada/saída de *trigger* e 4 saídas de pulso para o sistema de injeção de luz, que podem ser convertidas para entradas/saída genéricas. Para explicar melhor cada parte do sistema de aquisição, pode-se dividi-lo nos seguintes itens:

1. Bloco de alimentação;

2. FPGA;

3. Microcontrolador ESP32;
4. Sensores ambientais e comunicação;
5. Sistema de injeção de Luz;
6. Comandos de controle do sistema.

2.6.1 Alimentação

Como o DAQ é responsável pela distribuição da alimentação a todo o sistema, alguns conversores DC-DC foram inseridos ao projeto. A placa possui 2 conversores principais de maior potência, que são responsáveis pela alimentação de +5V e -5V. O conversor escolhido para gerar essas alimentações foi o UHE-5/5000-Q12-C, produzido pela *Murata Manufacturing*. Com a saída fixa de 5V, o conversor possui saída isolada, ajuste fino da tensão de saída e corrente de 5A. Com uma alimentação de entrada de 9V até 36V, os dois conversores, sendo um deles com a saída ligada inversamente ao sistema, gerando -5V, isolam todo o sistema de problemas de alimentação na entrada. Também possuem circuitos de proteção contra curto-circuito e alimentação fora do especificado, desligando a saída caso aconteça um desses casos.

Dois outros conversores, produzidos pela mesma fabricante e com saídas também isoladas são usados para fazer a saída de +12V. Como dois modelos estavam disponíveis no laboratório, ambos foram inseridos na placa, sendo eles UHE-12/2500-Q12-C e UQQ-12/8-Q12PB-C. Com o primeiro modelo é possível fornecer até 2,5A de corrente e, com o segundo, 8A, alimentando-os com uma tensão de 12V até 36V. Esses modelos são complementares, sendo somente um deles soldado na placa. Essa saída de alimentação é opcional, sendo usada somente quando uma fonte de alta tensão é necessária.

Esse conjunto de conversores mencionados acima são alimentados com a entrada de energia do sistema e, por isso, modelos com maiores proteções e, consequentemente, de maior custo, foram escolhidos. Já os modelos de 3.3V e alta tensão para o SiPM são alimentados pelo conversor DC-DC de +5V e, no -5V, o conversor para -2V. Os conversores de +3.3V e -2V são os mesmos utilizados na FEE, utilizando-se o mesmo projeto.

Para alimentação de polarização dos SiPMs, o conversor DC-DC C11204-02, produzido pela *Hamamatsu Photonics*, foi utilizado. Esse conversor foi escolhido devido ao autocontrole da saída em função da temperatura ambiente, compensando, assim, o ganho variável da SiPM em função da temperatura a qual é submetida. Ele é controlado pelo protocolo UART e, por meio dessa comunicação, é possível executar diversas operações, como, por exemplo, ligar/desligar a saída e configurar a reta de ajuste da tensão de saída pela temperatura. Em muitos locais onde o experimento será instalado, a variação térmica ou será controlada ou não deverá ser muito alta. Contudo, em um local em que a instalação desse projeto está sendo planejada, esse controle será essencial: o módulo antártico CRIOSFERA 1. Este local é alimentado atualmente somente por painéis solares durante o verão e, por um banco de baterias durante o inverno. Dessa forma, é inviável energeticamente possuir um sistema de aquecimento para controle de temperatura. Logo, o projeto precisa ser responsável por esse controle do ganho, devido à grande variação térmica ao longo do ano e, ao mesmo tempo, ser energeticamente eficiente.

2.6.2 FPGA

Responsável pela aquisição dos dados da FEE e controle de disparo do sistema de injeção de luz, o FPGA escolhido para realizar essas tarefas é vendido montado em um mezanino (*Core Board*), modelo CoreEP4CE10, produzido pela *Waveshare Electronics*, que contém Cyclone EP4CE10F17C8, fabricado pela *Intel Corporation*, um cristal de *clock* de 50MHz e uma memória

de armazenamento EPCS16 da mesma fabricante do FPGA. A vantagem dessa escolha reside na montagem e projeto da placa, já que o chip exige um controle da alimentação, sendo necessário um acréscimo de dezenas de capacitores e muitas vezes de soldagem de um chip BGA (*Ball Grid Array*), que requer equipamentos específicos e possui um processo que necessita maior atenção. No caso de falha, também é possível destacar e colocar um novo mezanino de forma relativamente fácil e rápida.

A *firmware* foi desenvolvida com base em outros projetos já realizados no laboratório. Projetada em forma de blocos interligados, o conceito aqui utilizado facilita o entendimento do processo. Na parte de aquisição dos dados da FEE, os pulsos são tratados, colocando-os em uma largura fixa, e, depois, esses pulsos são contados por um tempo específico. Após o tempo de contagem ser atingido, um pulso de fim do processo é enviado ao microcontrolador e os valores dos contadores são salvos em um banco de registradores, disponível somente para leitura de forma externa, via protocolo I2C. A parte de controle de injeção de luz é realizada por meio de um ajuste da frequência de disparo, da largura do pulso e dos canais que estão ligados, sendo todos esses parâmetros controlados individualmente, canal a canal, pelo banco de registradores de controle.

Na figura 2.33, o projeto principal pode ser observado. Ele pode ser dividido nos seguintes blocos:

1. Bloco de PLL: Responsável pela aumento e diminuição do *clock* de entrada, gerando os *clocks* de 100kHz e 200MHz usados no bloco de injeção de luz, em fase com *clock* de entrada;

2. Bloco I2C escravo: Responsável pela comunicação do FPGA com o microcontrolador;

3. Bloco de memórias: Responsável por controlar a entrada e saída de dados. Todo o processo é feito utilizando-se dois bancos de memórias:

   (a) Um banco de memória, habilitado para leitura e escrita por meio do protocolo I2C, é responsável pelo ajuste dos parâmetros do sistema;

   (b) O outro, habilitado somente para leitura, é responsável pelo armazenamento dos valores dos contadores após o tempo de contagem configurado ser atingido.

4. Bloco de controle: Responsável por verificar os parâmetros de tempo salvos na memória de controle e passar aos outros blocos as mudanças. Também realiza a verificação de tempo da contagem decorrida, enviando um pulso de fim ao bloco de contagem, quando atinge o tempo configurado na memória (pulso RELEASE) e outro pulso ao microcontrolador (pulso EspSync). O tempo é configurado em segundos em 2 registradores de 1 byte, podendo então ser utilizado desde 1s até 65535s;

5. Bloco de *Latch*: Responsável por receber os pulsos discriminados, vindos das eletrônicas de *front-end*, que possuem largura variáveis, e gerar um pulso de saída de largura fixa, sendo essa largura definida por um registrador de um *byte* [0-255], com passos de 20ns;

6. Bloco de coincidências: Responsável por fazer todas as combinações (lógica AND) de dois canais, três canais e a dos quatro canais de cada grupo. Além destas, também são realizadas todas as possíveis combinações quádruplas de dois canais adjacentes de um grupo com dois canais adjacentes de outro grupo. Por exemplo canais 0 e 1 do grupo A com canais 2 e 3 do grupo C. Essas coincidências entre grupos serão usadas para procura do fator do ângulo zenital;

7. Bloco de contadores: Responsável pela contagem de todos os pulsos dos individuais de cada grupo mais as coincidências realizadas entre eles. Quando o pulso de fim de contagem é recebido, os valores são salvos na memória e os contadores são zerados. Cada registrador possui 24bits para contagem, possibilitando a contagem de 16777215 pulsos dentro do tempo ajustado;

8. Bloco adaptador para registrador A: Como a comunicação I2C é realizada enviando blocos de 1 byte por vez, o banco de registadores possui esse tamanho por registrador, logo, uma conversão é necessária, pois os contadores possuem 24 bits (3 bytes) cada. Esse bloco é responsável por separar o valor registrado em cada contador em 3 grupos de 1 byte, que serão salvos em 3 registradores consecutivos na memória;

9. Bloco gerador de pulso: Responsável por controlar o disparo de cada LED. Dentro desse bloco, o *clock* de 100kHz é reduzido em *clocks* de 50kHz, 25kHz, 10kHz, 5kHz e 1kHz. Um registrador individual para cada um dos 4 canais mais um canal de *trigger* seleciona qual das 6 frequências mencionadas será a taxa de disparo do pulso de saída. Em outro registrador individual, é selecionada a largura do pulso de saída e uma lógica idêntica ao do *latch* é usada para esse processo, sendo que o passo de controle nesse caso é de 5ns. E, por último, 5bits de um registrador controlam se cada uma das saídas está habilitada.

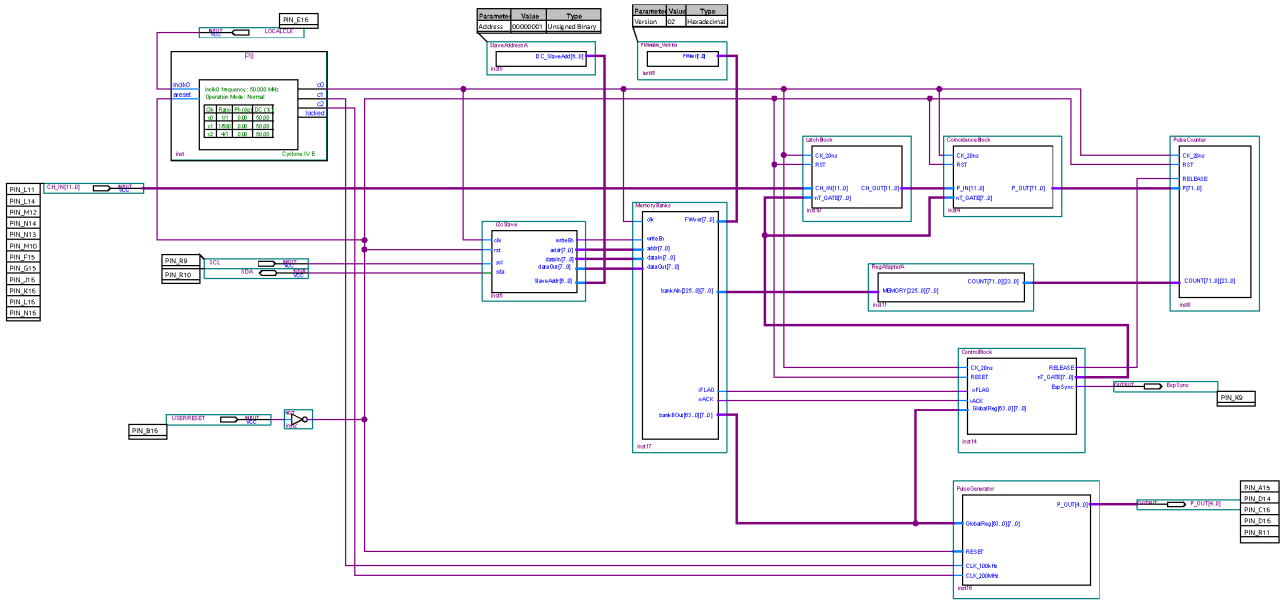


Figura 2.33: Projeto da *firmware* do FPGA.

O projeto completo da *firmware* pode ser encontrado no apêndice F.

2.6.3 Microcontrolador ESP32

O ESP32 é um microcontrolador da fabricante *Espressif Systems*, que trabalha com 3.3V. Ele apresenta a vantagem de possuir 2 núcleos que suportam uma frequência de 240MHz e incluir comunicação *wireless* e *bluetooth* integrados. Ele ainda possui um outro núcleo de baixo consumo, que usa somente 150μA, que pode ser usado para ligar os dois núcleos principais e esse núcleo também possui acesso à comunicação I2C. Com suporte a duas linhas I2C, duas linhas SPI e 3 UART em *hardware*, esse microcontrolador é responsável pela comunicação com o FPGA em uma linha I2C dedicada, com clock de 400kHz, seja para comandos de configuração, seja para leitura dos valores registrados nos contadores, comunicação com os sensores, controle

e monitoramento conversor DC-DC, que polariza os SiPMs (HV), aquisição de tempo pelo GPS e RTC, disparo da calibração do sistema com o injetor de luz e também é responsável por salvar os dados nos cartões microSD e envio dos dados por USB, *Ethernet* ou *Wireless*. Assim como o FPGA, o microcontrolador foi utilizado na placa como um mezanino (*Development Board*). O programa desenvolvido pode ser encontrado no apêndice G.

2.6.4 Sensores e comunicação

Todos os módulos escolhidos são fabricados pela *Adafruit Industries* na forma de mezaninos. Eles possuem a vantagem de trabalhar tanto com 5V, quanto com 3.3V. Os módulos utilizados na placa são o LSM9DS1, da fabricante *STMicroelectronics*, um *chip* com tecnologia MEMS(*MicroElectoMechanical System*), que inclui um magnetômetro, acelerômetro, giroscópio de 3 eixos cada e o BME280, produzido pela fabricante *Robert Bosch GmbH*, que inclui um sensor de temperatura, pressão e umidade. Ambos os sensores comunicam com o microcontrolador por meio de uma segunda linha de comunicação com protocolo I2C, com clock de 400kHz.

Portas de expansão na placa foram inseridas, possibilitando futuramente a inclusão de outros sensores ambientais, que já estão nos planos futuros, como medições de ozônio, metano e dióxido de carbono. As portas disponíveis são uma UART, I2C e SPI.

Os módulos RTC e GPS são usados para aquisição de tempo, essencial no projeto para o cálculo de fluxo. O CI do módulo RTC é o DS3231, da fabricante *Analog Devices*, que utiliza a mesma linha de comunicação I2C que os sensores, e o módulo GPS, modelo PA1616S, da fabricante *CDtop Technology*, que faz uso de uma linha de comunicação UART. A comunicação UART do DC-DC, responsável pela polarização dos SiPMs, também é realizada com o ESP32.

Outros módulos inseridos no sistema de aquisição foram dois microSD, responsáveis por salvar os dados adquiridos. A opção por 2 módulos na placa foi realizada por conta da futura instalação do experimento no módulo CRIOSFERA 1. Nesse local, não há um computador responsável pela aquisição de dados ou qualquer forma de *backup*. Sendo assim, uma cópia em um segundo cartão de memória é realizada, evitando perda de dados, caso um dos cartões seja corrompido. Esses dois módulos compartilham uma linha de comunicação com protocolo SPI com o módulo Ethernet controlado pelo *chip* W5500, da fabricante *WIZnet*. O módulo Ethernet, assim como o WI-FI disponível no ESP32, foi adicionado como forma de envio de dados remota sem necessidade de um computador, que poderá ser futuramente usado.

2.6.5 Formato de dados do DAQ do Experimento

O formato de dados salvos segue um padrão CSV(*comma separated values*), que é um arquivo de texto cujos valores adicionados são delimitados por vírgula. No formato utilizado, a vírgula foi substituída por tabulação que facilita a localização de dados específicos. A sequência dos dados é formada pelos identificadores de tempo, primeiramente do GPS e, em seguida, do RTC, ambos segundo o formato *unix timestamp*. Esse formato utiliza uma contagem de segundos desde o dia primeiro de janeiro de 1970.

No programa de aquisição, que é executado no computador, além desses identificadores, é adicionado, na frente da linha de dados recebida pelo USB do DAQ, um identificador obtido pelo tempo registrado na BIOS do computador, que é atualizado por um servidor NTP (*Network Time Protocol*). Todos esses *timestamps* são salvos utilizando-se o fuso horário GMT+0 (UTC), pois múltiplos experimentos podem ser enviados para locais com fusos horários diferentes e, mantendo-se o horário do meridiano de Greenwich, eventos podem ser facilmente comparados sem necessidade de conversão.

Na sequência, os valores dos contadores de cada canal são adicionados, seguidos pelas coincidências duplas, triplas e quádrupla de cada grupo. Após essa parte, as coincidências entre

grupos, seguidas dos dados dos sensores e, por último, o identificador do sistema de injeção de luz (ID) são adicionados. A figura 2.34 ilustra a sequência desse formato de dados escolhido.

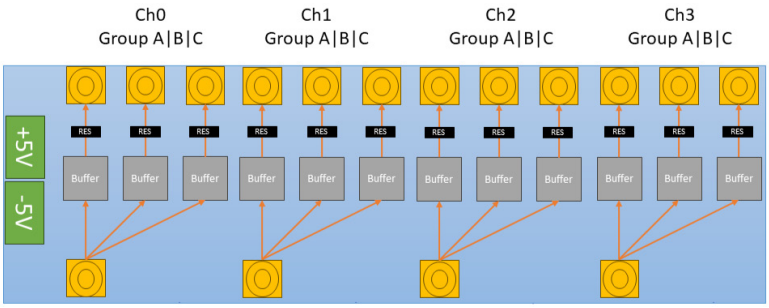
Time (GPS | RTC)	Counters Single Ch (3Groups = 12 Counters)	Counters Chs Coincidence (same group) (6 double coincidences 4 triple coin 1 quadruple coin)	Counters Chs Coincidence Between Groups (9 coincidences each 2 groups)	Slow Control (Sensors Data)	Light Injection System ID

Figura 2.34: Esquema do formato de cada linha de dados que o microcontrolador gera quando o tempo configurado para contagem é concluído.

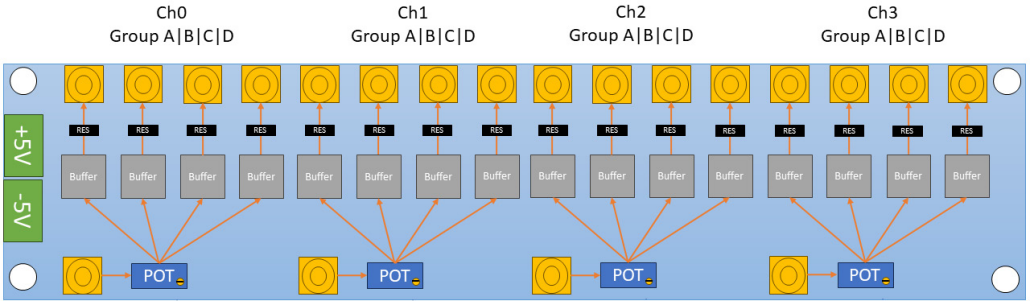
Um arquivo de dados é criado a cada hora, tanto para os dados salvos no cartão SD, quanto para os dados salvos no computador. Assim, quando o arquivo é aberto, um monitoramento do tempo é executado e, quando um período de 1h é terminado, o arquivo sendo escrito é finalizado e um novo arquivo é aberto. Essa escolha foi realizada para minimizar danos, caso algum arquivo seja corrompido, perdendo somente 1h de dados em vez de 24h, como, por exemplo, caso fosse utilizado um arquivo por dia. Ao mesmo tempo, mantém o padrão utilizado no experimento anterior, instalado no refúgio Ipanema em 2022. O programa de aquisição criado pode ser encontrado no apêndice H. Após o período de comissionamento do detector, foram inseridos 4 novos valores à linha de dados salva, que são obtidos do PID de controle da temperatura do refúgio.

2.6.6 Sistema de Injeção de Luz

O sistema de injeção de luz foi desenvolvido como uma forma de realizar a calibração do do sistema óptico/eletrônica de *front-end*, mensurando, de tempos em tempos a qualidade dos dados que estão sendo salvos. Por intermédio do protocolo, busca-se mensurar a eficiência do detector, canal a canal, de contar a mesma quantidade de pulsos que estão sendo disparados pelo LED UV, assim como verificar a existência de *crosstalk* entre canais, seja óptico, seja eletrônico. A placa foi projetada contendo somente 4 canais de injeção de LED e, inicialmente, seria necessário escolher 4 canais de um total de 12 existentes no experimento para receber a injeção. Entretanto, essa escolha não seria a ideal, visto que a avaliação de *crosstalk* só poderia ser feita corretamente nos canais adjacentes ao canal escolhido para ser injetado. Sendo assim, os 4 canais de injeção de LED do DAQ, ligados diretamente, por meio de chaves, ao I/O do FPGA, foram convertidos em entradas/saídas genéricas. Uma placa que repete o sinal da entrada em múltiplas saídas, por meio de *buffers* de corrente, foi desenvolvida. O *buffer* escolhido foi o modelo BUF602IDBVT, produzido pela *Texas Instruments*. Como cada grupo possui 4 planos de canais, possuir 4 canais de injeção de LED repetidos foi conveniente, pois é possível ligar cada plano de todos os grupos no mesmo canal do injetor. Logo, todos os planos superiores dos grupos (canal 0) foram ligados no canal 0 do injetor, seguido pelo plano logo abaixo, no canal 1 do injetor e assim por diante. Duas versões dessa placa foram projetadas e os diagramas de bloco podem ser vistos na figura 2.35.



(a) Versão 1.



(b) Versão 2.

Figura 2.35: Diagramas de bloco da placa do sistema de injeção de luz.

As principais diferenças da versão 1 para a versão 2 foram a inclusão de mais um grupo, deixando o experimento preparado para receber 4 grupos, e a inclusão de um potenciômetro na entrada dos *buffers*, que controla a tensão que será aplicada nos LED. Na primeira versão da placa, a mesma tensão de saída do FPGA, de 3.3V, é usada para alimentar os LEDs. Mesmo estando dentro da faixa de operação do modelo de LED utilizado, que o fabricante informa ser de 2.8V a 3.6V, a quantidade de luz gerada saturou o amplificador da eletrônica, distorcendo o sinal obtido, inclusive com o menor tempo de pulso que o LED dispara, como visto na figura 2.36. Esse tempo foi obtido com testes em diferentes larguras e o tempo mínimo do pulso para o LED disparar foi em torno de 8.5ns, existindo pequenas variações em cada LED.

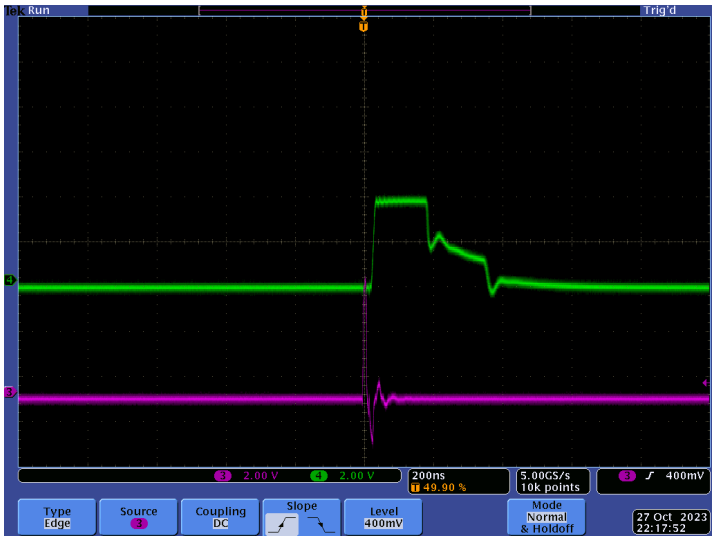


Figura 2.36: Sinal obtido na saída do segundo estágio da FEE (amplificador), com sistema de injeção de luz ligado em 10kHz e largura de pulso de 10ns, utilizando a versão 1 da placa repetidora.

Sendo o mesmo teste executado na versão 2 da placa, com a tensão ajustada para 2V com os potenciômetros, é possível ver na figura 2.37 que o sinal resultante possui um formato compatível com o pulso gerado por partícula passante no detector.

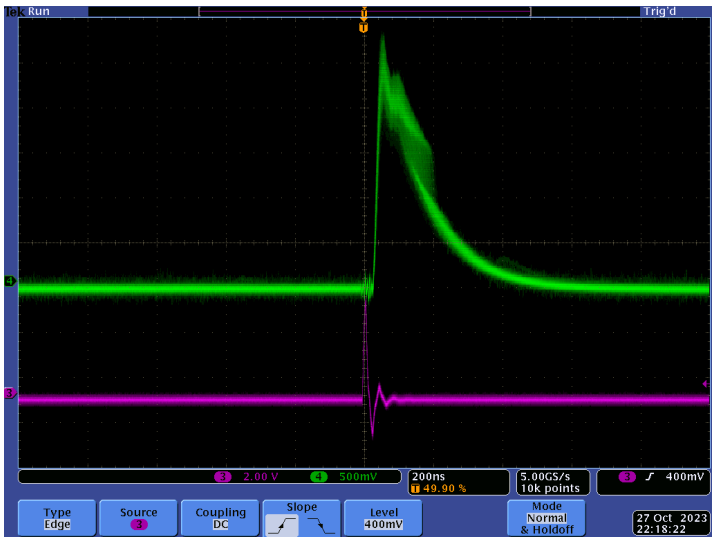
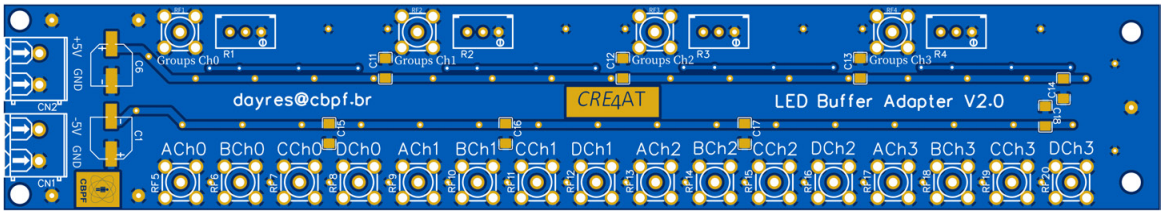
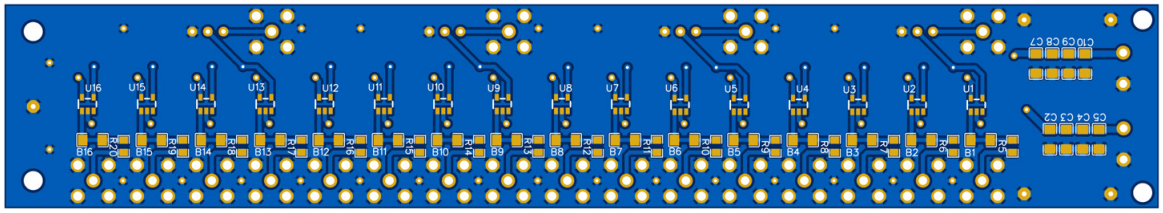


Figura 2.37: Sinal analógico obtido na saída do amplificador, com sistema de injeção de luz ligado em 10kHz e largura de pulso de 10ns, utilizando a versão 2 da placa repetidora com saída de 2V.

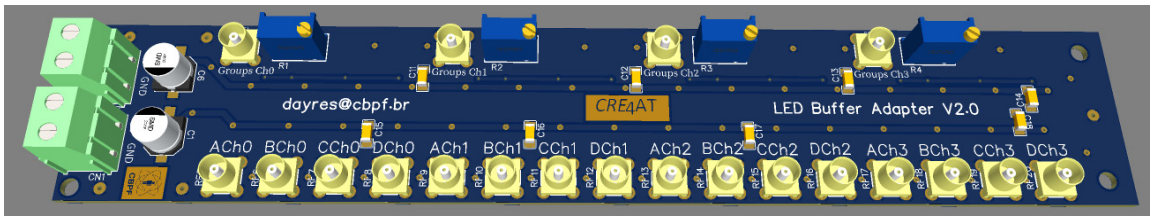
O projeto da versão 2 da placa pode ser encontrado no apêndice I e, na figura 2.38, é possível ver o projeto 2D e 3D da placa.



(a) Visão superior da placa.



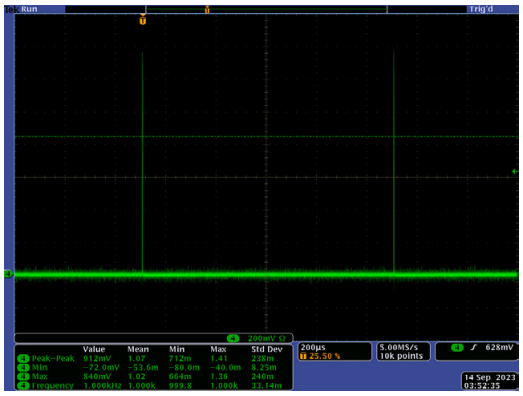
(b) Visão inferior da placa.



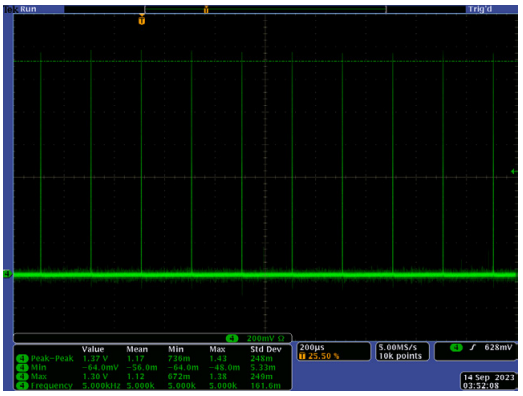
(c) Visão 3D da placa com componentes.

Figura 2.38: Visão da placa repetidora do sistema de injeção de luz.

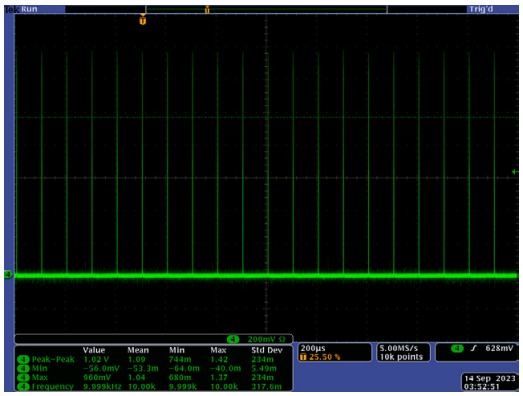
O protocolo de calibração foi configurado, na *firmware*, para ser executado uma vez por mês, todo dia primeiro, à 0h. O protocolo consiste em configurar o injetor para executar cada uma das possíveis frequências de disparo, iniciando em 1kHz, indo até 100kHz, ligando todas as possibilidades de canais separados e combinações. A saída do *buffer* para cada possibilidade de frequência de injeção pode ser vista na figura 2.39 e o fluxograma do processo na figura 2.40.



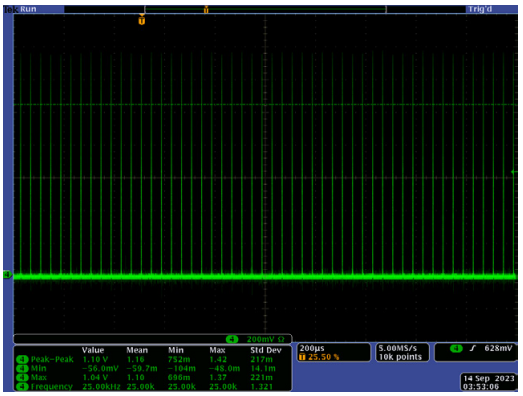
(a) Frequência de 1kHz.



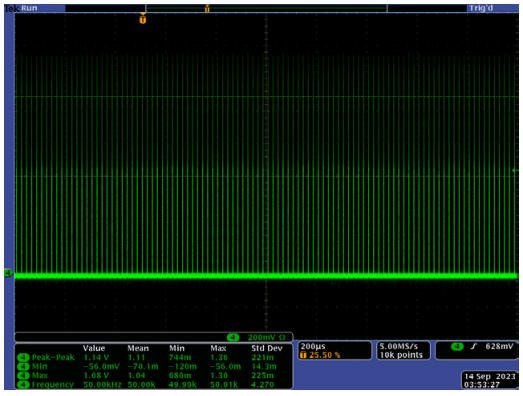
(b) Frequência de 5kHz.



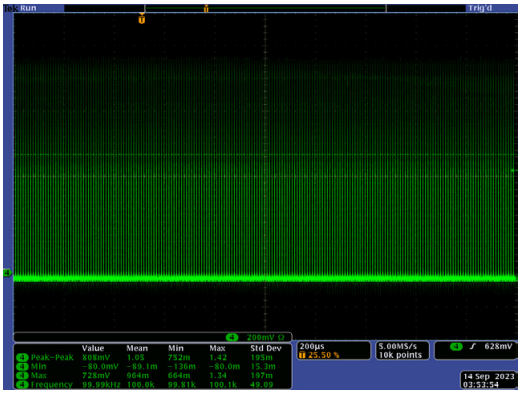
(c) Frequência de 10kHz.



(d) Frequência de 25kHz.



(e) Frequência de 50kHz.



(f) Frequência de 100kHz.

Figura 2.39: Saída do amplificador com as diferentes frequências de injeção de luz utilizadas durante a calibração.

Para que seja considerada correta, a contagem que foi realizada enquanto o microcontrolador altera valores do sistema de injeção no FPGA necessita ser eliminada. Para isso, foi inserido

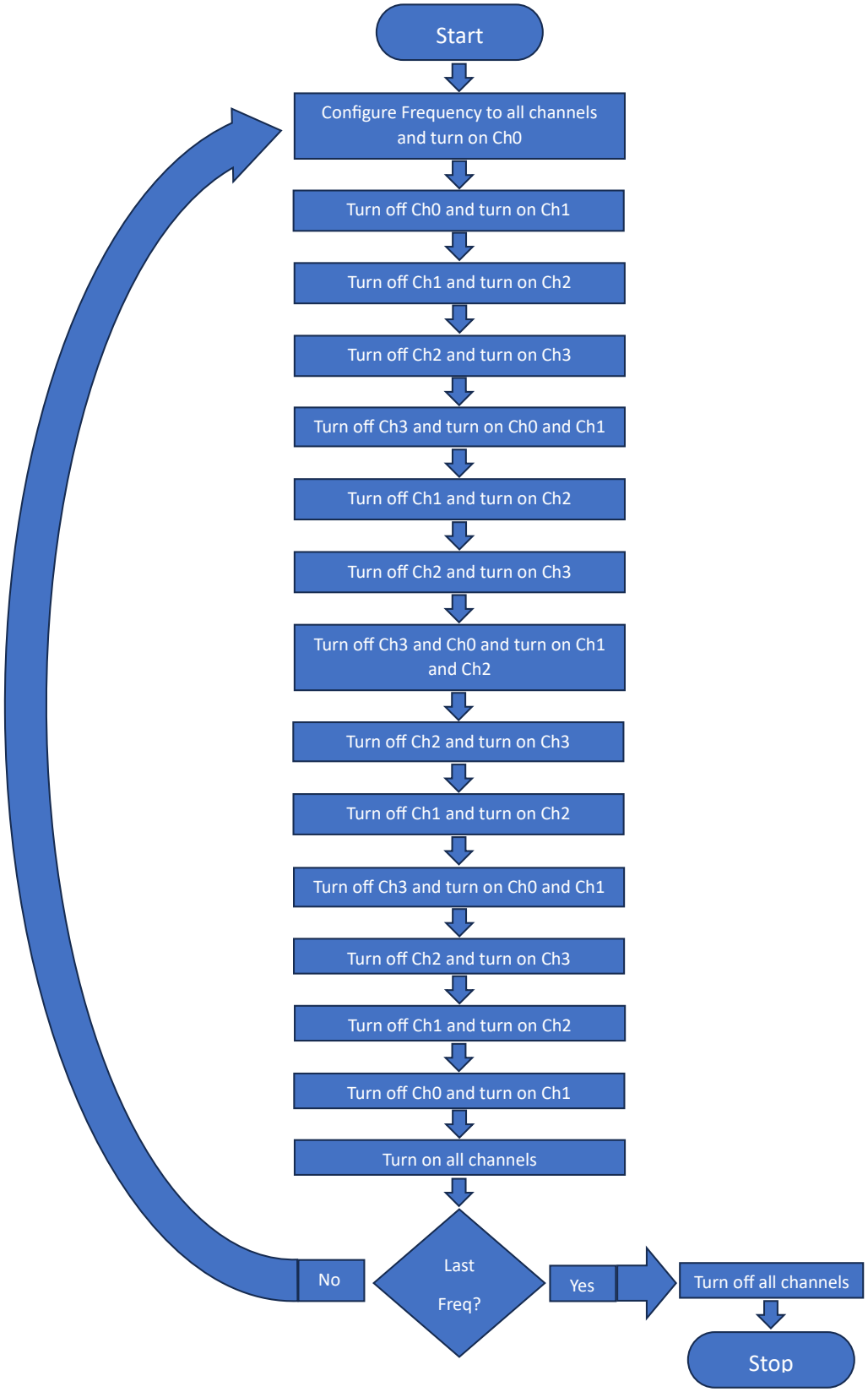


Figura 2.40: Fluxograma do processo de calibração do experimento CRE4AT pelo sistema de injeção de luz.

um protocolo de identificação do sistema de injeção de luz no fim de cada linha de dados. Os valores inseridos são os seguintes:

1. 0: Sistema de injeção de luz desativado;
2. 1: Sistema de injeção de luz sendo configurado (dados a serem descartados);
3. frequência+posição: Linhas de dados com essa sequência serão utilizadas na análise da calibração. O número é formado pela frequência com a qual o injetor está configurado mais a posição dentro do protocolo. A contagem de posição inicia em 2, para não haver confusão com os números 0 e 1 já utilizados. Por exemplo: se os canais 1 de cada grupo receberem uma injeção com frequência de 10kHz, que é a segunda posição do protocolo, o identificador será 10003.

Logo, a cada contagem válida do processo de calibração, perde-se a contagem seguinte, pois, durante a sua execução, os parâmetros do injetor são alterados.

2.6.7 Comandos de Controle

Os comandos de controle implementados são simples de serem utilizados e podem realizar o controle de todos os parâmetros disponíveis. Ao tentar executar um comando inexistente, um bloco de ajuda, conforme exibido abaixo, é mostrado ao usuário, indicando a maior parte do que pode ser executado.

```
Wrong Data Format!
Type: 'REG;[CONFIG]';     [CONFIG]: 'REGNum' to read or 'REGNum;Value' to write
        Reg List:
                [0] -> Bit 0-3(LED0-3 On[1]/Off[0]) - Bit 4 (Trigger Out On[1]/Off[0])
                [1-4] -> LED 0-3 Freq Selection
                        0 -> 1kHz
                        1 -> 5kHz
                        2 -> 10kHz
                        3 -> 25kHz
                        4 -> 50kHz
                        5 -> 100kHz
                [5] -> Trigger Freq Selection
                [6-9] -> LED 0-3 Pulse Width (Steps 20ns)
                [10] -> Trigger Pulse Width
                [11] -> LED Minimum Pulse Width
                [12] -> Latch Input Pulses Width Selection (Step 20ns)
                [13-14] -> Counter time in seconds (Reg 13 - Most significant Byte)
        RTC;year;month;day;hour;minute;second to set RTC time
        'GETDATA' to force FPGA count
        'SAVEDATA' to save FPGA CONFIG REG Bank in the memory
        'HV;[COMMAND]' -> Userful Commands:
                HPO -> Check all info from HV PS
                HGV -> Get Voltage read
                HGC -> Get Current read
                HGT -> Get Temperature read
                HON -> Turn On output
                HOF -> Turn OFF output
```

HST;Value -> Set HV Output Value
'StartLEDCalib' to force LED Calibration Procedure

O protocolo dos comandos disponíveis utiliza ponto e vírgula como separador, e, os principais comandos, como mostrado na caixa acima, são:

1. "REG;" que é utilizado para escrever e ler o banco de memória de controle dos parâmetros do FPGA. O envio de um comando REG, seguido de um único valor entre 0 e 255, será interpretado como leitura e esse valor será utilizado como o endereço do banco. Ao enviar dois valores separados por ponto e vírgula, o primeiro será o endereço do banco de memória e o segundo, o valor a ser escrito;

2. "RTC;" seguido de uma sequência de valores, como mostrado acima, que são a data e hora que configurarão o RTC;

3. "GETDATA" que iniciará o protocolo de leitura do banco de memória do FPGA, que contém os valores dos contadores. A saída desse comando será um linha de dados completa, com os valores de tempo, dos contadores, dos sensores e o ID do sistema de injeção de luz;

4. "SAVEDATA" serve para salvar os valores configurados na memória de controle do FPGA na memória flash do ESP32. Como o FPGA é apagado ao ser desligado, toda vez que é religado e a *firmware* é carregada, os valores utilizados na configuração devem ser configurados novamente e o microcontrolador é responsável por esse processo;

5. "HV" que será enviado ao conversor DC-DC, responsável pela polarização dos SiPMs. O protocolo utilizado por ele foi implementado na *firmware* do microcontrolador, que inclui um *byte* de *cyclic redundancy check* (CRC) para verificação de erros no fim da comunicação. A maior parte dos comandos, como "HPO"ou "HON", é enviada sem modificação ao conversor, porém, o comando "HST", que é utilizado para configurar a tensão de operação foi simplificado, facilitando a configuração do conversor pelo operador. Esse comando envia ao DC-DC os parâmetros da reta de calibração que utilizará para fazer a correção da tensão de saída em relação à temperatura. Como os parâmetros da reta são os mesmos, só alterando o valor de tensão que deverá alimentar os SiPMs quando a temperatura for de 25°C, os outros valores a serem enviados foram fixados em código;

6. "StartLEDCalib" forçará o início do protocolo de calibração pelo sistema de injeção de luz do experimento, independentemente do dia e hora. É utilizado para verificação e validação da montagem do experimento.

2.7 Caixa de Proteção de Entrada de Luz

Devido à necessidade de transporte do experimento para diversas regiões e, ao mesmo tempo, assegurar uma proteção extra à entrada de luz, uma caixa feita em polietileno preto de alta densidade foi desenvolvida, baseada na caixa utilizada no projeto piloto (CREAT1 ou CRE@AT), e produzida sob encomenda. A caixa possui vedação contra entrada de luz, possuindo curvas de 90°, assim como a óptica do experimento, tanto na vedação da tampa quanto na região frontal, que possui um painel onde os cabos de alimentação e dados são ligados, mostrado na figura 2.41. A caixa também ajuda no transporte do detector, uma vez que toda a estrutura e placas de *front-end*, DAQ e do sistema de injeção de luz são montadas dentro dela.

O projeto da caixa, validado anteriormente, foi alterado devido à nova dimensão do experimento, que deveria comportar toda a parte mecânica que é instalada. A caixa original possuía 19cm de altura, alterada para 40cm. O projeto completo pode ser encontrado no apêndice J.

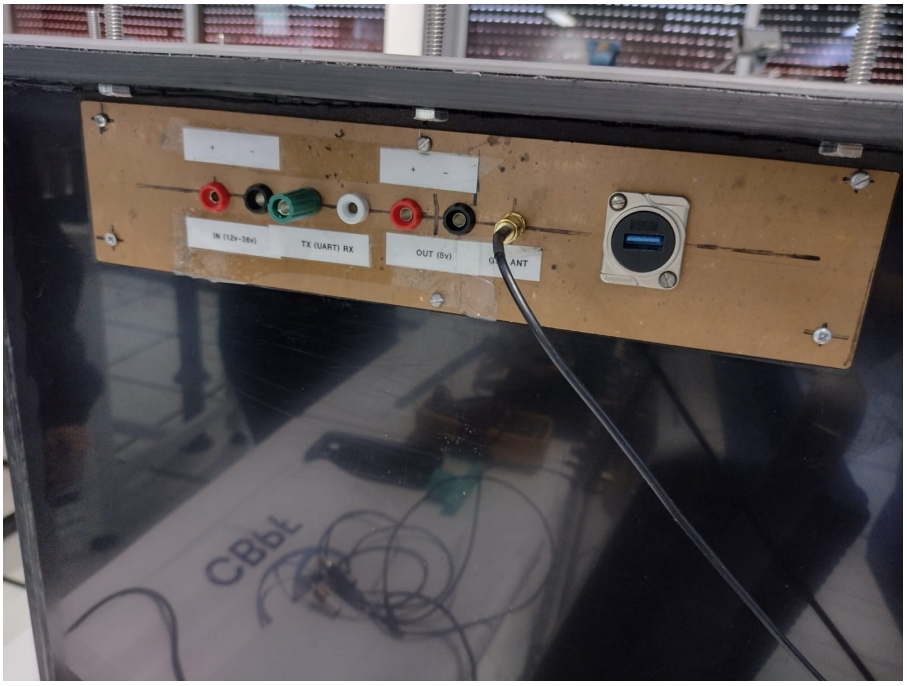


Figura 2.41: Painel da caixa feito para conexões de entrada e saída de sinais e alimentação.

O painel de conexões possui dois conectores bananas de alimentação, dois conectores de saída de 5V que, em conjunto com os conectores Tx e Rx (UART), fazem parte do projeto da placa de expansão de sensores ambientais externos, que serão instalados futuramente. Também possui um conector SMA para antena do sistema GPS e uma porta USB para comunicação com o computador. A placa de expansão possuirá sensores para medidas ambientais, como CO^2, ozônio, metano, dentre outros. A opção por essa solução externa deveu-se à necessidade de manter os sensores em contato constante com o ambiente, medindo a variação dessas grandezas. Este projeto está sob a responsabilidade de um outro estudante do grupo.

2.8 Caracterização da Eletrônica

Após a escolha dos valores para a largura de pulso, definida pelo integrador, e da amplificação ajustados pelos potenciômetros, um ponto de operação também definido foi a tensão de limiar (*threshold*) do comparador responsável pelo discriminação do sinal analógico. Para realizar essa escolha, foi realizado um escaneamento de *threshold*. Uma bancada de teste foi preparada para esse teste, utilizando um gerador de onda *Tektronix* AFG3252 para injetar luz por meio do LED ultravioleta no sistema, controlando tanto a frequência quanto a largura de pulso.

Três curvas de calibração foram montadas nesse teste:

1. Curva gerada pelo escaneamento sem injeção de luz;
2. Curva gerada pela injeção de luz em um canal;
3. Curva de injeção de luz no grupo (4 Canais).

O método usado consistiu em realizar a injeção de luz com a menor largura de pulso e amplitude possível, para promover a injeção da menor quantidade de fótons, gerando um pulso

no SiPM próximo ao gerado por partículas carregadas que atravessam o cintilador. O LED usado informa que a tensão de operação dele está entre 2.8V e 3.6V, porém, com o gerador de função, foi possível gerar luz a partir de 1.6V com 0.02% de *duty cycle*, que é o tempo em nível lógico alto em relação ao período total do pulso, em 10kHz.

Para validar essa escolha do pulso de injeção, utilizando esse sistema montado para o escaneamento de *threshold*, foi também desenvolvido um código para a leitura da carga injetada pelo LED UV na saída do amplificador da FEE em um QDC V965 da *CAEN*, por meio da bridge V1718 da *CAEN*. Os códigos podem ser encontrados no apêndice K. A figura 2.42 mostra o esquema de ligação dessa bancada de teste. Os seguintes módulos VME da *CAEN* foram necessários para realização desse teste:

1. Bridge V1718;
2. QDC V965;
3. Conversor TTL-NIM V976;
4. Módulo Linear Fanout N625.

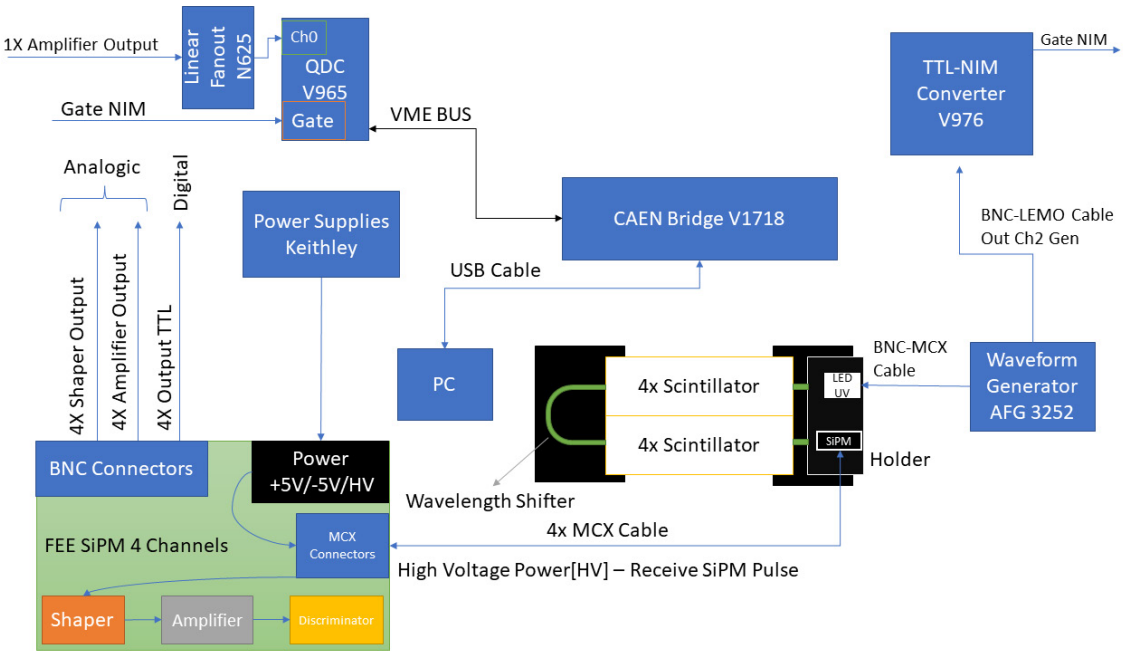


Figura 2.42: Diagrama de blocos da bancada de teste para leitura da carga injetada pelo LED UV.

Pelo fato de o QDC possuir entrada negativa de pulso e o sinal da saída amplificada ser um pulso positivo, primeiramente o sinal foi invertido por meio do *linear fanout* da *CAEN* com ganho em -1, para depois ser injetado no módulo QDC. O gerador de função ficou responsável pelo pulso gerado para o LED na primeira saída e um pulso de largura maior que a saída analógica do amplificador da *front-end* na segunda saída, que funciona como o *gate* do QDC. O pulso de *gate* é responsável por indicar o tempo em que a entrada analógica do QDC integrará o pulso de entrada para, posteriormente, ser convertido em um sinal digital correspondente à carga injetada durante esse tempo. O resultado do teste pode ser visto na figura 2.43.

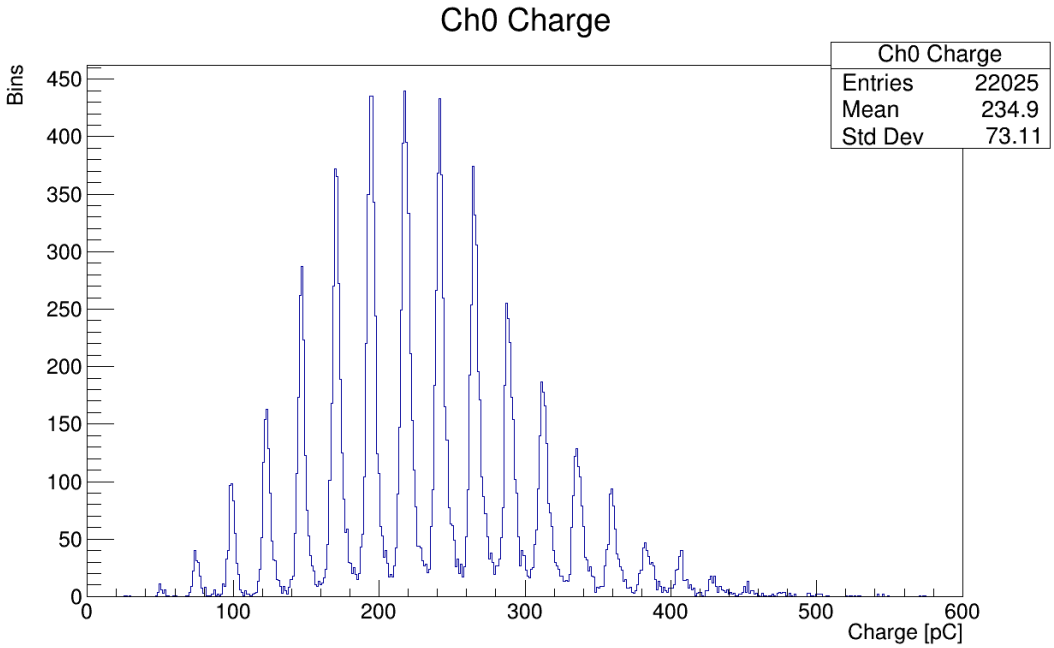


Figura 2.43: Gráfico gerado com os dados do QDC com injeção de luz em um canal do grupo.

Esse histograma mostra os picos de carga, quando o número de *pixels* do SiPM, que entram em efeito de avalanche aumenta. A diferença de carga entre dois picos consecutivos, que, como é possível observar, é constante, é o valor em carga de um pixel que foi disparado, logo, é o valor em carga de um fóton, após os ganhos dados pelo próprio SiPM e os estágios de *shaper* e amplificação. Um resultado importante retirado desse teste é a validação do sistema de injeção com o LED UV, assim como a verificação de que é possível injetar uma quantidade baixa o suficiente de luz sem que haja saturação do fotomultiplicador de silício, pois não existe um corte nos dados e o histograma foi decaindo gradativamente após a região de pico central, onde a probabilidade de *pixels* saturados é máxima com o pulso injetado no LED.

Para encontrar o ponto ideal de operação da tensão de limiar, foi realizado, a princípio, um escaneamento de *threshold* de forma manual em um canal, alterando o potenciômetro para o valor de tensão desejado. O tempo de aquisição de cada ponto escolhido foi de 100s e, por conta do tempo necessário para realizar esse processo, não foi possível ajustar muitos pontos, realizando o processo de forma não linear, diminuindo o passo próximo da região em que era esperado o aumento exponencial da contagem, em virtude da corrente de escuro e ruído. Este resultado para um canal, mostrado na figura 2.44, evidencia que, acima de 0.13V, o ruído e/ou corrente de escuro do transdutor são completamente eliminados com a alteração da inclinação de queda da curva.

Para escolha do valor ideal, dois fatores foram avaliados:

1. Caso a escolha seja um valor muito alto, longe do ruído eletrônico e da corrente de escuro, partículas carregadas que depositem baixa energia ao passarem pelo detector podem não ser identificadas, já que o pulso analógico poderá ficar abaixo do valor de limiar;

2. Caso o valor escolhido seja baixo demais, dentro da região de ruído, a probabilidade de múltiplos canais terem pulsos gerados por ruído/corrente de escuro nas saídas dos discriminadores coincidentes aumenta, e serão identificados como partículas passantes erroneamente.

Para otimizar a identificação de partículas passantes, levando em consideração os dois fatores acima, tendo em vista que o experimento faz uso de coincidência quádrupla, durante o

comissionamento, o valor do início da região de aumento exponencial de 130mV foi utilizado, para aumentar a eficiência do detector.

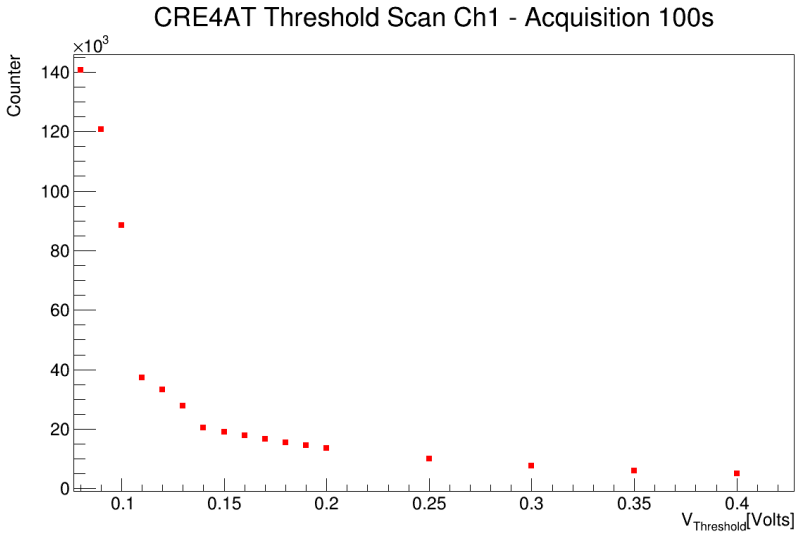


Figura 2.44: Resultado do escaneamento manual da tensão de limiar.

Após o envio do experimento, um escaneamento automatizado foi realizado para a obtenção de um resultado mais consistente, utilizando passos constantes durante todo o processo. Como a eletrônica desenvolvida não possui um sistema de controle automatizado da tensão de limiar, os potenciômetros que realizam o ajuste não foram montados em uma eletrônica e, no lugar deles, foi inserida uma tensão através de uma fonte de alimentação controlada *Keithley* 2200-72-1. Logo, todos os canais receberam o mesmo valor da tensão de limiar, sendo escaneados simultaneamente. Um código *python* foi programado para realizar o controle da fonte e do DAQ, realizando todo o processo de forma autônoma, que pode ser encontrado no apêndice L.

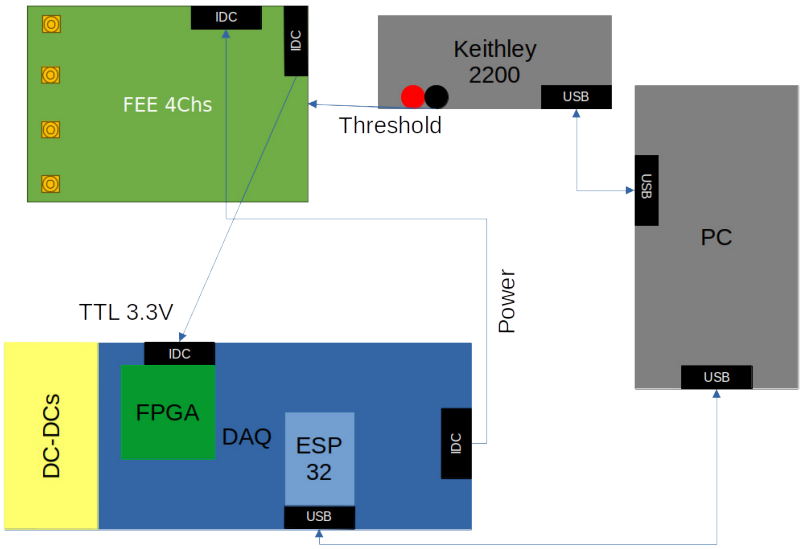


Figura 2.45: Diagrama de blocos simplificado da bancada montada para o escaneamento de *threshold* automatizado.

Durante todo o processo, foi escolhido utilizar um tempo de aquisição de 120s e, além da varredura da tensão de limiar, com passo de 10mV, uma varredura do valor de *latch* também foi realizada, com 100ns entre cada ajuste. Primeiramente, para descobrir o ponto de ruído da

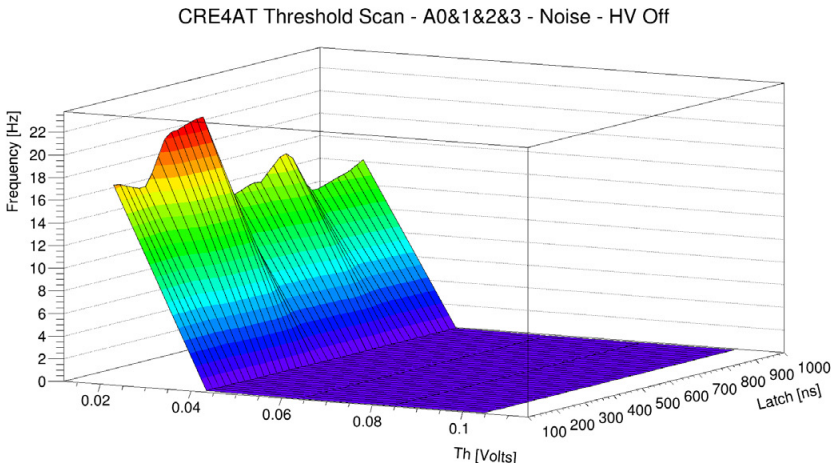
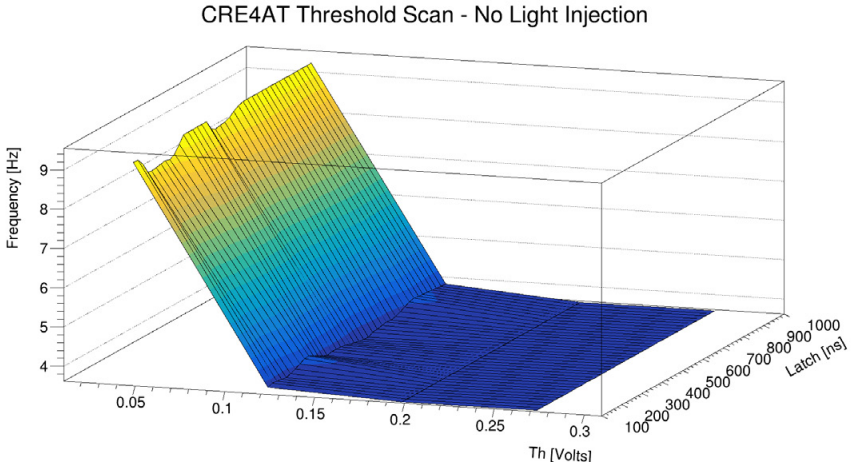


Figura 2.46: Varredura da tensão de limiar e *latch* sem alta tensão.

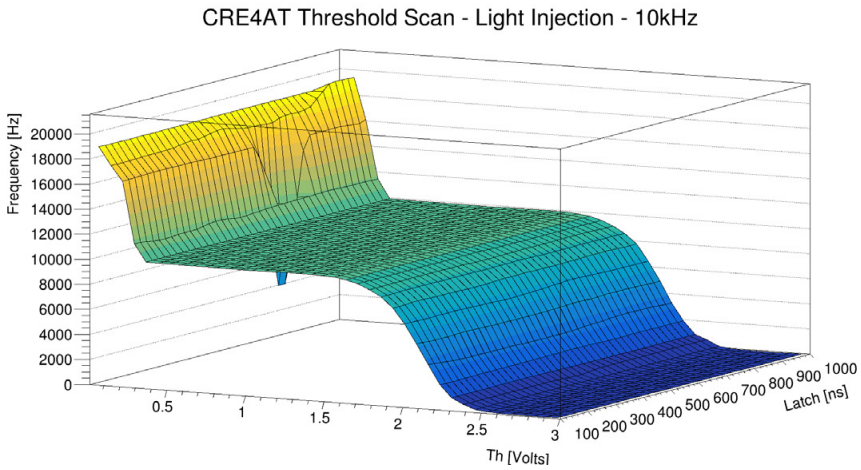
eletrônica, um escaneamento com alta tensão desligada foi realizado. Como é possível verificar na figura 2.46, quando o *threshold* foi configurado em 40mV, a frequência zera, achando assim o ponto de ruído eletrônico. Uma observação importante foi o fato de que, independentemente de o valor de *latch* escolhido, a contagem obtida permanece dentro da mesma faixa com uma flutuação esperada. Isso significa que a saída do comparador utilizado no discriminador não possui oscilações durante as transições, que poderiam resultar em múltiplas contagens para uma única transição.

Para melhor identificação do ponto de operação, uma outra varredura foi executada com alta tensão ligada, com e sem injeção de luz. Como essa aquisição foi realizada utilizando coincidência, é observado no resultado mostrado pelo gráfico da figura 2.47 (a) que, apesar da coincidência, o ruído e/ou corrente de escuro presente, abaixo dos 120mV da tensão de limiar, não são eliminados do resultado. Um sinal eletromagnético está presente no laboratório e contamina todos os canais da eletrônica, sendo contado na coincidência. Devido a esse resultado, foi definido alterar a tensão de limiar que será utilizada para 130mV, fora da região onde ocorre o aumento da contagem de forma muito superior.

Já a figura 2.47 (b) mostra o resultado com injeção de luz por meio do LED UV. É visível no gráfico a consistência do sistema de disparo e da carga depositada pela luz, validando o sistema que será utilizado para calibração durante o período de aquisição do experimento CRE4AT. Durante a aquisição, um ponto teve problema, ficando abaixo do valor esperado na região onde o ruído e/ou corrente de escuro estão presentes.



(a) Sem injeção de luz, mostrando região onde ocorre a mudança na taxa de variação.



(b) Com injeção de luz em 10kHz.

Figura 2.47: Varredura da tensão de limiar e *latch* com alta tensão.

Com relação ao DAQ, alguns parâmetros tiveram que ser configurados. Um desse parâmetros foi o *latch*, que é responsável por definir um tempo fixo para o pulso recebido da FEE, logo, um tempo morto onde outros pulsos não serão contados, e, ao mesmo, caso aconteça uma oscilação na saída do comparador, isso não seja identificado como passagens de múltiplas partículas. Como a escolha da largura de pulso selecionada ficou entre 400ns e 500ns, o *latch* selecionado foi de 600ns, tendo uma margem acima da largura configurada no pulso analógico.

Já o tempo de aquisição foi configurado para 10s, que é suficiente para os sensores disponíveis conseguirem realizar uma nova aquisição e, ao mesmo tempo, mantém o padrão do período de aquisição de dados já utilizado no experimento, atualmente instalado no refúgio Ipanema. Esse tempo de aquisição, apesar de não possuir estatística suficiente para cálculo de eficiência e fluxo, é interessante, pois eventos rápidos podem ser identificados com mais clareza e, quando necessário realizar medidas mais precisas, é possível somar os valores dos contadores salvos em diversas aquisições de 10s, reorganizando os dados para aumentar a estatística disponível.

Capítulo 3

Tecnologia de busca do fator do Ângulo Zenital

No modelo do fluxo de raios cósmicos, um fator presente carrega informações relativas às características atmosféricas ocorrentes na região, onde as medidas estão sendo efetuadas. Em função dessas peculiaridades, o seu valor se tornou um parâmetro de interesse a ser medido, no contexto do experimento CRE4AT. Esse fator será chamado de fator do ângulo zenital ou ZAF.

O modelo de produção de raios cósmicos galáticos utilizado segue a fórmula [60][16][61]:

$$Fluxo = \int_{\theta=0}^{\pi/2} \int_{\phi=0}^{2\pi} I(\theta) \cos^n(\theta) \sin(\theta)\, d\theta\, d\phi \tag{3.1}$$

O fator que carrega essas características pode ser visto na equação como "n", elevando o cosseno do ângulo zenital (θ). Como a equação é utilizada para o cálculo de fluxo ominidirecional, a integração ocorre em todas as direções, definidas pelas coordenadas esféricas, θ de 0 à $\pi/2$ e ϕ de 0 à 2π.

Tradicionalmente, para medir esse fator n, utiliza-se um detector segmentado em X e Y com múltiplos planos e, assim, torna-se possível recriar o trajeto da partícula que atravessa o detector e medir, diretamente, a distribuição de ângulos θ e ϕ associados a esses trajetos. Em posse dessa distribuição, pode-se realizar um ajuste do modelo $\cos^n(\theta) \times \sin(\theta)$ na distribuição de θ, encontrando o fator n que melhor realize o ajuste.

Existem algumas limitações nesse processo que podem afetar a precisão da medida, dentre elas a aceptância do detector, pois ele possui uma limitada faixa de ângulos θ que definem uma trajetória, cujas partículas passarão pelos planos. Outra limitação está na precisão de medidas espaciais do detector, dentre elas, a distância entre planos, a distância entre segmentações, seja em X, seja em Y. Todos esses fatores são relevantes na precisão do valores mensurados, pois um erro na medida da altura entre os planos gerará um erro de medida no ângulo encontrado, e, por conseguinte, a distribuição de θ será modificada.

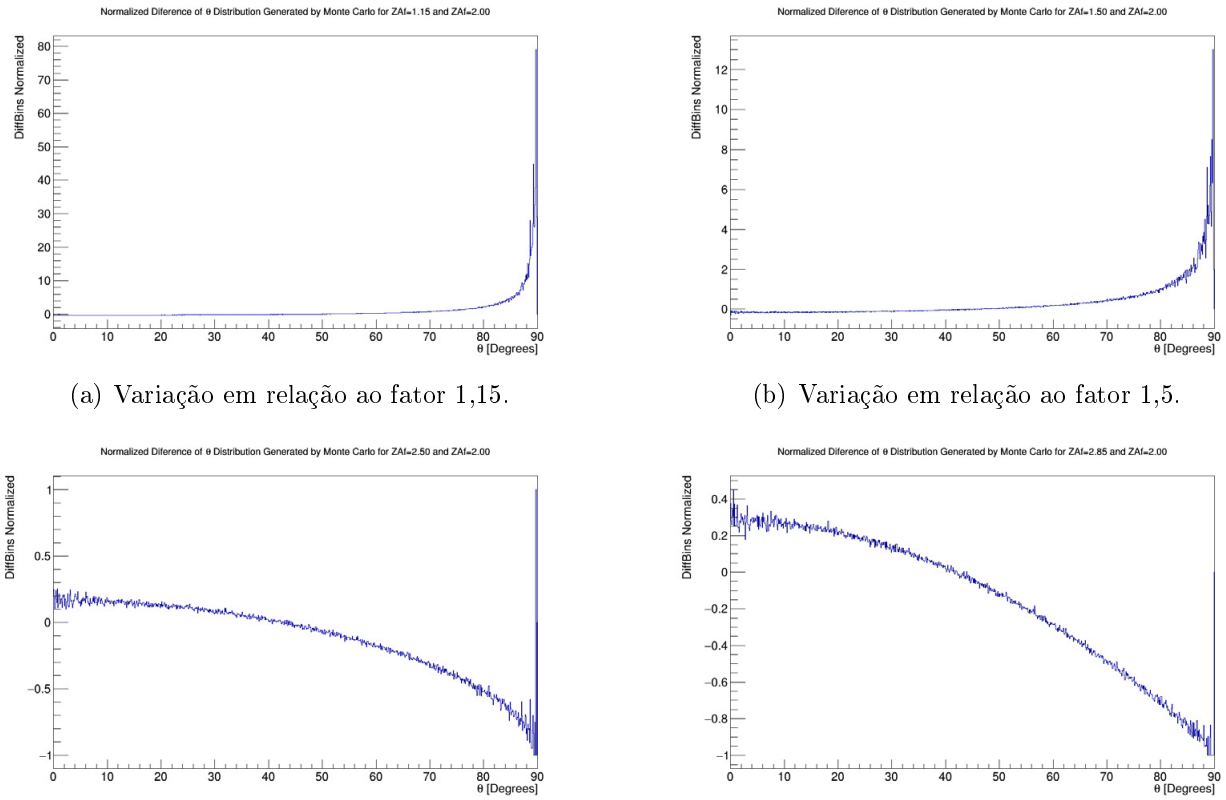
Outro aspecto a ser considerado é a quantidade de canais necessários para a produção de um detector segmentado. Para se obter uma boa resolução espacial, uma quantidade de canais relativamente alta por eixo é necessária, aumentando os riscos relativos à precisão espacial da montagem e medida relativa entre segmentos, além de elevar o custo do projeto. Para realizar uma comparação do CRE4AT com um detector segmentado, será utilizado um projeto disponível no laboratório multiusuário da coordenação de física de altas energias do CBPF, o Streamer. Esse detector com tecnologia de câmaras de gás possui 128 canais por plano, sendo 64 no eixo X e 64 em Y, e 3 planos para reconstruir a trajetografia.

Um estudo desse modelo de geração dos RCG, utilizando-se o método Monte Carlo, foi realizado por meio de uma análise comparativa da variação da distribuição de diferentes ângulos

θ em relação à referência, que é a distribuição gerada pelo valor médio do fator do ângulo zenital mais aceito pelo comunidade, que é em torno de 2 [62]. O valor final foi normalizado em relação à distribuição da referência. Logo, o seguinte cálculo foi realizado:

$$\text{Comparação}= \frac{Histograma_{\theta}(ZAF{=}X) - Histograma_{\theta}(ZAF{=}2.00)}{Histograma_{\theta}(ZAF{=}2.00)}$$

Os resultados obtidos estão na figura 3.1 e o código responsável por essa análise enconta-se no apêndice N.



(a) Variação em relação ao fator 1,15.

(b) Variação em relação ao fator 1,5.

(c) Variação em relação ao fator 2,5.

(d) Variação em relação ao fator 2,85.

Figura 3.1: Comparação normalizada da distribuição dos ângulos θ gerados pelo modelo em diferentes valores do fator que eleva o cosseno do ângulo zenital em relação ao valor médio de referência.

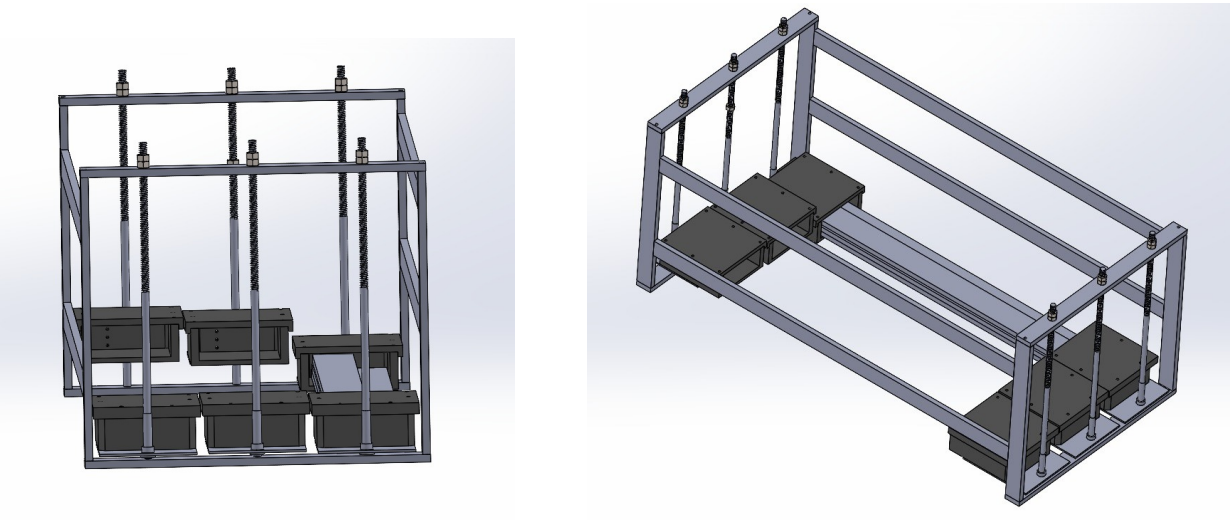
É evidente que a maior variação entre diferentes valores de ZAF ocorre em ângulos maiores da distribuição, próximos aos θ=90°.

Para realizar essa medida, evitando a necessidade de ter muitos canais, o CRE4AT faz uso dos grupos que já possui, sendo, no mínimo, 3 grupos, que, em combinação 2 a 2, geram 3 setores com uma faixa de ângulos θ idealmente diferentes. Essa região é selecionada por meio da diferença de altura entre 2 grupos, criando uma região de ângulos entre eles que define a aceptância entre esses grupos. A aceptância total do experimento é formada pela combinação das aceptâncias que existem a cada 2 grupos.

3.1 Estrutura mecânica do Experimento

Foi decidido criar uma estrutura, projetada em conjunto com a mecânica do CBPF, que, ao mesmo tempo, facilitasse o ajuste da altura desejada e também permitisse flexibilidade para,

caso necessário, realizar alterações nas alturas. A estrutura foi projetada em alumínio para ter resistência, sem aumentar significativamente o peso do projeto. Para controle das alturas, um parafuso é rotacionado, abaixando e elevando a altura fontal e traseira de cada grupo.



(a) Vista Frontal.

(b) Vista Lateral.

Figura 3.2: Estrutura Mecânica projetada.

Para facilitar o ajuste de altura, a mecânica do CBPF também fez o projeto de uma ferramenta que se encaixa no parafuso e a imprimiu em impressora 3D, mostrada na figura 3.3.

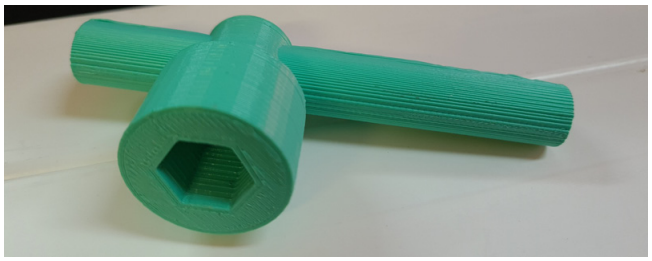


Figura 3.3: Ferramente produzida em impressora 3D para auxiliar no ajuste de altura.

Essa estrutura mecânica é instalada no interior da caixa de proteção de entrada de luz, juntamente com a eletrônica de aquisição e de injeção de luz. A figura 3.4, mostra a caixa do experimento enviado ao refúgio Ipanema (EACF, Antártica) montada.

Figura 3.4: Caixa de proteção com estrutura mecânica e placas instaladas.

Na figura seguinte, o desnível entre planos, gerado pela escolha dos ângulos diferentes para a obtenção de uma grande cobertura na aceptância total, é perceptível.

Figura 3.5: Foto do experimento montado na caixa, enfatizando o desnível entre grupos.

O espaçamento entre grupos e a altura escolhida podem ser vistos na figura 3.6. O espaçamento é definido pela distância entre os furos, onde os parafusos são inseridos, e, para definição da distância mínima, consideram-se as dimensões do conjunto óptico e da distância necessária para fazer a base, onde o conjunto é fixado. Esse valor é fixo e não pode ser alterado sem a modificação da estrutura. Já as alturas foram escolhidas e medidas a partir do cintilador superior até a base da caixa. Esses são os valores que são utilizados nas análises realizadas.

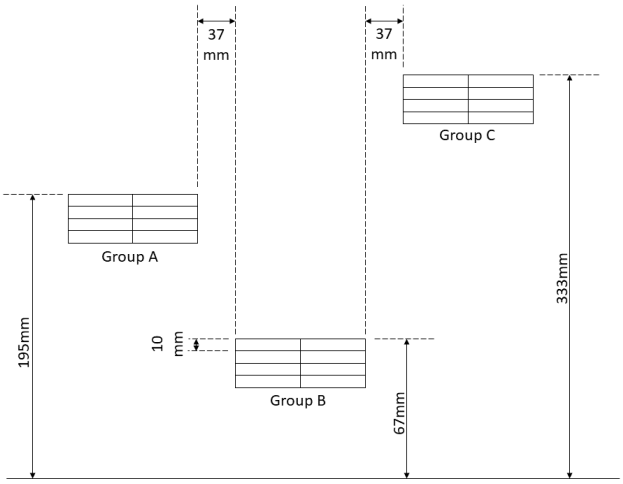


Figura 3.6: Posicionamento dos grupos no experimento CRE4AT enviado para o refúgio Ipanema.

3.2 Metodologia Utilizada

A vantagem dessa técnica está na flexibilidade com relação à aceptância do experimento, que pode ser facilmente ajustada para regiões de maior interesse e, ao mesmo tempo, ampliada com a inclusão de mais grupos. Para garantir que cada grupo esteja ajustado na horizontal, logo, perpendicular à normal, um acelerômetro é utilizado, garantindo que não haja inclinação em relação ao eixo Z. No experimento, cada grupo possui somente 3 parâmetros que podem sofrer com a imprecisão da medida: a inclinação do grupo, a altura e a distância em relação aos outros grupos, que, no caso de 3 grupos, são duas medidas fixas impostas pela estrutura mecânica.

Como os grupos possuem 4 cintiladores, a técnica utiliza dois cintiladores adjacentes de um grupo em coincidência com outros dois cintiladores adjacentes do outro grupo, para formar um traço. Assim, existem 9 combinações possíveis a cada dois grupos que fazem parte da aceptância do grupo. Por não possuir um seccionamento em dois eixos, a reconstrução do traço se torna inviável e, assim, só é possível ter uma faixa de ângulos ϕ que são detectáveis e uma faixa pequena em θ, onde está a maior precisão.

Visto que é inviável solucionar o problema por ajuste do modelo à distribuição do ângulo zenital, um método novo foi criado para solucionar essa procura do parâmetro. O método consiste na criação de uma tabela, por meio da simulação Monte Carlo com a configuração do detector e o modelo de produção teórico. O método de simulação Monte Carlo consiste na geração de amostras aleatórias repetidamente, para a obtenção de resultados numéricos. A tabela contém as contagens esperadas em uma certa quantidade de eventos, simulados para todas as coincidências normalizadas entre grupos. Assim, no caso de 3 grupos, para cada valor do fator a ser simulado, 27 razões serão salvas, sendo essas razões calculadas da seguinte forma:

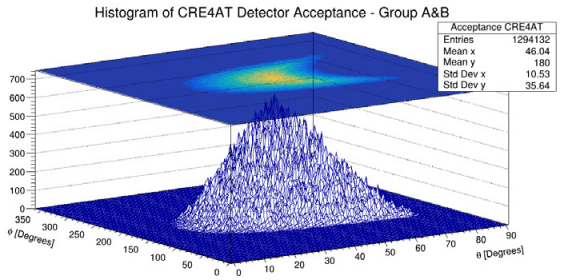
$$R_{GrupoW(ChY,ChY+1)GrupoX(ChZ,ChZ+1)} = \frac{Coin4_{GrupoW(ChY,ChY+1)GrupoX(ChZ,ChZ+1)}}{Coin4_{GrupoW} \times Coin4_{GrupoX}}$$

sendo:

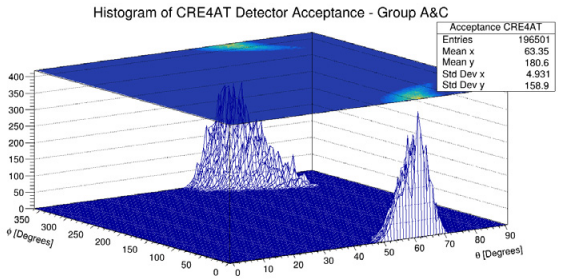
$R_{GroupW(ChY,ChY+1)GroupX(ChZ,ChZ+1)}$ a razão relacionada a 4 planos, sendo 2 planos adjacentes de um grupo X e dois planos adjacentes de um grupo W;

$Coin4_{GrupoW(ChY,ChY+1)GrupoX(ChZ,ChZ+1)}$ refere-se à contagem da coincidência quádrupla desses 4 planos, aos quais a razão está relacionada; e

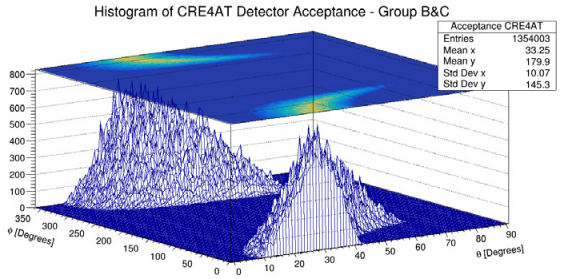
$Coin4_{GrupoW}$ e $Coin4_{GrupoX}$ referem-se às contagens em coincidência quádrupla de todos os planos dos grupos W e X escolhidos.



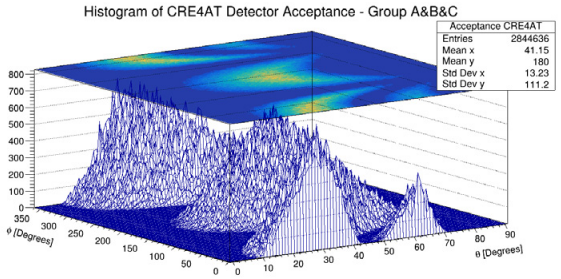
(a) Aceptância da combinação dos grupos AB.



(b) Aceptância da combinação dos grupos AC.



(c) Aceptância da combinação dos grupos BC.



(d) Aceptância da combinação dos três grupos.

Figura 3.7: Aceptância do experimento CRE4AT para procura do fator ZAF.

Essa tabela foi gerada do ZAF=0,01 até 5, com passos de 0,01, contendo 2 bilhões de eventos para cada ponto. O programa responsável por gerar a tabela pode ser encontrado no apêndice M. No DAQ do experimento, as mesmas contagens quádruplas são salvas e as razões também podem ser obtidas.

Na simulação, para geração do trajeto das partículas, um plano de produção é configurado, muito maior do que as dimensões do detector, para a obtenção de uma produção isotrópica, tanto em ϕ, quanto em θ. A isotropia é um fator relevante no método, visto que o detector é seccionado somente em 1 eixo e a técnica pressupõe que, independentemente da direção em que a medida é adquirida, a distribuição resultante será a mesma e não influenciará no resultado. Alguns fatores podem tornar esse pressuposto incorreto, alterando a distribuição adquirida em função da direção em que se encontra o eixo seccionado. Dentre eles, o fator geográfico local. No experimento que foi enviado à Antártica, para ser instalado no refúgio Ipanema, o fator geográfico é de suma importância, pois uma montanha encontra-se ao lado do refúgio, e uma parte dos RCG será absorvido por ela.

Por meio da simulação Monte Carlo, que gerou a tabela de valores para a procura do ZAF, também foram gerados histogramas contendo a aceptância de cada combinação entre grupos. Os histogramas contendo a aceptância entre grupo e total podem ser vistos na figura 3.7. Como pode ser observado nos gráficos, a região de ângulos zenitais aceitos pelos 3 grupos varia de, aproximadamente, 10° até 80°, cobrindo quase toda faixa possível, sem ocorrer sobreposição da região de aceitação entre grupos.

3.2.1 Validação do Método de procura

Para validar o método utilizado, uma simulação utilizando o equivalente a 24h de dados foi realizada para alguns diferentes fatores. O tempo escolhido coincide com o período de aquisição de dados, que será analisado no experimento CRE4AT. Como o esperado para cada metro quadrado de área de detecção é em torno de 100 a 150 partículas e o plano de geração possui 500cm para cada lado do detector, teremos uma área de, aproximadamente, $100m^2$.

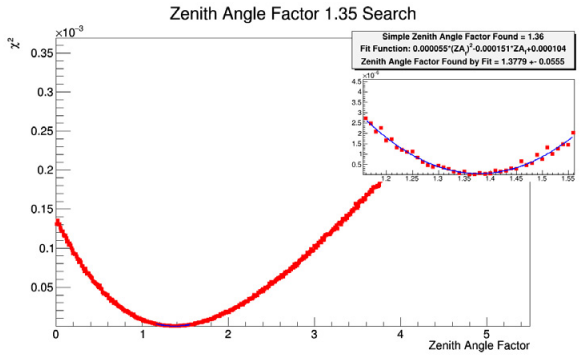
Logo, para 24h, o esperado de partículas geradas nesse plano é de, aproximadamente, 1 bilhão, sendo esse o número de eventos simulados nessa avaliação.

Para realizar a busca do fator, um método iterativo é utilizado, realizando a seguinte medida para cada valor de ZAF da tabela:

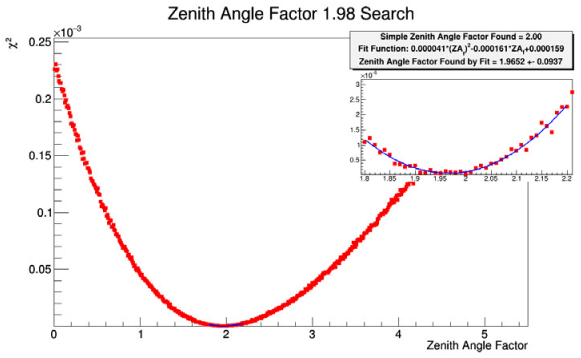
$$\chi^2 = \frac{\sum_0^{26} \left(R_{GrupoW(ChY,ChY+1)GrupoX(ChZ,ChZ+1)}(T) - R_{GrupoW(ChY,ChY+1)GrupoX(ChZ,ChZ+1)}(S)\right)^2}{27} \tag{3.2}$$

O χ^2 encontrado é o somatório dos quadrados das diferenças entre as 27 razões encontradas na tabela (T) para os 3 grupos, com as 27 razões geradas por meio da simulação realizada com o ZAF que se deseja procurar (S), normalizado. Em posse de todos os χ^2 obtidos, o valor mínimo, onde as razões da tabela se aproximam mais dos valores do ZAF simulado, é salvo.

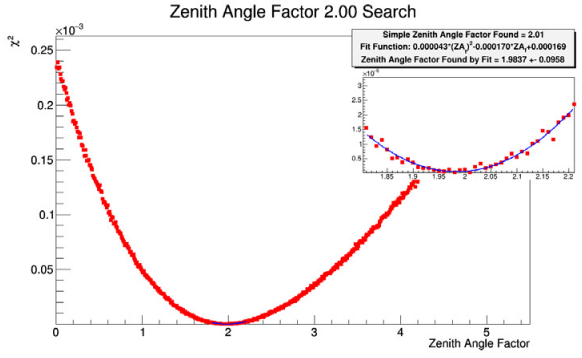
Em seguida, vinte pontos de χ^2 para cada lado do mínimo encontrado são ajustados em uma equação polinomial de 2º grau, obtendo um valor mínimo com maior grau de precisão. O método aqui apresentado foi validado, utilizando-se a simulação do fator para os valores: 1.35, 1.98, 2, 2.01, 2.51, 2.89. Esses valore foram escolhidos aleatoriamente, dentro da faixa de +-1 do valor de referência de, aproximadamente, 2, além da própria referência. A figura 3.8 mostra o método de procura graficamente, juntamente com o erro associado ao método pelo ajuste da função. Logo, observa-se que a técnica aqui utilizada possui um erro associado na ordem de 5%. Por ser uma simulação com as condições de posição inalteradas e uma condição geográfica de isotropia perfeita, esse resultado demonstra que a técnica possui um erro mínimo, sendo, no experimento, esperado um erro associado maior devido às limitações geográficas e de medida do posicionamento dos grupos do detector.



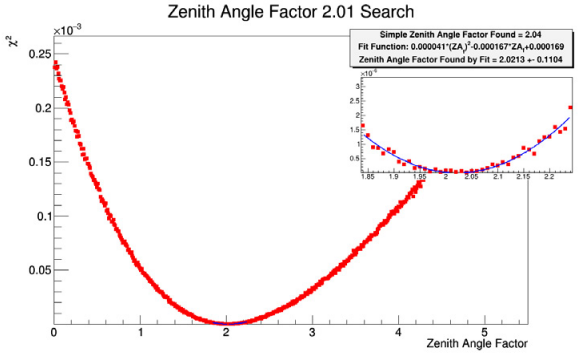
(a) ZAF=1.35



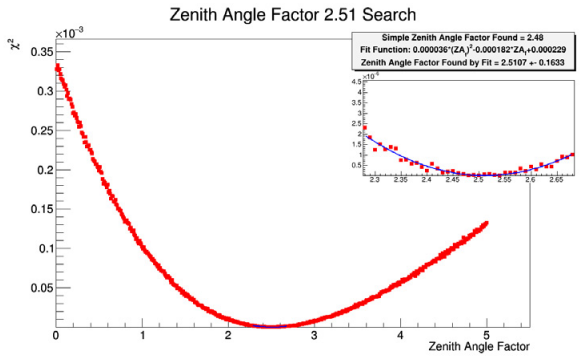
(b) ZAF=1,98.



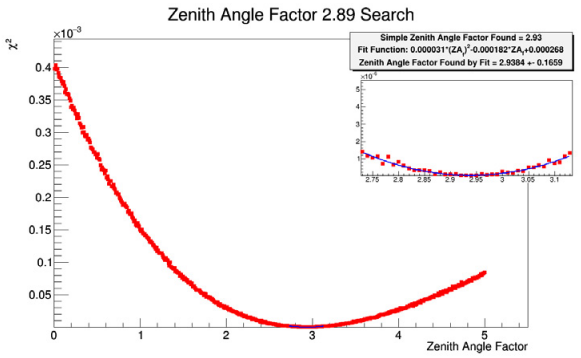
(c) ZAF=2,00.



(d) ZAF=2,01.



(e) ZAF=2,51.



(f) ZAF=2,89.

Figura 3.8: Validação do método de procura para valores ZAF simulados.

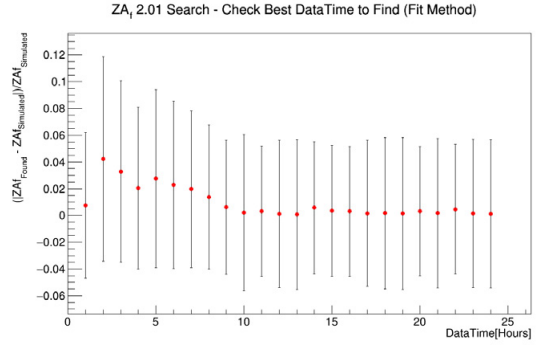
3.2.2 Análise da quantidade de dados necessária

Para validar a quantidade estatística de dados adquiridos, utilizados durante a análise, uma simulação, verificando a busca do valor em condições de menor tempo de aquisição, foi realizada, utilizando a mesma metodologia descrita na validação do método de procura. A busca do fator foi efetuada iterativamente para condição de 1h de dados até 24h, acrescentando o equivalente a 1h de dados a cada iteração.

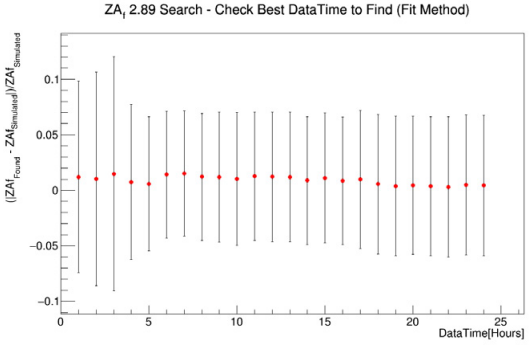
A análise exibida por meio dos gráficos da figura 3.9, verifica a variação do valor encontrado em relação ao valor esperado, normalizado. O resultado mostra que, independentemente do número de dados utilizados, em função da quantidade de horas de aquisição, o erro está compatível com o esperado associado ao método. Logo, em 1h de aquisição, já existe quantidade de dados estatisticamente suficiente para realizar a procura.

(a) ZAF=1,35.

(b) ZAF=1,98.



(c) ZAF=2,01.



(d) ZAF=2,89.

Figura 3.9: Análise da quantidade de dados necessária para procura do ZAF.

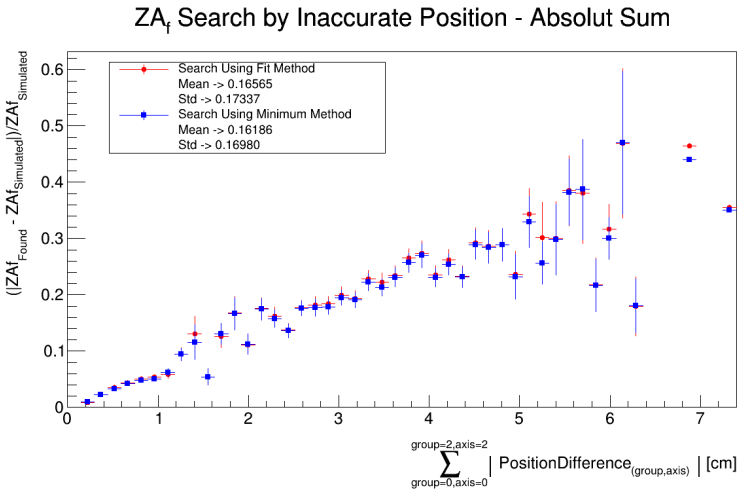
3.2.3 Validação do método por erro de medição

Inferindo-se que é inviável considerar que a medida do posicionamento dos grupos é efetuada sem desvios, uma reavaliação do método foi realizada, levando-se em conta as posições dos grupos, alteradas em relação às medidas utilizadas para criar a tabela de procura.

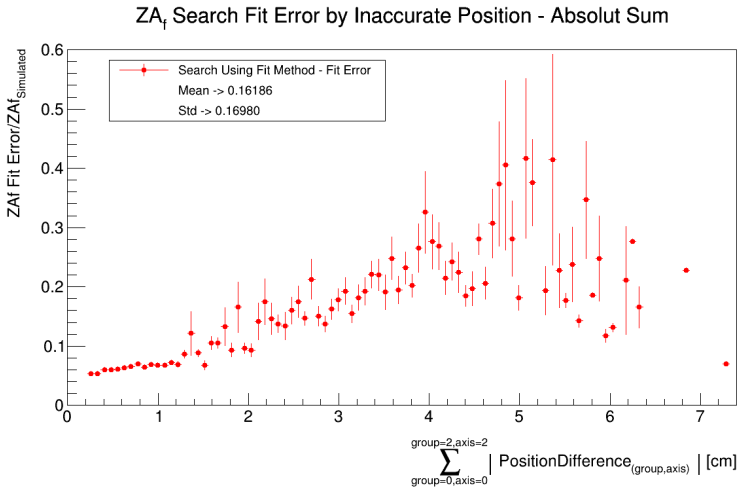
A metodologia utilizada consiste em gerar, aleatoriamente, desvios individuais para todos os eixos de cada grupo. Para gerar os valores aleatórios foi utilizada uma distribuição normal, com centro em 0cm e desvio-padrão de 0.5cm. Com o posicionamento alterado, é utilizado o método Monte Carlo, gerando o equivalente a 24h de aquisição de dados.

O processo foi repetido 2000 vezes, criando possibilidades de desvios diferentes. Assim como no processo de validação da quantidade de dados, foi obtido o desvio percentual do valor esperado para cada possibilidade, em função do somatório do módulo dos desvios. Todos os resultados foram inseridos em um histograma mostrado na figura 3.10.

Por meio do resultado, é possível observar que, caso o somatório dos erros das medidas de posição esteja em torno de 1cm, será esperado uma variação no resultado obtido de 10%, um acréscimo de 5% em relação ao erro intrínseco da técnica aqui utilizada. Isso significa que o resultado estará compreendido entre 1.8 e 2.2, considerando o valor referencial.



(a) Variação no valor do ZAF encontrado em função do desvio posicional.



(b) Variação no valor do erro do ajuste polinomial em função do desvio posicional.

Figura 3.10: Análise do erro do método de procura em função do desvio posicional.

Os códigos responsáveis pelas análises desses testes de validação do método de procura do ZAF encontram-se no apêndice O.

3.2.4 Teste de validação por erro geográfico

Com esta avaliação, analisou-se o impacto causado por grandes estruturas próximas ao local de instalação do experimento. Essas estruturas podem absorver parte das partículas, alterando a distribuição em função de θ e, por conseguinte, o ZAF encontrado pelo método. Vendo o mapa topográfico da figura 3.11, o modelo 3D da figura 3.12 e a foto na figura 3.13, observa-se a dimensão da montanha próxima ao refúgio. Para obter uma localização precisa do refúgio, foram utilizados os dados do GPS presente no DAQ e, com os valores de latitude e longitude obtidos, foi utilizado o *Google Maps*. Tendo em vista a complexidade para desenvolver uma simulação precisa do impacto causado pela montanha, escolheu-se utilizar os dados do comissionamento do detector durante a instalação para realizar essa avaliação, em 3 diferentes posicionamentos, rotacionando o detector em relação à montanha.

O resultado dessa avaliação será apresentada no próximo capítulo, onde será exibida a análise de dados do experimento.

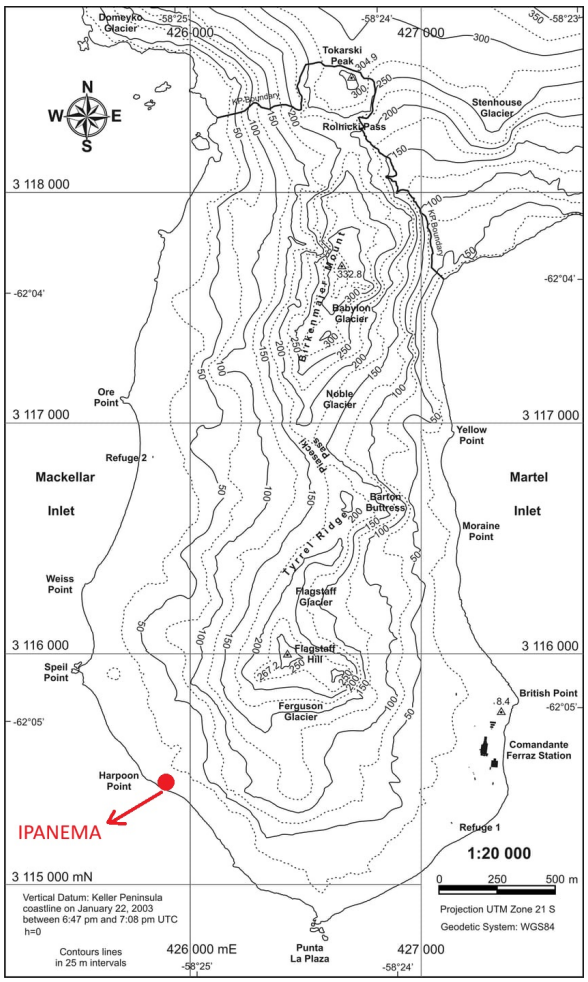


Figura 3.11: Mapa topográfico da península Keller, ilha do Rei George, adaptado com a localização do refúgio Ipanema em destaque[63].

Fonte: Mapa topográfico retirado do artigo "A new topographic map for Keller Peninsula, King George Island, Antarctica".

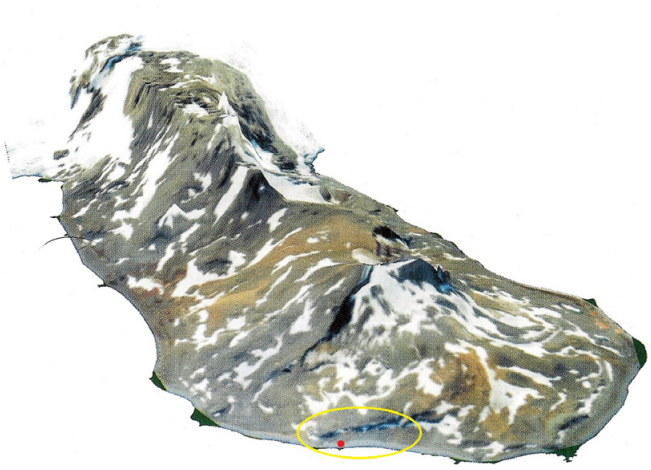


Figura 3.12: Visão 3D da península Keller modificada para destacar a cadeia montanhosa próxima ao refúgio Ipanema[64].
Fonte: "ANÁLISE MORFOMÉTRICA DA PENÍNSULA KELLER, ANTÁRTICA, ATRAVÉS DO SIG"

Figura 3.13: Imagem do refúgio com a montanha

Capítulo 4

Tratamento dos dados do Experimento CRE4AT

O sistema foi desenvolvido de forma que, diariamente, às 21h (GMT-3) ou à 0h (UTC), os dados são recebidos em um servidor localizado no laboratório multiusuário da coordenação de física de altas energias do CBPF. Após um período de 10 minutos, os dados são copiados para um segundo servidor de segurança e, nele, os dados são processados. Um diagrama de blocos com a sequência executada pode ser visto na figura 4.1.

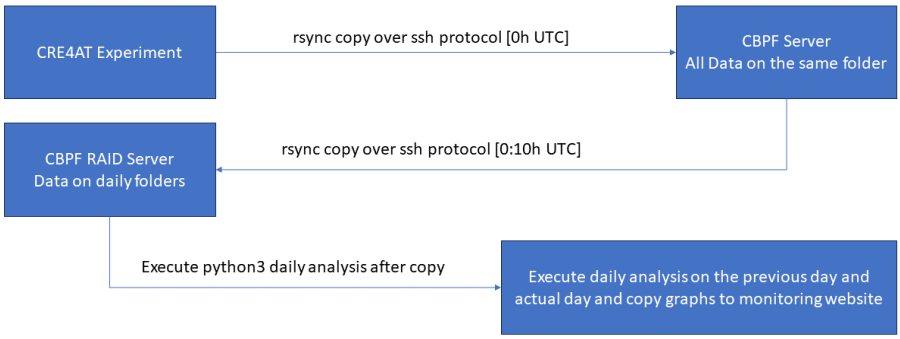


Figura 4.1: Fluxo dos dados executado desde o experimento até a análise.

Os dados recebidos pelo primeiro servidor são organizados em uma única pasta, da mesma forma como estão no computador local do experimento. Esses arquivos são enviados por meio do programa rsync de código aberto, que realiza a cópia, utilizando o protocolo seguro encriptado SSH. O rsync faz a cópia de forma eficiente, pois verifica quais arquivos já estão no destino e se não houve alteração, copiando somente os arquivos novos ou modificados, compactando-os antes do envio. Caso ocorra a perda de conexão, o rsync gerencia a perda de conexão de forma automática, continuando o processo após o restabelecimento da *internet*. O envio é iniciado pelo computador instalado no local do experimento, por intermédio do gerenciador agendado de processos cron.

Em seguida, os arquivos são enviados para um servidor dentro do *datacenter* do laboratório, nomeado de RAID, reorganizando os dados por pastas diárias. Após a cópia, o processo de análise é automaticamente iniciado. No *shell script*, que gerencia a cópia entre servidores do CBPF, foi desenvolvido um método que verifica se os dados foram recebidos e, caso seja identificada a existência de problemas no recebimento desses dados, por exemplo, caso haja falta de conexão, algo comum devido ao mau tempo na Antártica, um arquivo é criado, no qual os dias que estão faltando são adicionados. O *shell script* encontra-se no apêndice P. Quando a cópia é realizada após restabelecimento da conexão, todos os dias que não foram analisados e estão identificados no arquivo são, primeiramente, organizados no RAID e analisados, para,

posteriormente, proceder à análise do dia.

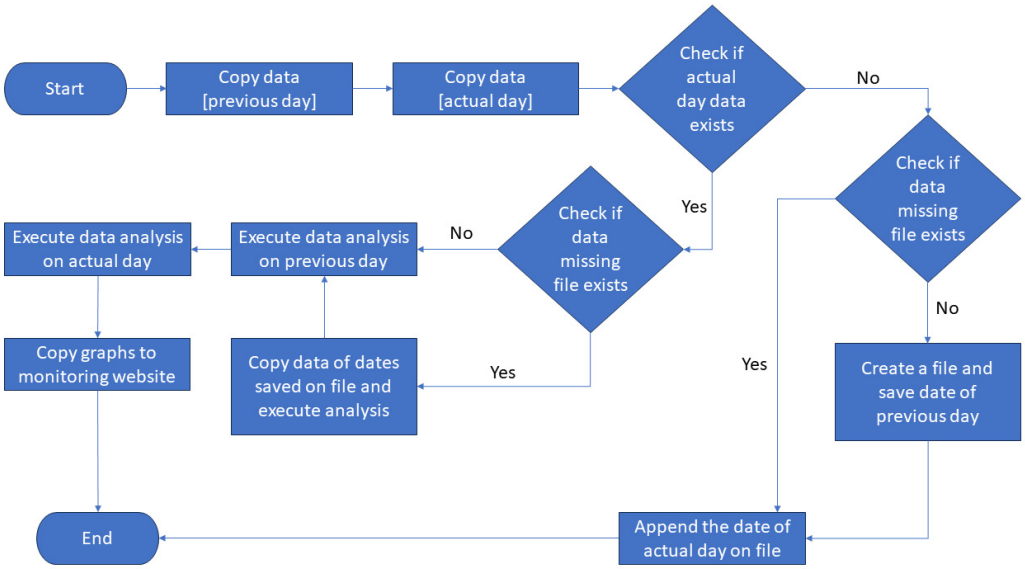


Figura 4.2: Fluxograma da execução do *shell script* diário responsável pela cópia e análise dos dados.

Como mostrado na figura 4.2, a cópia e a análise dos dados do dia anterior são necessárias, pois, quando os dados desse dia são enviados, é possível que o último arquivo do dia não esteja completo. Consequentemente, no dia seguinte, quando a análise é executada novamente, os dados restantes que entraram no último arquivo serão analisados e os gráficos de monitoramento atualizados.

4.1 Análise de dados

A análise da dados do experimento foi desenvolvida de tal forma que um único código em *python* executa não somente a análise do fluxo, da procura por *bursts* e do ZAF, mas também é responsável pela criação dos gráficos pertinentes à avaliação diária dos dados recebidos, exporta os dados para geração dos gráficos de longa duração com todos os dados já processados e exporta um arquivo ROOT, gerado com o *framework* do ROOT/CERN, onde todos os dados sem modificação salvos pelo DAQ e os dados processados são adicionados, facilitando, posteriormente, análises adicionais. A figura 4.3 mostra a sequência executada durante o processo e o código encontra-se no apêndice Q.

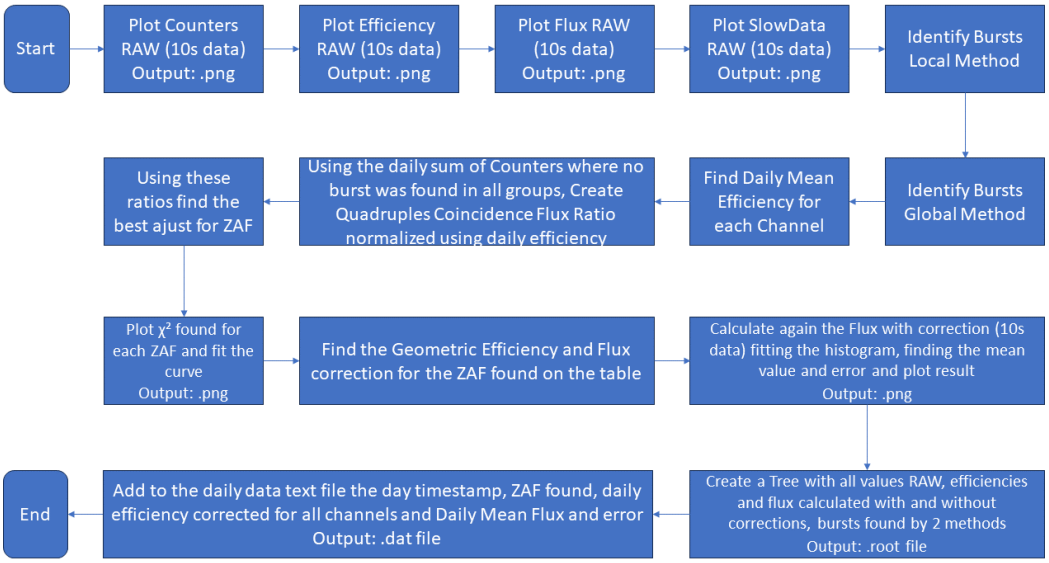
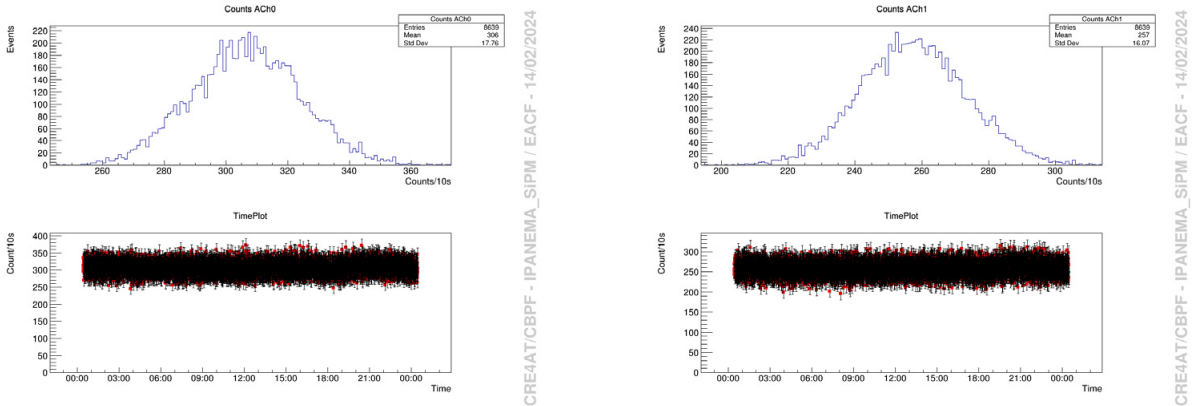


Figura 4.3: Fluxograma do processo de análise dos dados do experimento CRE4AT.

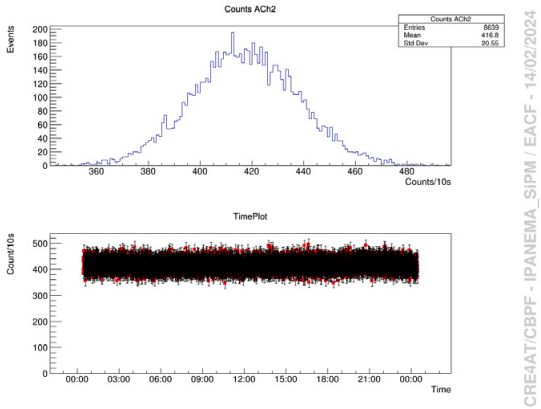
Os dados de comissionamento serão utilizados durante a instalação local do detector que foi enviado para o refúgio Ipanema. Picos nas contagens poderão ser notados em alguns dias na análise de *bursts*, mas esses dados não significam aumento repentino do fluxo, pois foram devidos às interferências causadas por transmissões de rádio, utilizado para contato do Refúgio com a EAFC. Os dias e horários em que o rádio é utilizado são anotados para serem removidos dos dados úteis do estudo posteriormente, uma vez que os dados podem ser contaminados com ruído eletromagnético. Durante a análise do trabalho, esses aumentos serão utilizados para testar o método de identificação de *bursts*, juntamente com os dados do protocolo de injeção de luz, que é acionado automaticamente no primeiro dia de cada mês, sendo inserida nesse trabalho a injeção do dia 01/03/2024. Para as primeiras análises, os dados do dia 14/02/2024 serão utilizados, pois nesse dia não houve comunicação de rádio e nenhum pico foi gerado. Assim, será possível mostrar as contagens e eficiências sem influência provocada pelo ruído.

Inicialmente, os dados de cada arquivo de 1h são adicionados a uma tabela e dela são criados gráficos com os valores dos contadores de cada canal, das coincidências duplas dentro de cada grupo, das coincidência triplas e quádrupla. Não são criados gráficos para os contadores responsáveis pela coincidências cruzadas entre grupos utilizados na procura do ZAF. Alguns exemplos desses gráficos podem ser vistos nas figuras 4.4, 4.5, 4.6 e 4.7, para o grupo A.

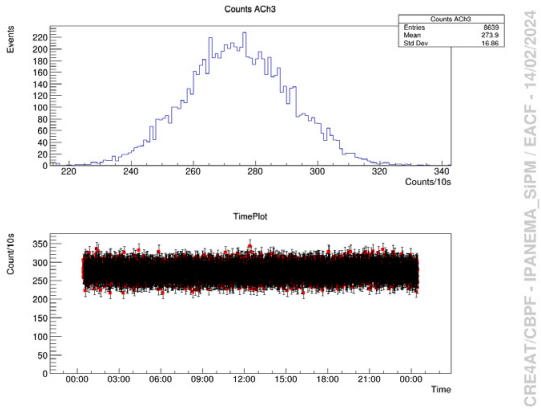


(a) Canal 0.

(b) Canal 1.



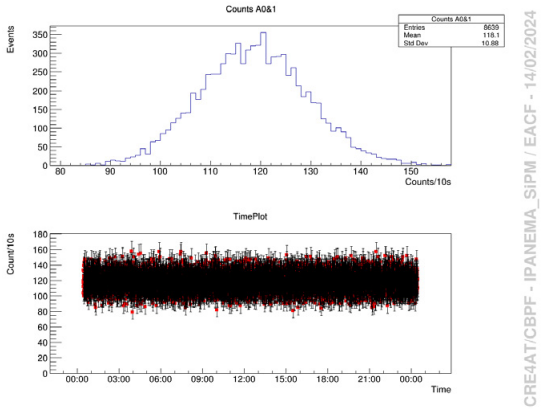
(c) Canal 2.



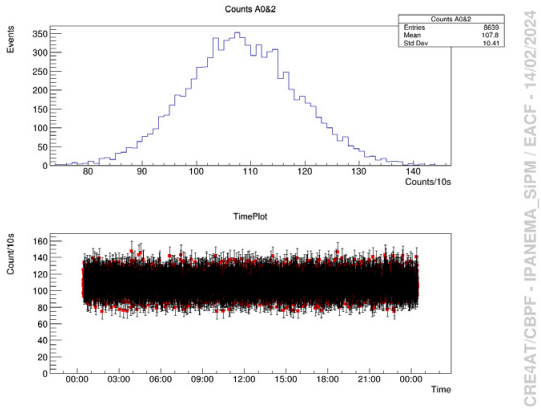
(d) Canal 3.

Figura 4.4: Gráficos dos contadores dos canais do grupo A.

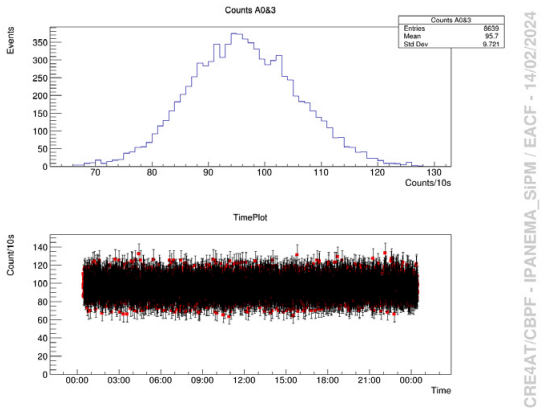
Analisando-se as imagens acima, em virtude de algumas variações intrínsecas de canal a canal, observa-se que as contagens obtidas, que possuem relação com a tensão de limiar escolhida, apresentam uma variação grande entre canais, indo de uma média de contagens de 257 para o canal 1 até 416 para o canal 2. Dentre as variações citadas, percebem-se a tolerância dos valores de resistência, que fazem parte do ganho e do ajuste do *shaper*, e a precisão óptica da fibra WLS em relação à área sensível do SiPM.



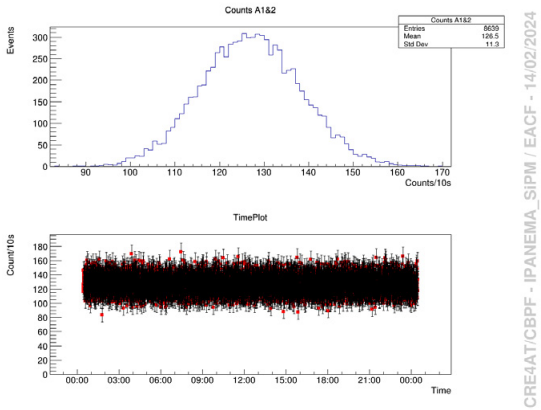
(a) Coincidência entre canal 0 e 1.



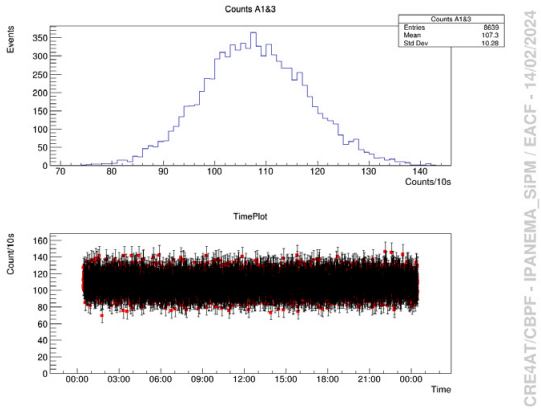
(b) Coincidência entre canal 0 e 2.



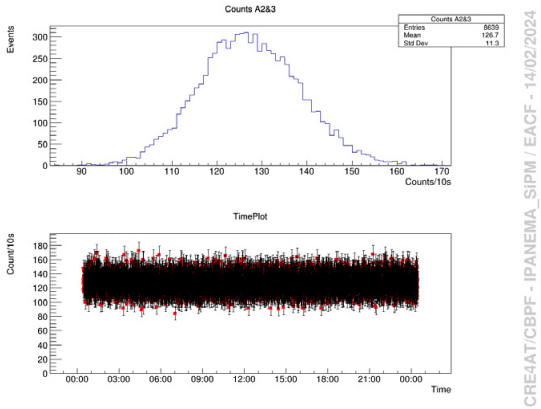
(c) Coincidência entre canal 0 e 3.



(d) Coincidência entre canal 1 e 2.



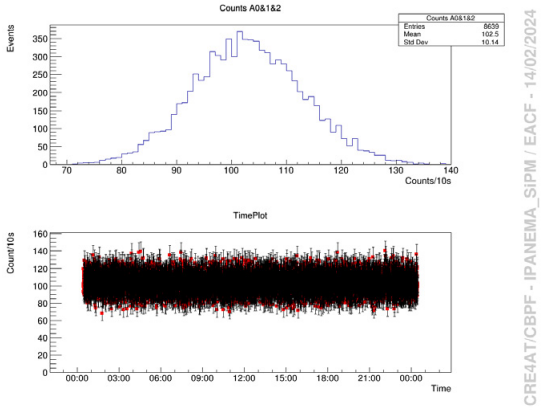
(e) Coincidência entre canal 1 e 3.



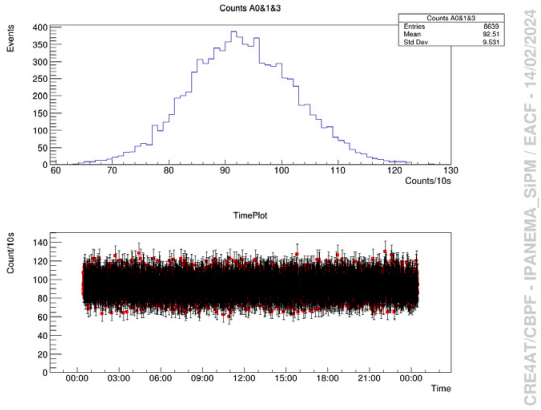
(f) Coincidência entre canal 2 e 3.

Figura 4.5: Gráficos dos contadores das coincidências duplas do grupo A.

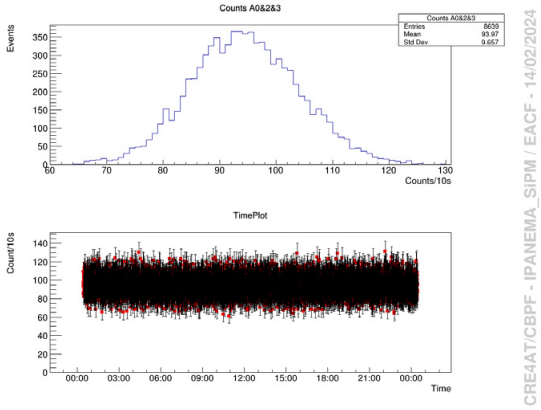
Utilizam-se as coincidências para a eliminação de possíveis ruídos eletrônicos e corrente de escuro que tenham passado a tensão de limiar e, indesejavelmente, entraram na contagem. A diferença dos valores obtidos entre contadores é reduzida gradualmente das coincidências duplas até a quádrupla, como pode ser visto.



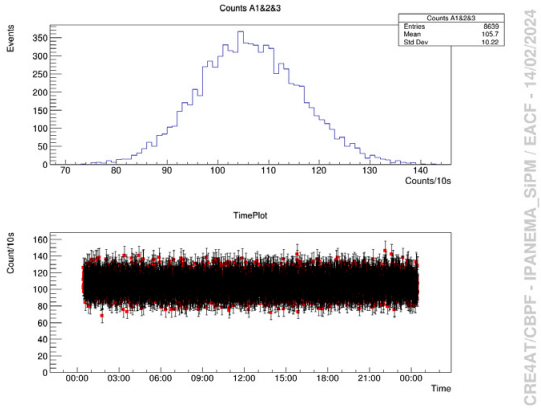
(a) Coincidência entre canal 0, 1 e 2.



(b) Coincidência entre canal 0, 1 e 3.



(c) Coincidência entre canal 0, 2 e 3.



(d) Coincidência entre canal 1, 2 e 3.

Figura 4.6: Gráficos dos contadores das coincidências triplas do grupo A.

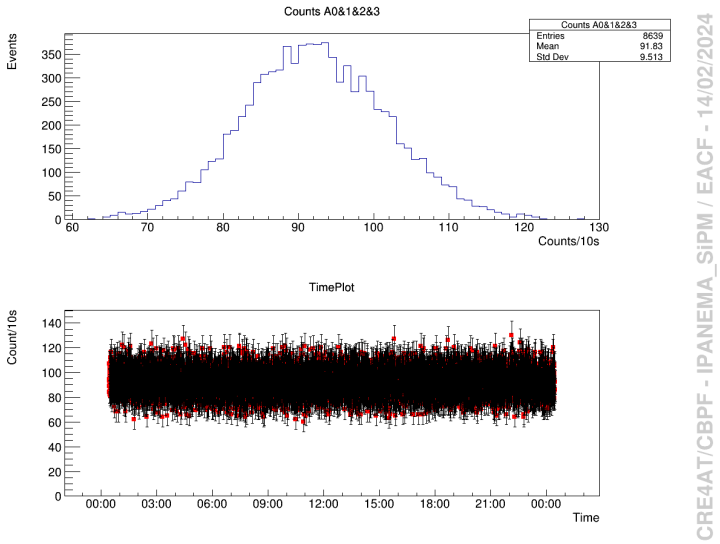


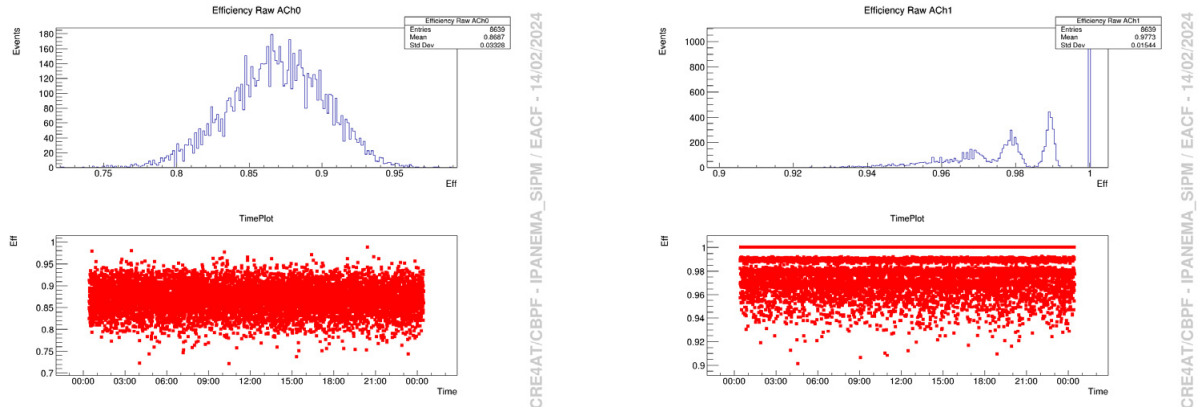
Figura 4.7: Gráfico do contador de coincidência quádrupla do grupo A.

4.1.1 Fluxo e eficiência

Em seguida, o valor da eficiência de cada canal é obtido. A eficiência é usada para medir percentualmente quantas partículas passaram por um canal sem serem detectadas. Pelo método utilizado neste trabalho, ela é calculada de forma relativa aos outros canais e isto significa que, se todos os canais tiverem baixa eficiência igualmente, por exemplo, caso o *threshold* ajustado seja alto demais, o resultado obtido será próximo do valor máximo, 1 ou 100%, não medindo a real eficiência do detector. A eficiência é influenciada por uma série de fatores, desde as configurações da eletrônica, como a relação entre o ganho e o valor de limiar utilizado, mas também de fatores mecânicos/ópticos. Essa eficiência relativa de cada canal é obtida através da seguinte equação:

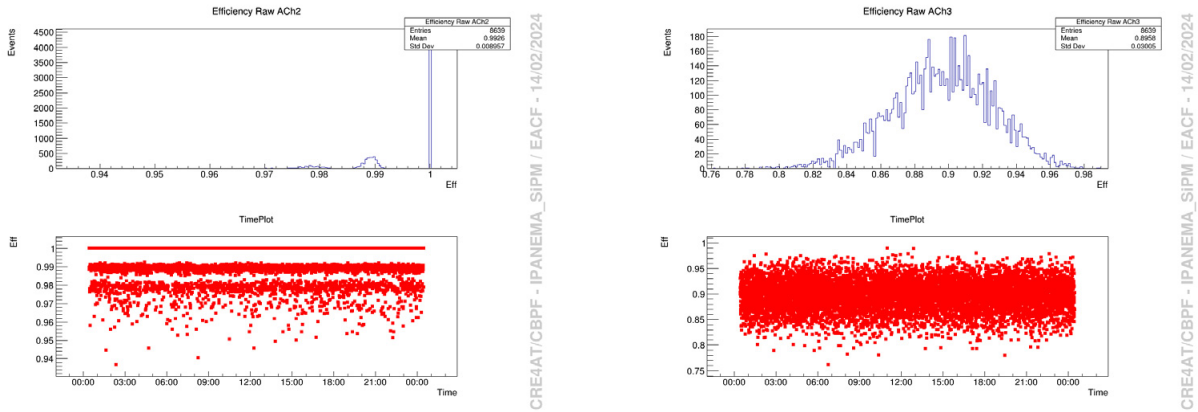
$$Eff_{GxChy} = \frac{Coin4_{Gx}}{Coin3_{GxChs_{all-Chy}}} \tag{4.1}$$

Sendo Eff_{GxChy} a eficiência de um canal 'y' de um grupo 'x', $Coin4_{Gx}$ a coincidência quadrupla do grupo 'x' e $Coin3_{GxChs_{all-Chy}}$ a coincidência tripla do grupo 'x', contendo todos os canais exceto o canal do qual se deseja obter a eficiência. Aplicando essa relação, o resultado obtido para cada um dos 4 canais do grupo A pode ser visto na figura 4.8.



(a) Canal 0.

(b) Canal 1.

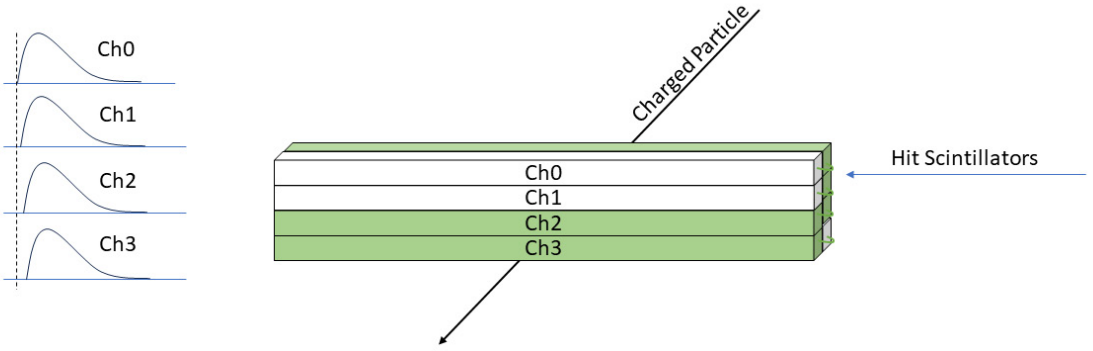


(c) Canal 2.

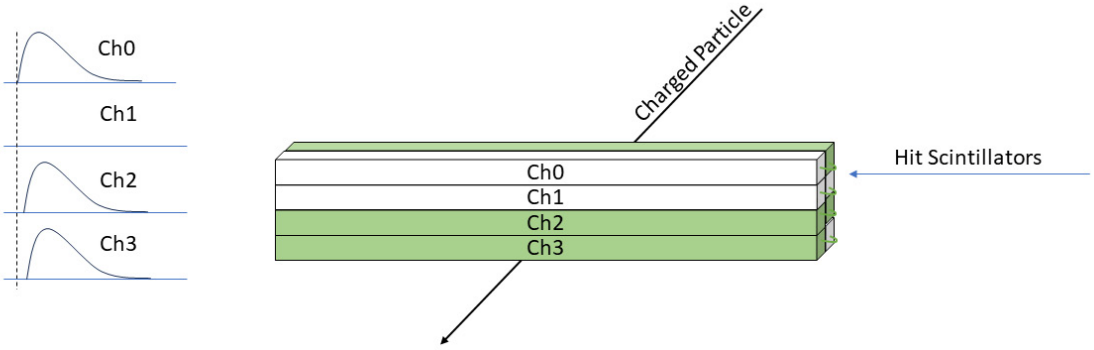
(d) Canal 3.

Figura 4.8: Gráficos da eficiência relativa dos canais do grupo A.

Uma questão a ser observada nesse processo está em relação à influência da geometria do detector na medida relativa de eficiência. Como é visto na figura 4.9a, o esperado com a passagem de uma partícula carregada pelos planos de detecção será um sinal coincidente nos 4 canais que formam o grupo e, em alguns casos, um ou mais canais podem não ter um pulso de saída, como mostrado na figura 4.9b. Neste caso, a medida de eficiência relativa desse plano mostrará de forma correta a relação percentual de perdas da detecção deste plano.



(a) Todos os planos detectam a partícula passante.



(b) Um dos planos não detecta a partícula passante.

Figura 4.9: Esquema do sinal gerado por cada canal com a passagem da partícula.

Porém, é observado na figura 4.8 que os canais externos do grupo possuem uma eficiência mais baixa do que os canais no interior do grupo. Essa diferença ocorre por uma questão

geométrica intrinsecamente atrelada à medida que é realizada. Como os planos de detecção possuem uma espessura, algumas partículas atravessam apenas alguns deles, devido ao ângulo de incidência com que os atingem, como pode ser visto na figura 4.10. Neste caso, a contagem tripla será incrementada e a eficiência reduzida de forma equivocada no plano onde não ocorre a passagem, pois a falta de detecção pelo plano externo está correta. Neste caso, uma correção é necessária para ajustar o valor, sendo ela obtida por meio do modelo de produção.

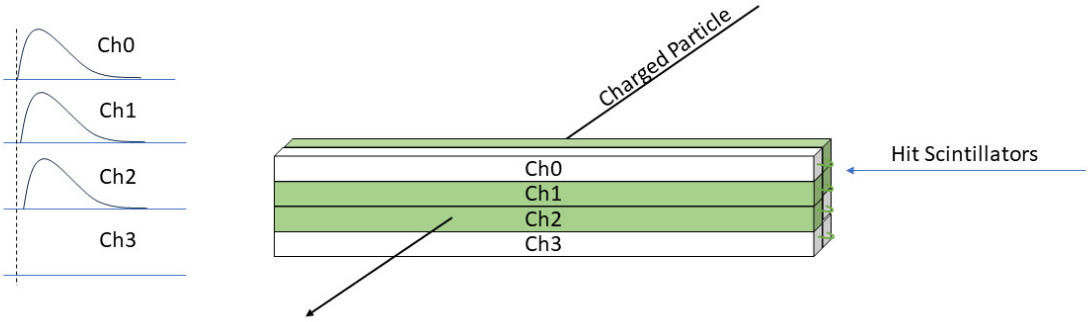


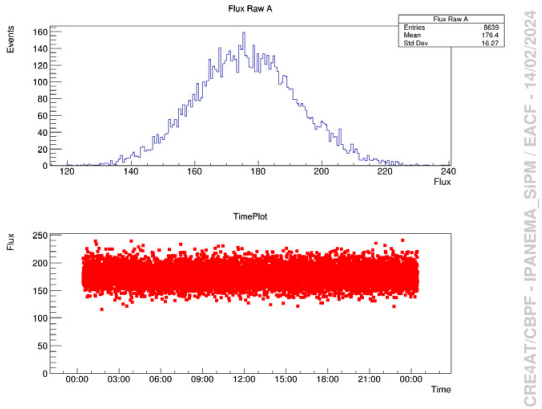
Figura 4.10: Problema da geometria do detector na eficiência.

Para realizar essa correção, é necessário obter uma relação percentual da quantidade de vezes que uma partícula passa por 3 planos em relação aos 4 do grupo. Essa relação é dependente do modelo de múons cósmicos e, consequentemente, do ZAF utilizado. Durante a simulação, onde a tabela de procura do ZAF é criada, essa relação que corrigirá o valor dessas eficiências é salva para cada fator simulado. Aplicando essa correção, o fator geométrico é e removido. Sendo assim, a eficiência passa a medir somente a relação de perdas nas medidas do traço, obtendo o que se deseja medir. Como na simulação a eficiência de cada plano é perfeita, isto é, quando uma partícula passa pelo plano, o sinal será sempre contabilizado, o valor que corrige cada eficiência será obtido pelo inverso da relação utilizada para medir as eficiências, pois essa relação $\frac{Coin3_{GxChs_{all-Chy}}}{Coin4_{Gx}}$ obterá a proporção de traços que passam somente por 3 planos em relação aos que passam pelos quatro planos, de acordo com o modelo de produção. Como o valor de correção depende do ZAF, a correção somente é obtida após o valor do fator ser encontrado.

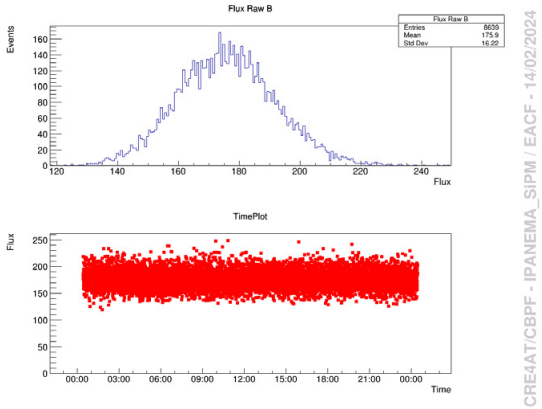
Com o valor das eficiências sem correção geométrica de cada plano, em cada grupo, é realizado o cálculo do fluxo de partículas carregadas que passam pelos 4 planos, seguindo a equação:

$$Fluxo_{Gx} = \frac{Coin4_{Gx}}{Eff_{GxCh0} \times Eff_{GxCh1} \times Eff_{GxCh2} \times Eff_{GxCh3} \times Área_{plano} \times T_{\text{Aquisição}}} \tag{4.2}$$

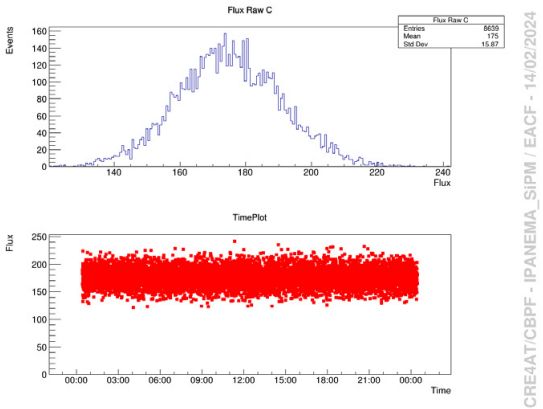
O cálculo do fluxo leva em consideração a contagem em coincidência de todos os planos de um grupo, corrigindo a contagem pela eficiência medida de cada plano, neste caso ainda não corrigida, e dividindo essa contagem pela área e pelo tempo de aquisição. Assim, obtém-se para cada grupo uma medida de fluxo independente e os gráficos com os valores medidos podem se vistos na figura 4.11.



(a) Grupo A.



(b) Grupo B.



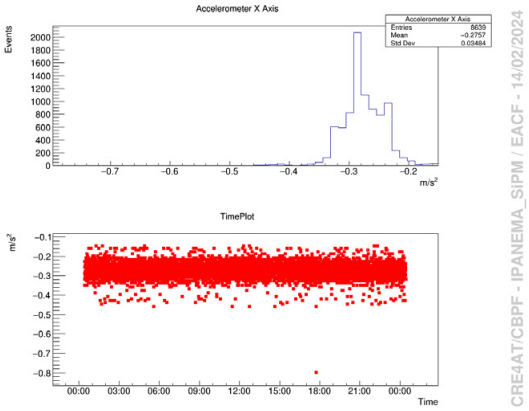
(c) Grupo C.

Figura 4.11: Gráfico das medidas de fluxo para cada grupo sem correção geométrica.

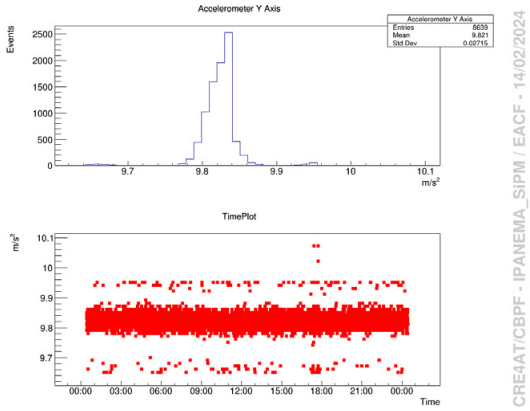
É notável que o cálculo do fluxo é realizado em uma área superficial. Entretanto, para obtê-lo foram utilizados 4 cintiladores que possuem uma espessura e, assim, essa medida se afasta do pressuposto utilizado. Em virtude disto, uma correção geométrica também se faz necessária nesta medida e, assim como no caso da eficiência, são dependentes do modelo e, logo, do fator utilizado. Assim, na simulação realizada para cada ZAF, também é inserida uma relação entre a contagem do primeiro canal em relação à coincidência quadrupla dos planos. Como o detector não possui um plano de espessura infinitesimal sem obstáculos superiores e inferiores, os planos que podem mais se aproximar desse ideal são os planos externos, que possuem outro plano em somente uma de suas faces e com uma espessura 4 vezes inferior à do grupo.

4.1.2 Monitoramento dos sensores

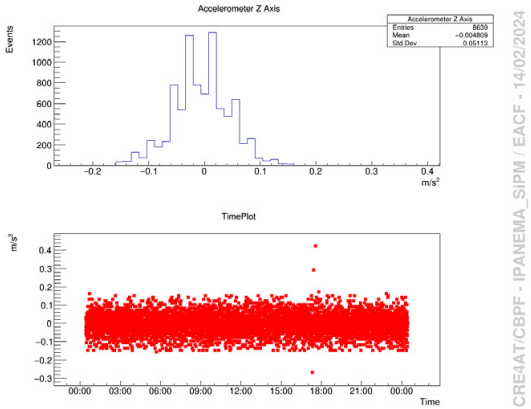
Em seguida, são criados os gráficos com os valores obtidos dos sensores do experimento (*slow control*). Estão inclusos os dados dos sensores ambientais como pressão, umidade, temperatura, acelerômetro, giroscópio, magnetômetro e dados de monitoramento do sistema, como o valor da alta tensão aplicada e o valor da temperatura lida pelo conversor DC-DC de alta tensão, que é utilizado para correção de ganho.



(a) Eixo X.



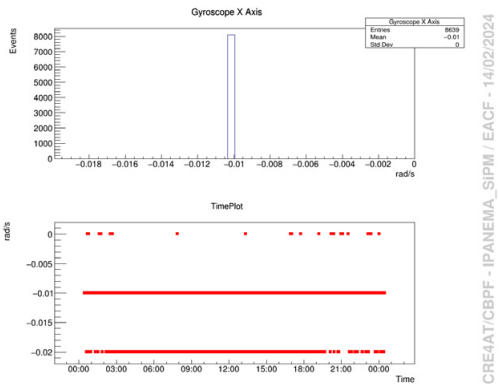
(b) Eixo Y.



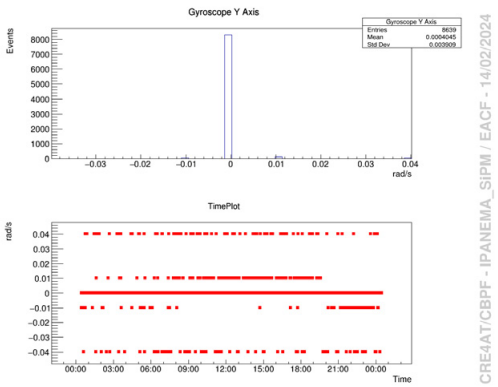
(c) Eixo Z.

Figura 4.12: Gráficos com dados do acelerômetro para os 3 eixos.

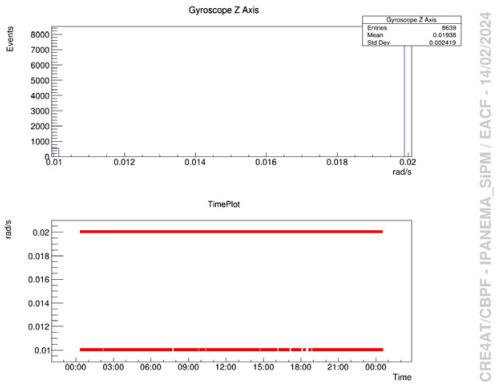
Por conta do posicionamento da placa utilizado, com o sensor na horizontal, é possível ver que a aceleração da gravidade, em vez de ser medida no eixo Z do sensor, sua maior componente é medida pelo eixo Y. Por meio do valor em cada eixo do sensor, é possível obter o ângulo que a caixa contendo o experimento forma em relação à normal. Essa medida é extremamente relevante ao longo da tomada de dados, visto que a medida do fator ZAF e do fluxo são influenciados em função da inclinação que o experimento possui. De posse desses valores, caso haja alguma alteração nas condições, será factível realizar uma correção nos dados obtidos.



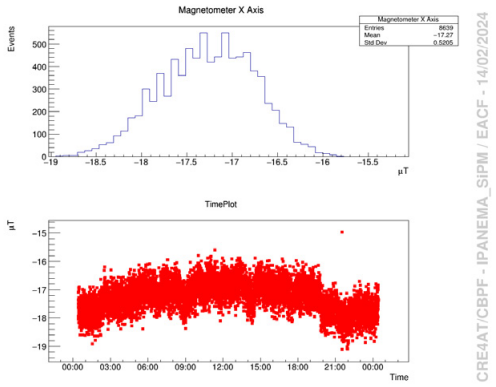
(a) Giroscópio eixo X.



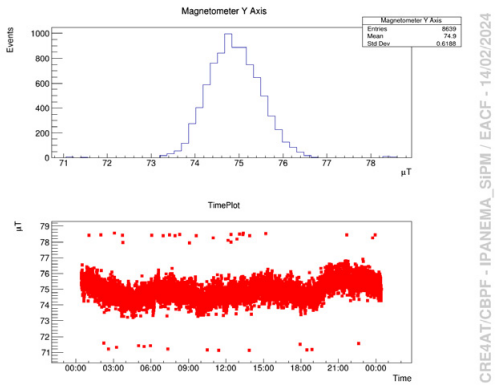
(b) Giroscópio eixo Y.



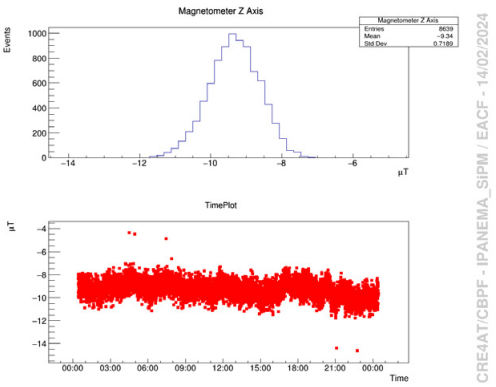
(c) Giroscópio eixo Z.



(d) Magnetômetro eixo X.

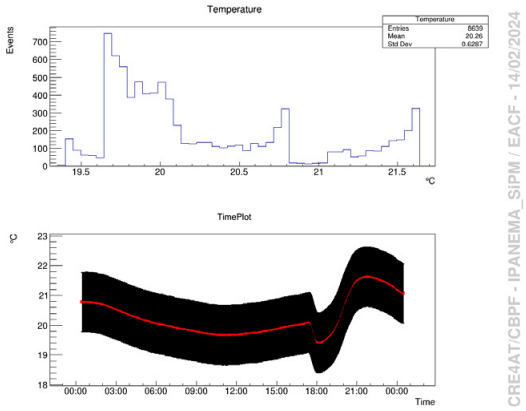


(e) Magnetômetro eixo Y.

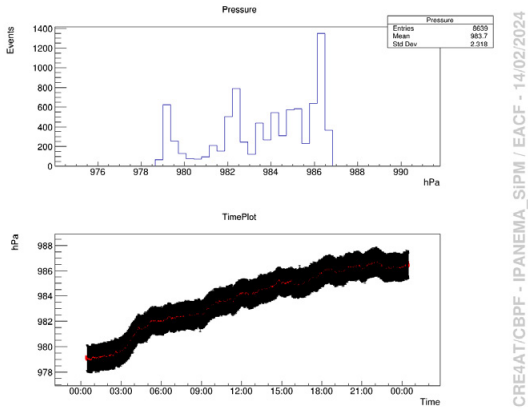


(f) Magnetômetro eixo Z.

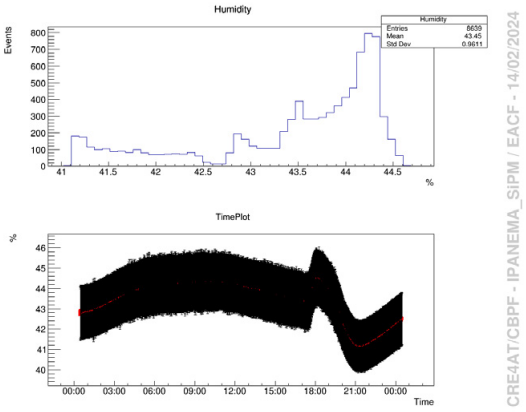
Figura 4.13: Gráficos com dados do acelerômetro e magnetômetro para os 3 eixos.



(a) Temperatura.

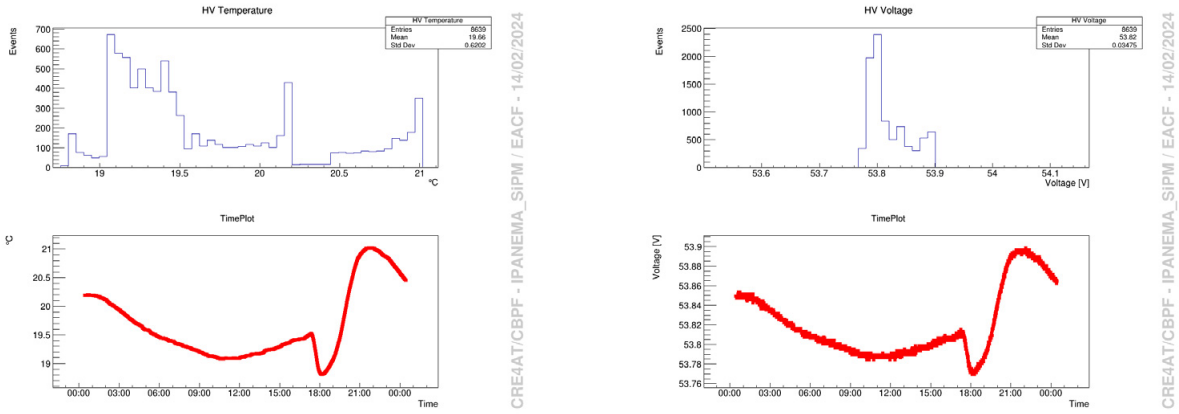


(b) Pressão.



(c) Umidade.

Figura 4.14: Gráficos de temperatura, pressão e umidade.



(a) Temperatura.

(b) Alta Tensão.

Figura 4.15: Gráficos gerados pelos dados obtidos da fonte de alta tensão.

Observe que a temperatura obtida pelo sensor LM94021BIMG/NOPB, produzido pela *Texas Instruments*, ligado ao DC-DC da *Hamamatsu* é coerente, dentro da faixa de erro, com a medida obtida pelo sensor BME280, mostrado na figura 4.14a. Essa medida é utilizada para corrigir a tensão de saída do conversor, que segue suas variações mantendo a relação linear de correção, e, desse modo, o ganho do SiPM é corrigido. O valor da temperatura aqui monitorado sofre variações ao longo do dia, devido a um sistema de controle (PID) criado por outro integrante do grupo, para manter a temperatura mais alta em Ipanema. A temperatura do refúgio foi configurada para ficar em 20°C e o sistema trabalha ligando e desligando um aquecedor.

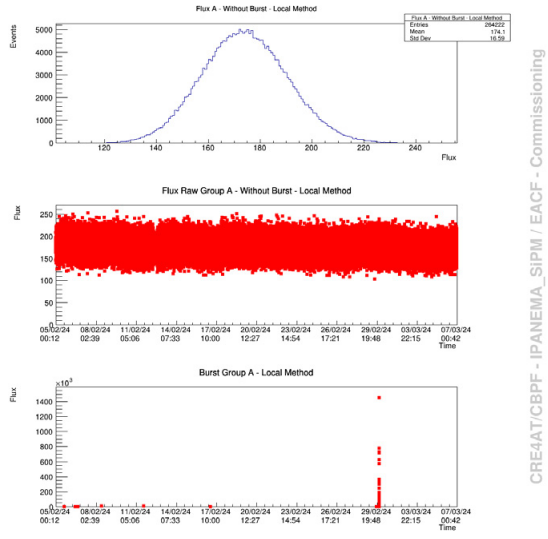
4.1.3 Métodos de procura de aumento de fluxo abrupto *Bursts*

De posse dos valores de fluxo de cada grupo, um processo de busca por *bursts* é realizado. Essa busca é essencial para duas vertentes que o projeto possui: na busca pelo fator ZAF e na procura por processo de formação de nuvens embrionárias. Para realizar a busca, dois métodos distintos são utilizados. No primeiro método, chamado de método local dentro do contexto do experimento, uma busca por processo de varredura sobre os dados é realizada. Este método possui maior sensibilidade a variações diárias do fluxo. Já o método global é realizado sobre o histograma, com os dados diários gerando um fator em relação à distribuição, que pode ser facilmente reanalisado posteriormente.

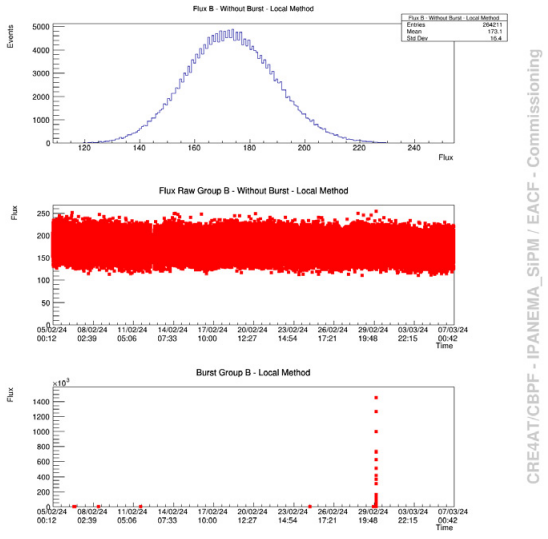
4.1.3.1 Método local

Neste método, uma série temporal com os dados de fluxo de um grupo é analisada por meio de uma varredura. Essa varredura é realizada utilizando-se um subconjunto de tamanho fixo da série temporal. O subconjunto é utilizado para obter a média e o desvio-padrão dos dados que fazem parte dele. Com esses valores, o ponto seguinte de fluxo é analisado e, caso seu valor seja maior que a média em 5 vezes o desvio-padrão, ele será considerado um chuveiro atmosférico e a análise seguirá para o próximo ponto sem deslocar os dados que fazem parte do subconjunto, até encontrar um valor que não seja considerado um chuveiro, que será inserido nesse bloco e o valor mais antigo removido. A varredura é feita tanto da esquerda para direita, logo, de 10 em 10s de dados adquiridos, seguindo do início para o fim do dia, quanto da direita para esquerda fazendo o sentido inverso do dia.

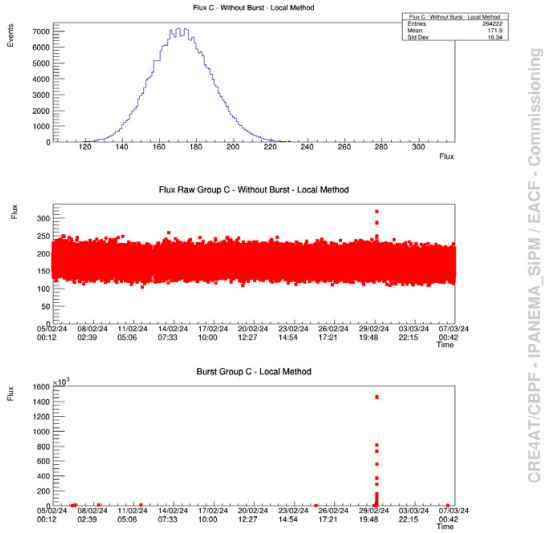
Inicialmente, esse subconjunto possui somente 5 valores de fluxo medidos. Nesse contexto, os 5 primeiros pontos do dia, da esquerda para direita, e os 5 últimos, da direita para esquerda, não serão analisados, uma vez que fazem parte da amostra inicial. Após percorrer nos dois sentidos, eliminando os 5 pontos iniciais e finais, é realizada a análise, se os resultados para os dois casos são iguais. Se forem iguais, esse resultado será utilizado. Para os 5 primeiros pontos, o resultado obtido da direita para esquerda será utilizado, e dos 5 últimos, o resultado obtido percorrendo no sentido inverso. Não sendo os resultados iguais, o subconjunto é aumentado de tamanho em 1 posição, logo, nesta conjuntura, ele terá 6 valores de fluxo, e a análise é realizada novamente. Esse processo segue até o limite de 50 posições dentro do bloco. Se chegar no limite, o valor de 25 posições será utilizado e uma lógica de coincidência entre os dois sentidos será realizada, com exceção dos extremos onde o mesmo processo acima mencionado será utilizado. O resultado final é um conjunto booleano, sendo os zeros correspondentes a fluxos sem *bursts* e uns a identificação de *bursts*. Os gráficos da figura 4.16 mostram o processo de identificação, separando os dados marcados como *bursts* dos dados identificados como fluxo padrão, para o período de 01/02/2024 até 06/03/2024.



(a) Grupo A.



(b) Grupo B.



(c) Grupo C.

Figura 4.16: Gráficos gerados pelo método local de identificação de aumento de fluxo abrupto.

Chama a atenção que, durante a injeção de luz executada no dia 01/03, alguns valores acima do normal entram no gráfico pertencente às medidas sem *bursts*. Isso ocorre devido a uma característica que acontece durante o protocolo do LIS. Como entre uma aquisição de dados e outra ocorre a troca dos valores configurados, gerando contagens aleatórias, algumas acabam ficando um pouco acima da média dos RCG. Apesar de estar acima, como essa medida está contida dentro de 5σ, ela não é considerada *burst* e seu valor é adicionado ao subconjunto. Com esse valor acrescentado, a média acaba aumentando, assim como o σ, permitindo que valores um pouco maiores sejam acrescentados também. Devido a esse fator, alguns valores de fluxo até próximo de 300 $\frac{partculas}{m^2 s}$ aparecem no gráfico.

4.1.3.2 Método global

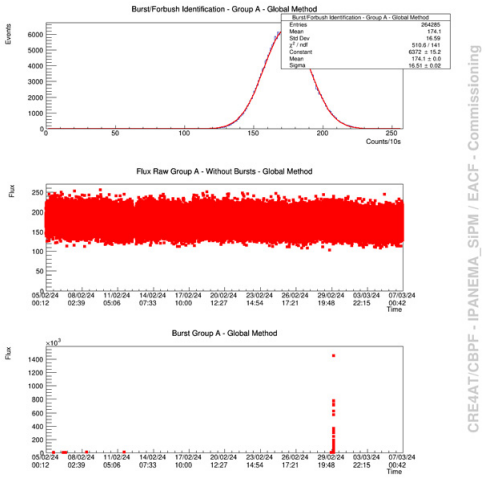
No método global, os valores medidos de fluxo são adicionados a um histograma. A distribuição obtida é ajustada com uma gaussiana e os valores de média e desvio-padrão são salvos. Com esses valores, é gerado um fator para cada ponto, seguindo a seguinte equação:

$$fator = \frac{Fluxo - \mu}{\sigma} \tag{4.3}$$

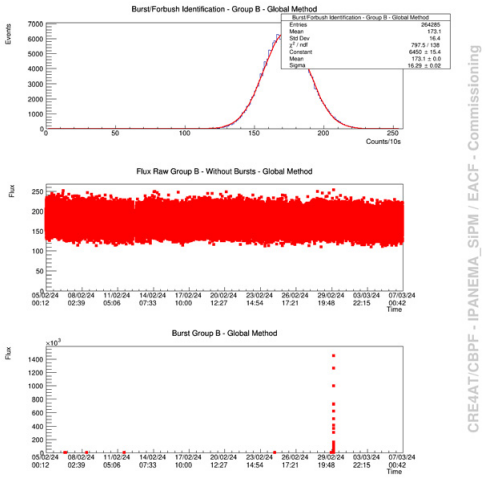
onde μ é a média da gaussiana ajustada e σ o desvio-padrão.

Por meio desse fator, é possível identificar quantas vezes a diferença do fluxo medido à média possui em relação ao desvio-padrão. Depois, esses fatores passam por um corte de 5, selecionando os pontos que possuem acima de 5 vezes o desvio-padrão, que serão considerados *bursts* e abaixo, sendo considerados decréscimo de *Forbush*. Apesar de essa técnica não possuir sensibilidade a variações diárias de fluxo, já que todos os valores são adicionados ao histograma e o ajuste gaussiano leva em conta todos os dados do dia. Contudo, por ser gerado um fator na saída que é salvo em um arquivo de dados, caso seja necessário realizar um corte diferente, poderá ser facilmente realizado sem precisar passar pelo processo novamente, diferentemente do método local, que gera somente um vetor de saída que informa se aquele ponto é considerado um *burst* ou não.

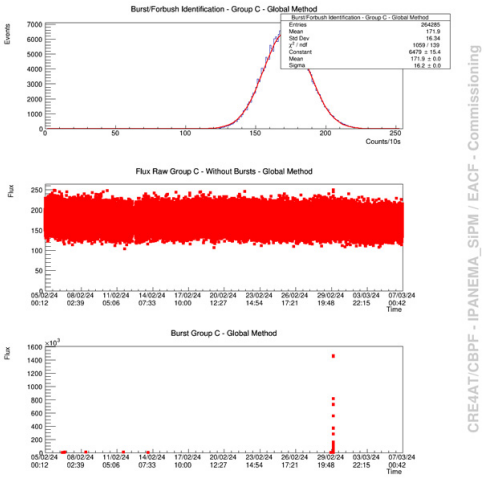
Na figura 4.17, é visível a separação dos *bursts*. Diferentemente do método anterior, como todo os dados são levados em conta no ajuste gaussiano da curva, a precisão em separar eventos ruidosos, como durante o processo de injeção de luz, é claramente superior.



(a) Grupo A.



(b) Grupo B.



(c) Grupo C.

Figura 4.17: Gráficos gerados pelo método global de identificação de aumento de fluxo abrupto.

Além do funcionamento do método para remoção dos valores de fluxo que ultrapassam a variação estatística, uma observação é realizada em função da estabilidade da medida sem *bursts*, dentro da flutuação normal, apesar da grande diferença de temperatura presente ao longo desse período de comissionamento, como pode ser visto na figura 4.18. Esse resultado confirma o funcionamento do sistema de ajuste linear de ganho dos SiPMs, em função da temperatura.

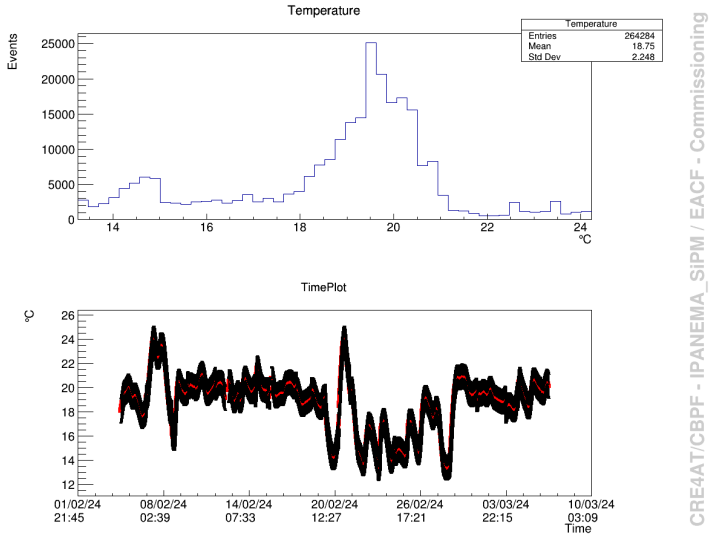


Figura 4.18: Gráfico da temperatura medida ao longo do período de comissionamento.

4.1.4 Procura pelo fator ZAF

De posse da análise de procura de *bursts*, a busca pelo fator é factível, pois, sem essa informação, não é possível realizar essa procura de forma confiável, já que durante o evento de um chuveiro intenso, diversas partículas atravessam os planos de detecção e é possível que ocorram coincidências cruzadas entre grupos, sem necessariamente serem pertencentes a um único traço, isto é, a uma partícula cruzando os dois grupos dentro da faixa de ângulo que o detector aceita. Com isso, se múltiplas partículas cruzarem os grupos dentro do período do *latch* selecionado, algo provável durante um evento de chuveiro, será contado como um evento, porém esse evento é invalido, deteriorando a precisão do resultado.

Logo, para aumentar a precisão da medida, os eventos de *bursts* são eliminados dos dados que serão utilizados durante a procura do ZAF. Como dois métodos de identificação de *bursts* são realizados, os dois possíveis resultados podem ser utilizados para seleção dos dados. Como padrão, foi escolhido o método global para realizar essa separação e os dados selecionados são aqueles que não são considerados *bursts* nos três grupos simultaneamente.

Para realizar a comparação com a tabela gerada com a simulação, os valores da série temporal de cada contador são somados, obtendo uma contagem por dia, após o corte, eliminando os *bursts*. Com esses valores, é possível realizar o cálculo do fluxo cruzado. Para o cálculo do fluxo, as eficiências de cada canal são necessárias e, para obter um valor diário mais preciso, são utilizados os valores calculados a cada 10s de dados, retirando os valores que foram eliminados durante o corte, e a média é calculada. Para o cálculo do fluxo entre os canais 0 e 1 dos grupos A e B, por exemplo, a seguinte equação de fluxo é utilizada:

$$Fluxo_{A01\&B01} = \frac{\text{Coincidência}_{A01\&B01}}{Eff_{A0} \times Eff_{A1} \times Eff_{B0} \times Eff_{B1} \times \text{Área} \times T_{\text{Aquisição}}} \tag{4.4}$$

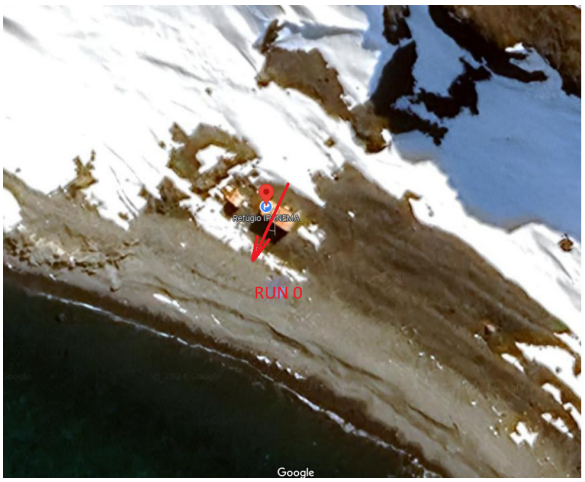
Para realizar a procura na tabela, a mesma razão utilizada durante a simulação, com uma pequena modificação, será utilizada:

$$R_{A01\&B01} = \frac{Fluxo_{A01\&B01}}{Fluxo_A + Fluxo_B} \tag{4.5}$$

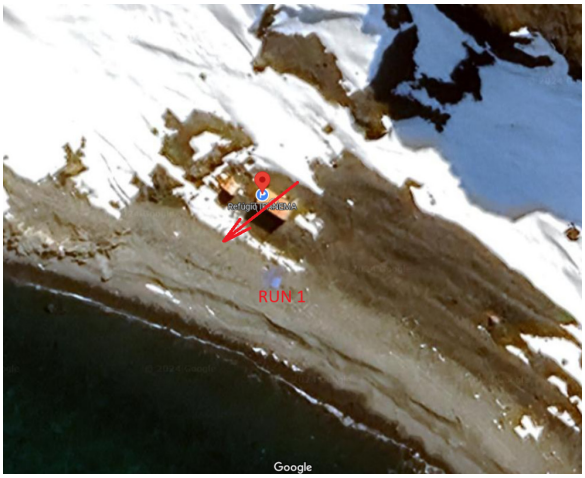
O fluxo é utilizado em vez da coincidência quádrupla, devido à correção do valor pela eficiência dos planos de detecção, que, na simulação, é 100%. Após a obtenção dos valores das

razões, o mesmo método de obtenção do valor mínimo de χ^2 e, posteriormente, o ajuste com uma equação polinomial de grau 2 é utilizado, obtendo-se o melhor valor de ZAF.

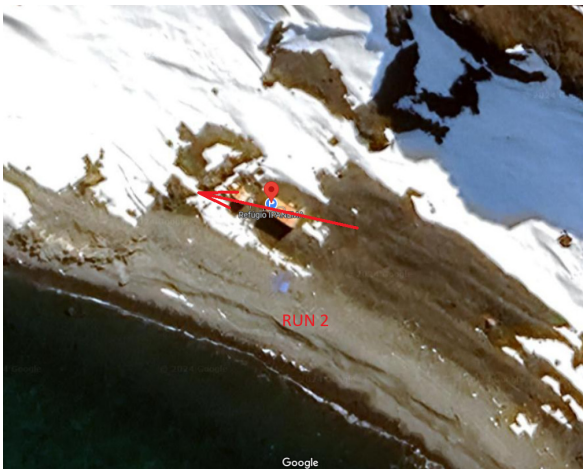
Para analisar a influência das montanhas próximas ao refúgio, os três posicionamentos escolhidos foram os seguintes, mostrados na figura 4.19:



(a) Posição dos cintiladores durante o primeiro período de aquisição.



(b) Posição dos cintiladores durante o segundo período de aquisição.



(c) Posição dos cintiladores durante o terceiro período de aquisição.

Figura 4.19: Posicionamento dos cintiladores em relação à montanha para os três períodos de aquisição.

Como não havia uma maneira precisa de medir o posicionamento, a primeira posição foi escolhida de forma visual em relação à montanha, tentando colocar o eixo seccionado do detector perpendicular à montanha. Nas posições seguintes, tentou-se girar aproximadamente 45° cada. Posteriormente, para localizar as posições geográficas do detector foram utilizadas as medidas do magnetômetro presente no DAQ e corrigido o norte geográfico em relação ao norte magnético terrestre.

Uma média dos valores medidos pelo magnetômetro a cada período de aquisição foi realizada e, como o DAQ foi colocado em uma posição vertical, o eixo Y com o eixo Z do sensor foram trocados. Os seguintes valores foram obtidos:

Tabela 4.1: Resultados da medida de campo magnético para o primeiro posicionamento

Run0	Média [μT]	Desvio-padrão [μT]
X	-9.88	0.91
Y	+74.81	1.05
Z	-15.19	1.14

Tabela 4.2: Resultados da medida de campo magnético para o segundo posicionamento

Run1	Média [μT]	Desvio-padrão [μT]
X	-17.19	0.53
Y	+74.94	0.59
Z	-8.74	1.05

Tabela 4.3: Resultados da medida de campo magnético para o terceiro posicionamento

Run2	Média [μT]	Desvio-padrão [μT]
X	-17.99	0.88
Y	+73.86	1.03
Z	+7.30	1.02

Utilizando os eixos X e Z para obter os ângulos do vetor campo magnético em cada aquisição, foram obtidos os seguintes valores:

Tabela 4.4: Ângulos dos vetores campo magnético para cada aquisição

Run	Ângulo [Graus]
0	237
1	207
2	158

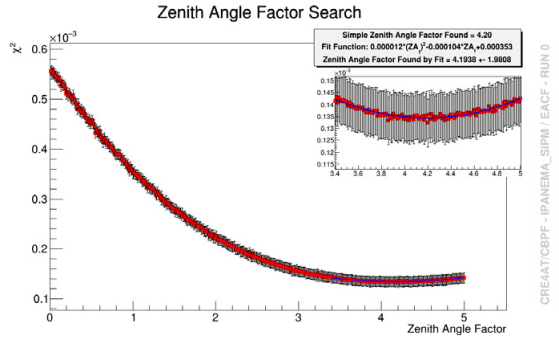
Esses ângulos são obtidos em relação ao norte magnético terrestre. Para convertê-los em direções geográficas, a declinação magnética local é necessária. O valor foi obtido por meio do site `https://www.ngdc.noaa.gov/geomag/calculators/mobileDeclination.shtml#WMM`, que, para localização de Ipanema, foi de aproximadamente +10° para todo o período de aquisição. Logo, para obter o ângulo correto geográfico foi necessário acrescentar 10° aos valores encontrados na tabela 4.4. Essas direções foram representadas nas imagens da figura 4.19.

As aquisições foram realizadas nos seguintes períodos:

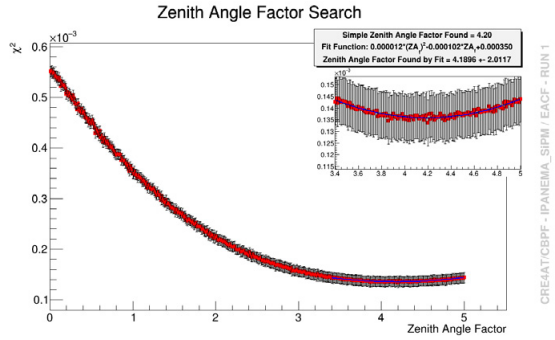
Tabela 4.5: Período de aquisição para cada rotação

Run	Data de início [UTC +0]	Data de fim [UTC +0]
0	05/02/2024 - 00h:12min	12/02/2024 - 14h:04min
1	12/02/2024 - 19h:52min	15/02/2024 - 16h:23min
2	15/02/2024 - 21h:39min	20/02/2024 - 18h:55min

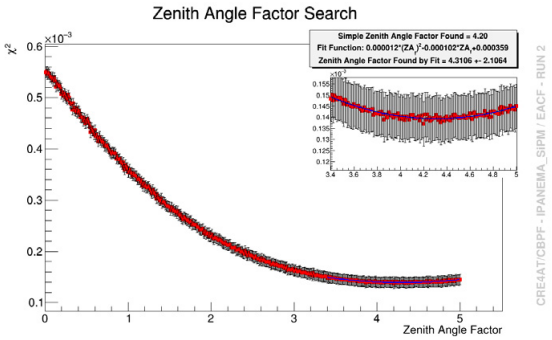
O resultado obtido para cada período de aquisição pode ser visto na figura 4.20.



(a) ZAF encontrado para o primeiro período de aquisição.



(b) ZAF encontrado para o segundo período de aquisição.



(c) ZAF encontrado para o terceiro período de aquisição.

Figura 4.20: Valores de ZAF obtidos para cada período de aquisição.

Para realizar o ajuste da curva, 80 pontos de cada lado em torno do mínimo de χ^2 foram utilizados, devido a uma menor curvatura encontrada nos dados do detector em relação aos dado simulados, aumentando a margem de erro no ajuste. Os valores obtidos de ZAF ficaram em torno de 4, sendo maior no terceiro posicionamento, quando a região seccionada do detector estava praticamente paralela à montanha. De acordo com dados obtidos para múons atmosféricos de diversas faixas energéticas, quanto menor a energia do múon menores são os ângulos na distribuição obtida, como é possível ver na figura 4.21.

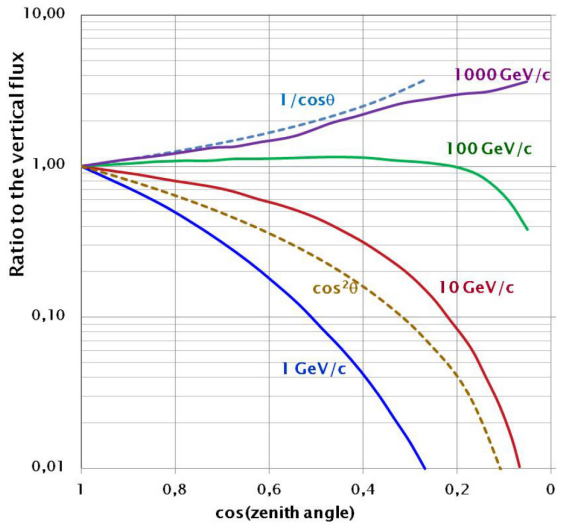


Figura 4.21: Distribuição angular de múons a nível do solo, por energia[65]. Retirado do artigo "Atmospheric muons: experimental aspects".

Fazendo uma adaptação desse gráfico, foi gerado um novo, realizando um ajuste sobre a curva dos múons com 1GeV, utilizando a equação $\cos^n \theta$, sendo n a variável. Além desse ajuste, foram adicionadas às curvas $\cos^2 \theta$, $\frac{1}{\cos \theta}$, $\cos^4 \theta$ e $\cos^{4.5} \theta$, sendo os dois últimos escolhidos por serem próximos aos valores encontrados.

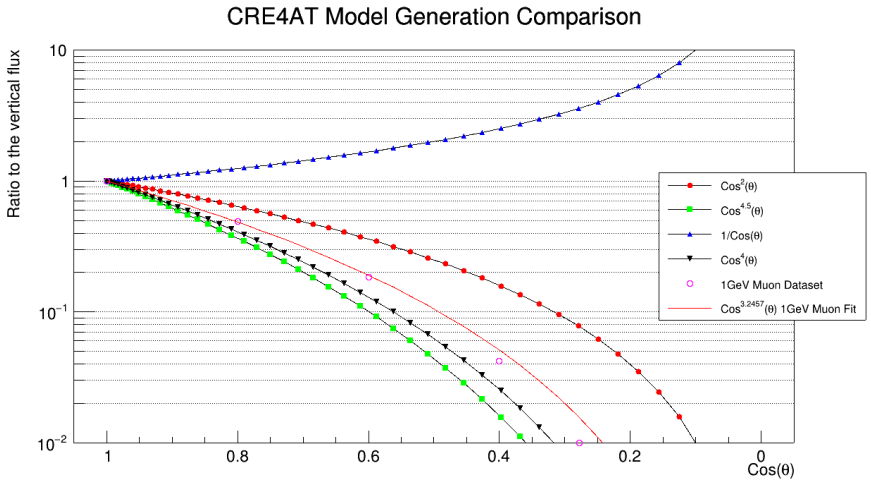


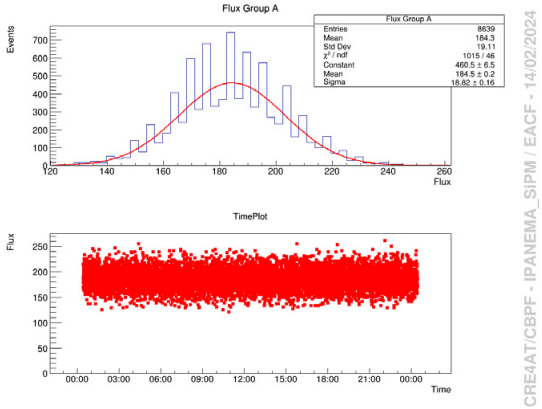
Figura 4.22: Gráfico comparando possíveis valores do ZAF com os dados de múons 1GeV e ajuste do modelo a esses dados.

Assim, como já mostrado por meio da análise feita no capítulo anterior, na figura 3.1, é possível observar que, quanto maior o valor de ZAF, menores são os valores da distribuição em ângulos θ maiores. Esse resultado mostra que o aumento no valor do ZAF encontrado quando o detector estava aproximadamente paralelo à montanha é correto, já que a montanha absorve parte das partículas que iriam compor a distribuição em ângulos θ maiores. Ao mesmo tempo, é notável que, para múons da faixa de 1GeV, o valor de ZAF que melhor ajusta o modelo é de, aproximadamente, 3.25, assim como ZAF de 4 e 4.5 estão próximos dos dados para múons de 1GeV. Para o resultado obtido nos primeiro e segundo períodos de aquisição, onde o detector estava mais perpendicular à montanha mais próxima do refúgio, duas explicações poderiam guiar para o resultado obtido: apesar de estarem mais distantes, nas laterais de Ipanema ainda existem montanhas que podem afetar a distribuição em ângulos maiores, onde é mais crucial a medida para diferenciar os modelos e, devido a características locais, os múons que podem chegar na região possuem energias menores, tendo uma distribuição naturalmente menor nos maiores ângulos e o modelo que melhor ajusta com ZAF maior.

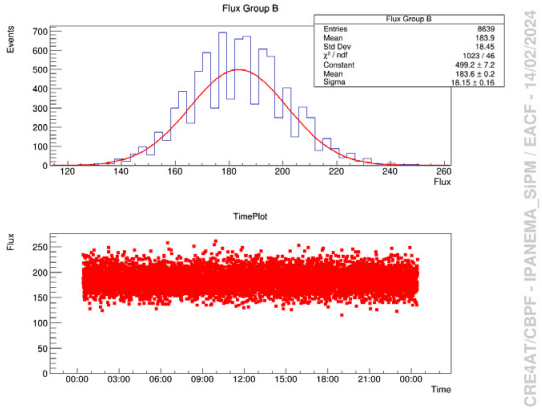
4.1.5 Correções pelo ZAF

Em posse do valor do ZAF, uma série de correções são executadas, aumentando a precisão dos valores obtidos na análise. Primeiramente, como o valor de ZAF é obtido por meio do ajuste do polinômio, o valor encontrado de ZAF é arredondado para 2 casas decimais, para possibilitar a obtenção de uma solução na tabela gerada na simulação. Em seguida, são extraídos os valores de correção das eficiências e do fluxo.

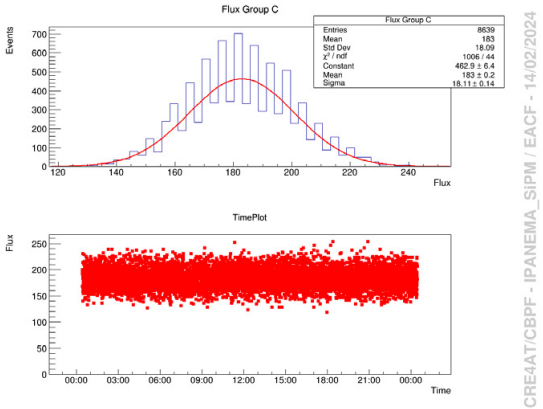
As eficiências obtidas a cada 10s de aquisição de dados são corrigidas, assim como o valor médio diário. O fluxo a cada 10s também é corrigido e para o seu cálculo são utilizados os valores de eficiência média corrigida, como pode ser visto na figura 4.23.



(a) Grupo A.



(b) Grupo B.



(c) Grupo C.

Figura 4.23: Gráficos de fluxo corrigidos com ajuste gaussiano.

Com os valores corrigidos de fluxo, um ajuste gaussiano é realizado sobre os dados onde não existe chuveiro, obtendo a média diária e o erro da média. Esses valores, juntamente com o ZAF encontrado e as eficiências diárias corrigidas, são salvos em um arquivo que é atualizado diariamente, para realizar o monitoramento desses valores durante todo o período de aquisição.

Todos os dados adquiridos, juntamente com os dados processados durante a análise, são salvos no fim em um arquivo ROOT, para posterior processamento em novas técnicas e novos estudos.

4.2 Análise dos dados de injeção de luz - LED UV

O protocolo de calibração do experimento, executado uma vez por mês, no dia 1°, às 00h (UTC), gera uma sequência de dados com injeção de luz, por meio do LED UV instalado no sistema óptico. Para analisar esses dados, um programa que exporta uma tabela para cada grupo foi desenvolvido, mostrando, de forma normalizada pela frequência de injeção esperada, quantos pulsos foram contados em cada canal. As colunas indicam o valor percentual da injeção para o canal mostrado e a linha mostra qual canal ou canais foram injetados naquela aquisição.

Como as colunas e linhas são colocadas na mesma ordem, é esperada uma sequência de uns diagonalmente na tabela, indicando que o valor injetado naquele canal ou canais foi contado sem perdas. Com essa calibração, é analisado tanto o funcionamento correto de cada canal

individualmente e em grupo, nas injeções das duplas, triplas e quádrupla, quanto *crosstalk* entre canais, seja eletrônico, seja óptico. Nas tabelas exibidas abaixo, são apresentados os resultados do protocolo para o primeiro grupo com as injeções de luz de 1kHz, 5kHz, 10Khz, 25kHz, 50kHz e 100kHz.

Tabela 4.6: Resultados da injeção de luz em 1kHz e 5kHz - Grupo A

Injeção de 1kHz															
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
ACh0	1.001	0.002	0.002	0.004	0.003	0.000	-0.000	-0.000	-0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000
ACh1	0.005	1.000	0.043	0.002	0.005	-0.001	-0.000	0.046	-0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000
ACh2	-0.000	0.001	1.012	0.022	-0.001	0.000	0.000	0.001	0.000	0.022	0.000	-0.000	0.000	0.000	0.000
ACh3	0.004	0.003	0.278	1.020	0.001	0.002	0.001	0.000	0.001	0.278	0.001	0.001	0.001	0.001	0.001
A0&1	0.999	1.003	0.124	-0.000	0.999	0.120	-0.001	0.120	-0.002	-0.001	0.120	-0.001	-0.001	-0.002	-0.001
A0&2	1.002	0.016	1.026	0.036	0.013	0.998	0.035	0.014	-0.000	0.037	0.014	-0.001	0.035	-0.000	-0.001
A0&3	0.998	0.002	0.440	1.027	0.004	0.440	0.998	-0.000	0.003	0.442	-0.000	0.003	0.441	0.000	0.000
A1&2	0.012	1.002	1.016	0.035	0.010	0.010	0.001	1.001	0.035	0.035	0.010	0.001	0.001	0.035	0.001
A1&3	0.009	0.998	0.735	1.019	0.008	0.006	0.007	0.731	1.000	0.733	0.006	0.008	0.005	0.730	0.005
A2&3	0.003	0.005	1.038	1.035	0.000	0.001	0.001	0.003	0.002	1.005	0.000	0.001	0.001	0.002	0.001
A0&1&2	1.003	1.000	1.037	0.077	0.999	1.000	0.078	1.000	0.079	0.079	1.000	0.078	0.078	0.079	0.078
A0&1&3	1.004	1.000	0.862	1.022	1.000	0.860	1.000	0.860	1.000	0.862	0.860	1.000	0.860	0.860	0.860
A0&2&3	1.004	0.027	1.047	1.047	0.028	1.000	1.001	0.028	0.028	1.005	0.028	0.028	1.001	0.028	0.028
A1&2&3	0.014	1.005	1.050	1.044	0.016	0.016	0.016	1.001	1.001	1.005	0.016	0.016	0.016	1.001	0.016
A0&1&2&3	1.008	1.012	1.082	1.064	1.001	1.002	1.001	1.005	1.004	1.015	1.001	1.001	1.001	1.003	1.001
Injeção de 5kHz															
ACh0	1.000	0.003	0.000	-0.000	0.003	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
ACh1	0.003	0.999	0.050	-0.001	0.004	0.000	-0.000	0.049	-0.000	-0.000	0.000	-0.000	-0.000	-0.000	-0.000
ACh2	0.000	0.000	1.031	0.019	0.000	0.000	0.000	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000
ACh3	0.000	-0.000	0.284	1.030	0.000	0.000	0.000	-0.000	0.000	0.284	0.000	0.000	0.000	-0.000	0.000
A0&1	1.003	1.001	0.115	0.000	1.000	0.115	0.000	0.115	-0.000	0.000	0.115	0.000	0.000	-0.000	-0.000
A0&2	1.003	0.013	1.050	0.037	0.014	1.000	0.037	0.014	0.001	0.037	0.014	0.001	0.037	0.001	0.001
A0&3	1.002	0.007	0.446	1.040	0.006	0.445	1.000	0.004	0.006	0.447	0.004	0.006	0.445	0.004	0.004
A1&2	0.010	1.001	1.046	0.038	0.010	0.010	0.000	1.000	0.038	0.038	0.010	0.001	0.001	0.038	0.001
A1&3	0.008	1.001	0.735	1.041	0.008	0.007	0.008	0.734	1.000	0.738	0.006	0.008	0.006	0.734	0.006
A2&3	-0.000	0.000	1.062	1.057	-0.000	-0.000	0.000	0.000	0.000	1.003	-0.000	-0.000	0.000	0.001	-0.000
A0&1&2	1.004	1.003	1.070	0.078	1.000	1.000	0.078	1.000	0.078	0.079	1.000	0.078	0.078	0.078	0.078
A0&1&3	1.004	1.002	0.861	1.053	1.000	0.861	1.000	0.861	1.000	0.865	0.861	1.000	0.861	0.861	0.861
A0&2&3	1.004	0.029	1.097	1.077	0.029	1.001	1.001	0.030	0.029	1.008	0.029	0.029	1.001	0.029	0.029
A1&2&3	0.015	1.004	1.091	1.074	0.015	0.015	0.015	1.001	1.001	1.006	0.015	0.015	0.015	1.001	0.015
A0&1&2&3	1.005	1.008	1.135	1.097	1.001	1.001	1.001	1.004	1.004	1.014	1.001	1.001	1.001	1.003	1.001

Tabela 4.7: Resultados da injeção de luz em 10kHz e 25kHz - Grupo A

Injeção de 10kHz														
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
ACh0	0.999	0.003	0.000	-0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ACh1	0.004	1.001	0.050	0.000	0.005	0.000	0.000	0.051	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ACh2	0.000	0.000	1.055	0.019	0.000	0.000	0.000	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000
ACh3	-0.000	-0.000	0.281	1.047	0.000	0.000	0.000	0.000	0.000	0.282	0.000	0.000	0.000	0.000	0.000
A0&1	1.003	1.002	0.118	-0.000	1.000	0.118	0.000	0.118	0.000	-0.000	0.118	0.000	-0.000	-0.000	-0.000
A0&2	1.003	0.014	1.087	0.036	0.014	1.000	0.036	0.014	0.001	0.036	0.014	0.001	0.036	0.001	0.001
A0&3	1.003	0.006	0.440	1.065	0.006	0.441	1.000	0.003	0.006	0.443	0.003	0.006	0.441	0.004	0.003
A1&2	0.009	1.002	1.081	0.039	0.009	0.009	0.001	1.000	0.039	0.039	0.009	0.001	0.001	0.039	0.001
A1&3	0.006	1.001	0.733	1.064	0.008	0.006	0.008	0.733	1.000	0.735	0.006	0.008	0.006	0.733	0.006
A2&3	0.000	0.000	1.102	1.088	-0.000	0.000	0.000	0.000	0.000	1.002	-0.000	-0.000	0.000	0.000	-0.000
A0&1&2	1.004	1.003	1.121	0.081	1.000	1.000	0.082	1.000	0.082	0.082	1.000	0.082	0.082	0.082	0.082
A0&1&3	1.004	1.003	0.859	1.085	1.000	0.859	1.000	0.859	1.000	0.863	0.858	1.000	0.859	0.859	0.859
A0&2&3	1.004	0.027	1.153	1.116	0.027	1.001	1.000	0.028	0.028	1.006	0.027	0.027	1.001	0.028	0.027
A1&2&3	0.015	1.004	1.140	1.111	0.015	0.015	0.015	1.001	1.001	1.005	0.015	0.015	0.015	1.001	0.015
A0&1&2&3	1.005	1.007	1.203	1.146	1.000	1.001	1.001	1.003	1.002	1.011	1.001	1.000	1.001	1.002	1.001
Injeção de 25kHz															
ACh0	0.993	0.003	0.000	0.000	0.003	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
ACh1	0.004	1.001	0.050	0.000	0.004	0.000	0.000	0.051	0.000	0.000	0.000	-0.000	-0.000	0.000	-0.000
ACh2	-0.000	0.000	1.087	0.019	0.000	0.000	0.000	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000
ACh3	0.000	-0.000	0.276	1.075	-0.000	0.000	0.000	-0.000	0.000	0.276	0.000	0.000	0.000	0.000	-0.000
A0&1	1.002	1.001	0.117	0.000	0.999	0.117	0.000	0.117	0.000	0.000	0.117	0.000	0.000	0.000	0.000
A0&2	1.002	0.014	1.141	0.036	0.014	1.001	0.037	0.014	0.001	0.037	0.014	0.001	0.037	0.001	0.001
A0&3	1.001	0.006	0.438	1.105	0.006	0.437	1.000	0.003	0.006	0.439	0.003	0.006	0.438	0.003	0.003
A1&2	0.009	1.002	1.132	0.037	0.009	0.009	0.001	1.001	0.037	0.038	0.009	0.001	0.001	0.038	0.001
A1&3	0.007	1.001	0.728	1.103	0.007	0.006	0.007	0.728	1.000	0.730	0.006	0.007	0.006	0.728	0.006
A2&3	0.000	0.000	1.161	1.138	-0.000	0.000	0.000	0.000	0.000	1.003	-0.000	-0.000	0.000	0.000	-0.000
A0&1&2	1.003	1.003	1.201	0.081	1.000	1.001	0.081	1.001	0.081	0.081	1.001	0.081	0.081	0.081	0.081
A0&1&3	1.003	1.002	0.855	1.136	1.000	0.855	1.000	0.855	1.000	0.858	0.855	1.000	0.855	0.855	0.855
A0&2&3	1.003	0.029	1.235	1.178	0.029	1.002	1.001	0.029	0.029	1.006	0.029	0.029	1.002	0.029	0.029
A1&2&3	0.016	1.002	1.221	1.177	0.016	0.017	0.017	1.002	1.001	1.006	0.016	0.016	0.016	1.002	0.016
A0&1&2&3	1.004	1.004	1.305	1.218	1.000	1.002	1.001	1.002	1.001	1.009	1.002	1.001	1.003	1.003	1.003

Tabela 4.8: Resultados da injeção de luz em 50kHz e 100kHz - Grupo A

Injeção de 50kHz															
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
ACh0	0.992	0.003	-0.000	0.000	0.003	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
ACh1	0.005	1.000	0.050	-0.000	0.005	0.000	0.000	0.050	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ACh2	-0.000	0.000	1.076	0.019	-0.000	0.000	-0.000	0.000	-0.000	0.019	-0.000	-0.000	-0.000	-0.000	-0.000
ACh3	-0.000	-0.000	0.271	1.072	-0.000	-0.000	0.000	-0.000	0.000	0.272	-0.000	-0.000	-0.000	-0.000	-0.000
A0&1	0.999	1.000	0.119	0.000	0.999	0.119	0.000	0.119	0.000	-0.000	0.119	0.000	0.000	-0.000	-0.000
A0&2	0.999	0.014	1.122	0.038	0.014	1.000	0.038	0.014	0.001	0.038	0.014	0.001	0.038	0.001	0.001
A0&3	0.998	0.006	0.432	1.101	0.006	0.431	0.999	0.003	0.006	0.433	0.003	0.006	0.431	0.003	0.003
A1&2	0.009	1.001	1.114	0.037	0.009	0.009	0.001	1.002	0.037	0.038	0.009	0.001	0.001	0.037	0.001
A1&3	0.008	1.000	0.725	1.098	0.008	0.006	0.008	0.725	1.001	0.726	0.006	0.008	0.006	0.725	0.006
A2&3	-0.000	0.000	1.139	1.131	-0.000	0.000	0.000	0.000	0.000	1.004	-0.000	-0.000	0.000	0.000	-0.000
A0&1&2	1.001	1.001	1.170	0.083	0.999	1.002	0.083	1.002	0.083	0.083	1.002	0.083	0.083	0.083	0.083
A0&1&3	1.001	1.001	0.854	1.131	0.999	0.854	1.000	0.854	1.001	0.857	0.854	1.000	0.854	0.855	0.854
A0&2&3	1.000	0.029	1.202	1.167	0.029	1.002	1.001	0.030	0.030	1.007	0.030	0.029	1.004	0.030	0.030
A1&2&3	0.016	1.001	1.187	1.164	0.016	0.016	0.016	1.003	1.002	1.007	0.016	0.016	0.016	1.004	0.016
A0&1&2&3	1.001	1.001	1.256	1.196	0.999	1.004	1.002	1.003	1.002	1.010	1.003	1.001	1.004	1.005	1.004
Injeção de 100kHz															
ACh0	0.999	0.003	-0.000	0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	-0.000	-0.000	0.000	-0.000
ACh1	0.004	1.000	0.049	0.000	0.004	0.000	-0.000	0.049	0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000
ACh2	-0.000	0.000	1.026	0.019	0.000	0.000	-0.000	0.000	0.000	0.019	-0.000	-0.000	-0.000	0.000	-0.000
ACh3	0.000	-0.000	0.257	1.031	-0.000	0.000	0.000	0.000	0.000	0.257	0.000	-0.000	0.000	0.000	-0.000
A0&1	1.000	1.000	0.116	-0.000	1.000	0.116	0.000	0.116	0.000	-0.000	0.116	0.000	-0.000	0.000	-0.000
A0&2	0.999	0.014	1.044	0.038	0.014	1.001	0.038	0.014	0.001	0.038	0.014	0.001	0.038	0.001	0.001
A0&3	0.999	0.006	0.418	1.045	0.006	0.418	1.001	0.003	0.006	0.419	0.003	0.006	0.418	0.003	0.003
A1&2	0.009	1.000	1.041	0.038	0.009	0.009	0.001	1.001	0.038	0.038	0.009	0.001	0.001	0.038	0.001
A1&3	0.007	1.000	0.709	1.043	0.008	0.006	0.008	0.709	1.001	0.710	0.006	0.008	0.006	0.710	0.006
A2&3	-0.000	0.000	1.051	1.057	0.000	0.000	0.000	0.000	0.000	1.003	0.000	0.000	0.000	0.000	0.000
A0&1&2	1.000	1.000	1.063	0.084	1.000	1.002	0.084	1.002	0.084	0.084	1.002	0.084	0.084	0.084	0.084
A0&1&3	1.000	1.000	0.843	1.056	1.000	0.843	1.001	0.843	1.001	0.844	0.843	1.001	0.844	0.844	0.844
A0&2&3	1.000	0.031	1.075	1.073	0.031	1.002	1.002	0.031	0.031	1.004	0.031	0.031	1.003	0.031	0.031
A1&2&3	0.017	1.000	1.069	1.068	0.017	0.017	0.017	1.002	1.002	1.004	0.017	0.017	0.017	1.003	0.017
A0&1&2&3	1.000	1.000	1.098	1.062	1.000	1.002	1.001	1.003	1.001	1.005	1.002	1.001	1.004	1.004	1.004

Neste resultado obtido, chama a atenção o valor de *crosstalk* do terceiro canal do grupo A, ACh2. Quando a tripla A0&1&3 é injetada, este canal conta em torno de 86% dos pulsos, apesar de não estar sendo injetado. O mesmo ocorre quando o canal ACh3 é injetado e ele conta en torno de 27% dos pulsos. Também é possível verificar que a taxa de *crosstalk* não é dependente da frequência injetada.

Com esse resultado, é possível avaliar, por exemplo, anomalias que possam ocorrer nos resultados futuros e monitorar, caso haja alterações ao longo do tempo devido à dilatação e contração do sistema, fatores relevantes para o experimento que será instalado no módulo antártico CRIOSFERA 1.

Os resultados completos dos três grupos podem ser encontrados no apêndice A.

4.3 Site do experimento CRE4AT

Para realizar o monitoramento e disponibilização pública dos dados, um site foi desenvolvido. Ele é atualizado diariamente de forma automatizada, com os gráficos gerados na análise. O site pode ser encontrado no link: https://wp.cbpf.br/cre4at/.

O site foi projetado para ser facilmente navegável. Por meio do menu superior é possível encontrar os experimentos existentes ou que estão previstos para entrar em operação. Em cada localização disponível, um submenu oferece as opções de abrir os gráficos diários, acessível por meio da escolha do dia, ou de longa duração. Também existe uma opção com informações do experimento instalado no local selecionado. Na opção IPANEMA, são encontrados os resultados das 3 vertentes do projeto instaladas. Para o monitoramento do CLOUD, for desenvolvido um sistema automatizado para criar um filme diário com as imagens de monitoramento. Para o detector PMT e SiPM (tema do trabalho), os gráficos com os resultados das análises são exibidos.

Figura 4.24: Site público de monitoramento do experimento CRE4AT.

Capítulo 5

Conclusão

O presente trabalho apresentou o desenvolvimento de um detector de partículas carregadas, com sistema de injeção de luz para calibração, para estudar e buscar respostas sobre a possibilidade de o processo de formação de nuvens ser iniciado por meio de ionização causada pelos raios cósmicos galácticos, tema que vem sendo discutido e apresentado em outros trabalhos, mas ainda necessitando de entendimento para confirmação. Esse processo, se confirmado, poderá influenciar na compreensão atual de mudanças climáticas que o planeta vem sofrendo ao longo dos anos, bem como ajudar em previsões climáticas, ajustando os modelos existentes atualmente.

Durante o desenvolvimento do projeto, foi decidido projetar um sistema modular, sendo ele composto por:

1. Grupos de 4 canais de fotomultiplicadores de silício, sendo um por plano, formado pelos cintiladores plásticos, fibra *wavelength shifter* e conjunto óptico;

2. Placas eletrônicas de *front-end*, contendo 4 canais cada, onde o sinal analógico de um grupo é tratado e digitalizado;

3. Uma placa do sistema de aquisição, alimentação e controle, responsável pela lógica executada com os sinais digitalizados dos SiPMs, provenientes das eletrônicas de front-end, assim como aquisição dos parâmetros ambientais pelos sensores presentes, escrita dos dados, comunicação externa, controle do sistema de injeção de luz e distribuição das tensões de alimentação;

4. Uma placa do sistema de injeção de luz, responsável por repetir o sinal de cada canal para todos os grupos presentes;

5. Estrutura mecânica, onde é possível movimentar os grupos em um eixo e, assim, realizar a escolha da aceptância do detector.

A escolha por um sistema modular ocorreu devido à facilidade em alterar as configurações necessárias para cada local no qual será feita a instalação.

5.1 Comissionamento do detector

Durante os testes executados no laboratório do CBPF, o detector foi validado por meio de métodos automatizados para busca dos parâmetros, como valor de limiar e *latch*, assim como as primeiras aquisições de raios cósmicos e injeção de luz por meio do sistema de LIS.

Localmente na Antártica, foi validado o sistema de busca do fator que eleva o cosseno do ângulo zenital no modelo de múons cósmicos e testada a influência de barreiras como uma montanha, no método desenvolvido para essa busca.

5.2 Interpretação dos Resultados obtidos

A medida de fluxo realizada pelos três grupos é compatível dentro de uma pequena variação entre eles. O sistema é influenciado pelas transmissões de radiocomunicação com a Estação Antártica Comandante Ferraz, produzindo aumentos nas contagens e, consequentemente, nos fluxos medidos. Como medida provisória, ao ser utilizado o rádio, o horário será anotado, caso possíveis aumentos nas medidas do fluxo não sejam reconhecidos como *bursts*.

A correção do valor de alta tensão devido à mudança de temperatura está sendo realizado de forma eficiente, uma vez que o resultado de medida de fluxo permaneceu estável durante todo o período, apesar de variações da ordem de 10°C terem ocorrido durante esse mês de aquisição para comissionamento do experimento.

Na medida do fator que eleva o cosseno do ângulo zenital, no modelo de múons cósmicos, o valor encontrado está acima do esperado, devido à geografia local, onde ângulos maiores são reduzidos da distribuição encontrada, por conta da região montanhosa, e, assim, o resultado coincide com a medição de múons menos energéticos. Com relação ao estudo realizado para verificar a influência da montanha próxima ao refúgio, girando o eixo seccionado do detector em direção à montanha, o valor encontrado do fator aumentou, resultado esse compatível com o esperado devido à absorção de parte dos múons pela montanha próxima. Como o fator é medido para avaliar a variação dele a longo prazo, esse valor obtido mais alto do que o esperado não será impeditivo para análise de mudanças atmosféricas que possam vir a acontecer.

5.3 Trabalhos futuros

Após a constatação de que o rádio utilizado para comunicação com a EACF produz ruído no sistema, uma estrutura de metal, funcionando como Gaiola de Faraday poderá ser inserida no sistema, bloqueando a entrada de ruído eletromagnético, que poderá impactar de forma negativa a medida adquirida.

O sistema também já foi preparado para receber uma placa de expansão contendo novos sensores ambientais, como CO^2, ozônio, metano, entre outros. Um estudante do grupo é responsável pelo desenvolvimento dessa PCB expansora.

Após um longo período de aquisição de dados, será necessário verificar a existência de *bursts*, assim como correlacioná-los com as outras vertentes do projeto, seja pelo imageamento do CRE4AT CLOUD, seja pela procura por meio de dados de satélite, objetivando buscar formações de núcleos de condensação e, posteriormente, nuvens, durante eventos abruptos, nos quais ocorre o aumento de chuveiros atmosféricos.

5.4 Publicações

Durante o período de atualização do LHCb, o trabalho de desenvolvimento do sistema de teste das eletrônicas do sistema de traço à base de fibras cintilantes foi realizado. A leitura dessas fibras é feita por meio de fotomultiplicadores de silício. Este trabalho foi de suma importância para obter o conhecimento necessário dessa tecnologia, que também foi utilizada no detector do experimento CRE4AT. Deste trabalho, uma nota técnica do procedimento de teste foi publicada [66] e pode ser encontrada no apêndice S, assim como um artigo sobre a atualização do experimento LHCb[67].

Bibliografia

[1] Alexandre Medeiros. "As Origens Históricas do Eletroscópio". Em: *Revista Brasileira de Ensino de Física* 24.3 (set. de 2002), pp. 353–361. ISSN: 1806-1117. DOI: 10.1590/S0102-47442002000300013. URL: https://doi.org/10.1590/S0102-47442002000300013.

[2] Michael Friedlander. "A century of cosmic rays". Em: *Nature* 483.7390 (mar. de 2012), pp. 400–401. ISSN: 1476-4687. DOI: 10.1038/483400a. URL: https://doi.org/10.1038/483400a.

[3] Victor Hess. *On the Observations of the Penetrating Radiation during Seven Balloon Flights*. 2018. arXiv: 1808.02927 [physics.hist-ph].

[4] Henrik Svensmark e Eigil Friis-Christensen. "Variation of cosmic ray flux and global cloud coverage—a missing link in solar-climate relationships". Em: *Journal of Atmospheric and Solar-Terrestrial Physics* 59.11 (1997), pp. 1225–1232. ISSN: 1364-6826. DOI: https://doi.org/10.1016/S1364-6826(97)00001-1. URL: https://www.sciencedirect.com/science/article/pii/S1364682697000011.

[5] F. Arnold. "Ion nucleation—a potential source for stratospheric aerosols". Em: *Nature* 299.5879 (set. de 1982), pp. 134–137. ISSN: 1476-4687. DOI: 10.1038/299134a0. URL: https://doi.org/10.1038/299134a0.

[6] Fangqun Yu e Richard P. Turco. "Ultrafine aerosol formation via ion-mediated nucleation". Em: *Geophysical Research Letters* 27.6 (2000), pp. 883–886. DOI: https://doi.org/10.1029/1999GL011151. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/1999GL011151. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/1999GL011151.

[7] Richard P. Turco, Jing-Xia Zhao e Fangqun Yu. "A new source of tropospheric aerosols: Ion-ion recombination". Em: *Geophysical Research Letters* 25.5 (1998), pp. 635–638. DOI: https://doi.org/10.1029/98GL00253. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/98GL00253. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/98GL00253.

[8] Programa Antártco Brasileiro. *Programa Antártico Brasileiro (PROANTAR)*. https://www.marinha.mil.br/secirm/pt-br/proantar/sobre. Accessed: 2024-02-15.

[9] Programa Antártco Brasileiro. *Estação Antártica Comandante Ferraz*. https://www.marinha.mil.br/secirm/pt-br/proantar/eacf. Accessed: 2024-02-15.

[10] Programa Antártco Brasileiro. *Projeto CBPS-UERJ reativa o Módulo Ipanema*. https://www.marinha.mil.br/secirm/en/pt-br/proantar/noticias/projeto-cbps-uerj-ipanema. Accessed: 2024-02-15.

[11] CERN. *Cosmic rays: particles from outer space*. https://home.cern/science/physics/cosmic-rays-particles-outer-space. Acessado em: 15-11-2023.

[12] A.C. Fauth et al. "Raios Cósmicos Detectados Através de um Telescópio de Partículas". Em: *Memorias LIADA 2006*. XI° Convención Internacional de Astronomía de la LIADA, 2006. LIADA, Argentina, 2006.

[13] C. M. G. LATTES, G. P. S. OCCHIALINIDR. e C. F. POWELLDR. "Observations on the Tracks of Slow Mesons in Photographic Emulsions*". Em: *Nature* 160.4066 (out. de 1947), pp. 453–456. ISSN: 1476-4687. DOI: 10.1038/160453a0. URL: https://doi.org/10.1038/160453a0.

[14] Claudia Fracchiolla. "Estudo da Resolução Angular do Observatório Pierre Auger". Dissertação (mestrado). Departamento de Física - PUC-RIO, 2007.

[15] S. Cecchini e M. Spurio. "Atmospheric muons: experimental aspects". Em: *Geoscientific Instrumentation, Methods and Data Systems Discussions* 2 (ago. de 2012). DOI: 10.5194/gid-2-603-2012.

[16] A.C. Fauth et al. "Demonstração experimental da dilatação do tempo e da contração do espaço dos múons da radiação cósmica". Em: *Revista Brasileira de Ensino de Física* 29.4 (2007), pp. 585–591. ISSN: 1806-1117. DOI: 10.1590/S1806-11172007000400017. URL: https://doi.org/10.1590/S1806-11172007000400017.

[17] Hiroyuki K. M. Tanaka. "Muometric positioning system (μPS) with cosmic muons as a new underwater and underground positioning technique". Em: *Scientific Reports* 10.1 (nov. de 2020), p. 18896. ISSN: 2045-2322. DOI: 10.1038/s41598-020-75843-7. URL: https://doi.org/10.1038/s41598-020-75843-7.

[18] Daniel Martelozo CONSALTER. "Estudo de raios cósmicos com E>10^{18}eV do detector de superfície do Observatório Pierre Auger". Dissertação (mestrado). Universidade Estadual de Campinas, Instituto de Fisica Gleb Wataghin, 2009.

[19] Luis A. Anchordoqui. "Ultra-high-energy cosmic rays". Em: *Physics Reports* 801 (2019). Ultra-high-energy cosmic rays, pp. 1–93. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2019.01.002. URL: https://www.sciencedirect.com/science/article/pii/S037015731930002X.

[20] Dariusz Góra e For the Pierre Auger Collaboration. "The Pierre Auger Observatory: Review of Latest Results and Perspectives". Em: *Universe* 4.11 (2018). ISSN: 2218-1997. DOI: 10.3390/universe4110128. URL: https://www.mdpi.com/2218-1997/4/11/128.

[21] Francis Halzen e Ali Kheirandish. *IceCube and High-Energy Cosmic Neutrinos*. 2022. arXiv: 2202.00694 [astro-ph.HE].

[22] M. Aguilar et al. "Electron and Positron Fluxes in Primary Cosmic Rays Measured with the Alpha Magnetic Spectrometer on the International Space Station". Em: *Phys. Rev. Lett.* 113 (12 set. de 2014), p. 121102. DOI: 10.1103/PhysRevLett.113.121102. URL: https://link.aps.org/doi/10.1103/PhysRevLett.113.121102.

[23] Hiroko Miyahara, Yusuke Yokoyama e Yasuhiko Yamaguchi. "Influence of the Schwabe/Hale solar cycles on climate change during the Maunder Minimum". Em: *Proceedings of the International Astronomical Union* 5 (fev. de 2010), pp. 427–433. DOI: 10.1017/S1743921309993048.

[24] Guerrero, G. e de Gouveia Dal Pino, E. M. "Turbulent magnetic pumping in a Babcock-Leighton solar dynamo model". Em: *A&A* 485.1 (2008), pp. 267–273. DOI: 10.1051/0004-6361:200809351. URL: https://doi.org/10.1051/0004-6361:200809351.

[25] Joseph Larmor. "17. How Could a Rotating Body such as the Sun Become a Magnet?" Em: *A Source Book in Astronomy and Astrophysics, 1900–1975*. Ed. por Kenneth R. Lang e Owen Gingerich. Cambridge, MA e London, England: Harvard University Press, 1979, pp. 106–107. ISBN: 9780674366688. DOI: doi:10.4159/harvard.9780674366688.c20. URL: https://doi.org/10.4159/harvard.9780674366688.c20.

[26] David H. Hathaway. "The Solar Cycle". Em: *Living Reviews in Solar Physics* 12.1 (set. de 2015), p. 4. ISSN: 1614-4961. DOI: 10.1007/lrsp-2015-4. URL: https://doi.org/10.1007/lrsp-2015-4.

[27] J. M. T. Thompson e S. M. Tobias. "The solar dynamo". Em: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 360.1801 (2002), pp. 2741–2756. DOI: 10.1098/rsta.2002.1090. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2002.1090. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2002.1090.

[28] Ilya G. Usoskin. "A History of Solar Activity over Millennia". Em: *Living Reviews in Solar Physics* 10.1 (mar. de 2013), p. 1. ISSN: 1614-4961. DOI: 10.12942/lrsp-2013-1. URL: https://doi.org/10.12942/lrsp-2013-1.

[29] W. Baumjohann e R. Nakamura. "Magnetospheric Contributions to the Terrestrial Magnetic Field". Em: *Geomagnetism*. Ed. por Gerald Schubert. Vol. 5. 2007, pp. 77–92. DOI: 10.1016/B978-044452748-6.00088-2.

[30] M. Lockwood, R. Stamper e M. N. Wild. "A doubling of the Sun's coronal magnetic field during the past 100 years". Em: *Nature* 399.6735 (jun. de 1999), pp. 437–439. ISSN: 1476-4687. DOI: 10.1038/20867. URL: https://doi.org/10.1038/20867.

[31] Climate4you. *Climate4you*. http://www.climate4you.com/Sun.htm. Acesado em: 10-03-2024.

[32] E. Echer et al. "O número de manchas solares, índice da atividade do sol". Em: *Revista Brasileira de Ensino de Física* 25 (jun. de 2003). DOI: 10.1590/S0102-47442003000200004.

[33] Henrique Vieira de Souza. "Estudo da atividade solar com detectores de partículas situados em solo terrestre". Dissertação (mestrado). Universidade Estadual de Campinas, Instituto de Fisica Gleb Wataghin, 2017. DOI: https://doi.org/10.47749/T/UNICAMP.2017.988877.

[34] Keitarou Matsumoto et al. "Relationship of peak fluxes of solar radio bursts and X-ray class of solar flares: Application to early great solar flares". Em: *Publications of the Astronomical Society of Japan* 75.6 (set. de 2023), pp. 1095–1104. ISSN: 2053-051X. DOI: 10.1093/pasj/psad058. eprint: https://academic.oup.com/pasj/article-pdf/75/6/1095/54151707/psad058.pdf. URL: https://doi.org/10.1093/pasj/psad058.

[35] William Herschel. "IX. On the method of observing the changes that happen to the fixed stars; with some remarks on the stability of the light of our sun. To which is added, a catalogue of comparative brightness, for ascertaining the permanency of the lustre of stars". Em: *Philosophical Transactions of the Royal Society of London* 86 (1796), pp. 166–226. DOI: 10.1098/rstl.1796.0010. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rstl.1796.0010. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rstl.1796.0010.

[36] Charles Philip Sonett, Mark S Giampapa e Mildred Shapley Matthews. *The sun in time*. University of Arizona Press, 1991.

[37] C. Froehlich et al. "Solar irradiance variability from modern measurements". Em: (fev. de 1991).

[38] E. Friis-Christensen e K. Lassen. "Length of the Solar Cycle: An Indicator of Solar Activity Closely Associated with Climate". Em: *Science* 254.5032 (1991), pp. 698–700. DOI: 10.1126/science.254.5032.698. eprint: https://www.science.org/doi/pdf/10.1126/science.254.5032.698. URL: https://www.science.org/doi/abs/10.1126/science.254.5032.698.

[39] Joanna D. Haigh. "The Sun and the Earth's Climate". Em: *Living Reviews in Solar Physics* 4.1 (out. de 2007), p. 2. ISSN: 1614-4961. DOI: 10.12942/lrsp-2007-2. URL: https://doi.org/10.12942/lrsp-2007-2.

[40] José Pinto Peixoto et al. "Entropy budget of the atmosphere". Em: *Journal of Geophysical Research: Atmospheres* 96.D6 (1991), pp. 10981–10988. DOI: https://doi.org/10.1029/91JD00721. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/91JD00721. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/91JD00721.

[41] Kamalika Chatterjee, Rahul Kashyap e Jaywant Arakeri. "Experimental Study of Cloud Formation". Em: *Applied Mechanics and Materials* 110-116 (out. de 2011). DOI: 10.4028/www.scientific.net/AMM.110-116.2570.

[42] Phys.org Bob Yirka. *CERN CLOUD research team adds new pieces to puzzle of cloud formation*. https://phys.org/news/2011-08-cern-cloud-team-pieces-puzzle.html. Acessado em: 10-03-2024.

[43] S.W. Moser et al. "Principles and practice of plastic scintillator design". Em: *Radiation Physics and Chemistry* 41.1 (1993), pp. 31–36. ISSN: 0969-806X. DOI: https://doi.org/10.1016/0969-806X(93)90039-W. URL: https://www.sciencedirect.com/science/article/pii/0969806X9390039W.

[44] Claus Grupen e Boris Shwartz. "Interactions of particles and radiation with matter". Em: *Particle Detectors*. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology. Cambridge University Press, 2008, pp. 1–55.

[45] J.B. BIRKS. "CHAPTER 3 - THE SCINTILLATION PROCESS IN ORGANIC MATERIALS—I". Em: *The Theory and Practice of Scintillation Counting*. Ed. por J.B. BIRKS. International Series of Monographs in Electronics and Instrumentation. Pergamon, 1964, pp. 39–67. ISBN: 978-0-08-010472-0. DOI: https://doi.org/10.1016/B978-0-08-010472-0.50008-2. URL: https://www.sciencedirect.com/science/article/pii/B9780080104720500082.

[46] W.R. Leo. *Techniques for Nuclear and Particle Physics Experiments: A How-to Approach*. Springer, 1994. ISBN: 9780387173863. URL: https://books.google.ch/books?id=W7vHQgAACAAJ.

[47] U. Carneiro. "DESENVOLVIMENTO DO EXPERIMENTO ANTÁRTICO DE MONITORAÇÃO DE RAIOS CÓSMICOS PARA O MÓDULO CRIOSFERA I". Tese de dout. CEFET/RJ, mai. de 2015, p. 243.

[48] Zizhao Zong et al. "Study of light yield for different configurations of plastic scintillators and wavelength shifting fibers". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 908 (2018), pp. 82–90. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2018.08.029. URL: https://www.sciencedirect.com/science/article/pii/S0168900218309823.

[49] J. Silva et al. "Ageing studies of wavelength shifter fibers for the TILECAL/ATLAS experiment". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 580.1 (2007). Proceedings of the 10 th International Symposium on Radiation Physics, pp. 318–321. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2007.05.165. URL: https://www.sciencedirect.com/science/article/pii/S016890020701039X.

[50] J.C. Jackson et al. "A novel silicon Geiger-mode avalanche photodiode". Em: *Digest. International Electron Devices Meeting,* 2002, pp. 797–800. DOI: 10.1109/IEDM.2002.1175958.

[51] Nicola D'Ascenzo et al. "Analysis of photon statistics with Silicon Photomultiplier". Em: *Journal of Instrumentation* 10 (ago. de 2015). DOI: 10.1088/1748-0221/10/08/C08017.

[52] D. Renker. "Geiger-mode avalanche photodiodes, history, properties and problems". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 567.1 (2006). Proceedings of the 4th International Conference on New Developments in Photodetection, pp. 48–56. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2006.05.060. URL: https://www.sciencedirect.com/science/article/pii/S0168900206008680.

[53] P. Buzhan et al. "Timing by silicon photomultiplier: A possible application for TOF measurements". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 567.1 (2006). Proceedings of the 4th International Conference on New Developments in Photodetection, pp. 353–355. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2006.05.142. URL: https://www.sciencedirect.com/science/article/pii/S0168900206009752.

[54] M. McClish et al. "Recent advances of planar silicon APD technology". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 567.1 (2006). Proceedings of the 4th International Conference on New Developments in Photodetection, pp. 36–40. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2006.05.055. URL: https://www.sciencedirect.com/science/article/pii/S0168900206008655.

[55] B. Dolgoshein et al. "Status report on silicon photomultiplier development and its applications". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 563.2 (2006). TRDs for the Third Millenium, pp. 368–376. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2006.02.193. URL: https://www.sciencedirect.com/science/article/pii/S0168900206004578.

[56] A. Vacheret et al. "Characterization and simulation of the response of Multi-Pixel Photon Counters to low light levels". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 656.1 (2011), pp. 69–83. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2011.07.022. URL: https://www.sciencedirect.com/science/article/pii/S0168900211014513.

[57] M. Danilov. "Novel photo-detectors and photo-detector systems". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 604.1 (2009). PSD8, pp. 183–189. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2009.01.208. URL: https://www.sciencedirect.com/science/article/pii/S016890020900148X.

[58] P. Buzhan et al. "Silicon photomultiplier and its possible applications". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 504.1 (2003). Proceedings of the 3rd International Conference on New Developments in Photodetection, pp. 48–52. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)00749-6. URL: https://www.sciencedirect.com/science/article/pii/S0168900203007496.

[59] Elisabetta Nocerino. "The Semiconductor Multiplication System for Photoelectrons in a Vacuum Silicon Photomultiplier Tube and Related Front End Electronics". Tese de dout. Universitá degli Studi di Napoli "Federico II", nov. de 2016, p. 157.

[60] Peter K.F. Grieder. "Chapter 1 - Cosmic Ray Properties, Relations and Definitions". Em: *Cosmic Rays at Earth*. Ed. por Peter K.F. Grieder. Amsterdam: Elsevier, 2001, pp. 1–53. ISBN: 978-0-444-50710-5. DOI: https://doi.org/10.1016/B978-044450710-5/50003-8. URL: https://www.sciencedirect.com/science/article/pii/B9780444507105500038.

[61] Bryan Olmos Yáñez e Alexis A. Aguilar-Arevalo. "A method to measure the integral vertical intensity and angular distribution of atmospheric muons with a stationary plastic scintillator bar detector". Em: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 987 (2021), p. 164870. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2020.164870. URL: https://www.sciencedirect.com/science/article/pii/S0168900220312675.

[62] Ning Su et al. "A Comparison of Muon Flux Models at Sea Level for Muon Imaging and Low Background Experiments". Em: *Frontiers in Energy Research* 9 (2021). ISSN: 2296-598X. DOI: 10.3389/fenrg.2021.750159. URL: https://www.frontiersin.org/articles/10.3389/fenrg.2021.750159.

[63] Cláudio Mendes junior et al. "A new topographic map for Keller Peninsula, King George Island, Antarctica". Em: *PEsquisa Antártica Brasileira* 5 (mar. de 2012), pp. 105–113. DOI: 10.31789/pab.v5n1.010.

[64] Cláudio Wilson Mendes Júnior et al. "ANÁLISE MORFOMÉTRICA DA PENÍNSULA KELLER, ANTÁRTICA, ATRAVÉS DO SIG". Em: *Revista Brasileira de Cartografia* 62.4 (fev. de 2011). DOI: 10.14393/rbcv62n4-43692. URL: https://seer.ufu.br/index.php/revistabrasileiracartografia/article/view/43692.

[65] S. Cecchini e M. Spurio. "Atmospheric muons: experimental aspects". Em: *Geoscientific Instrumentation, Methods and Data Systems Discussions* 2 (ago. de 2012). DOI: 10.5194/gid-2-603-2012.

[66] Andre Massafferri Rodrigues et al. *Read-out-box Test System Protocol for SciFi Detector*. Rel. técn. Geneva: CERN, 2024. URL: https://cds.cern.ch/record/2889300.

[67] LHCb collaboration et al. *The LHCb upgrade I*. 2023. arXiv: 2305.10515 [hep-ex].

APÊNDICES

Apêndice A

Resultado do Teste de Injeção de Luz

Neste apêndice são apresentadas as tabelas com os resultados do teste de injeção de luz, realizado no comissionamento do detector no CBPF, antes de ser enviado para o refúgio Ipanema (Antártica). O resultado é apresentado em um conjunto de 6 tabelas por grupo, sendo duas por página, para injeção de luz nas frequências de 1kHz, 5kHz, 10kHz, 25kHz, 50kHz e 100kHz.

Tabela A.1: Resultados da injeção de luz em 1kHz e 5kHz - Grupo A

Injeção de 1kHz															
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
ACh0	1.001	0.002	0.002	0.004	0.003	0.000	-0.000	-0.000	-0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000
ACh1	0.005	1.000	0.043	0.002	0.005	-0.001	-0.000	0.046	-0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000
ACh2	-0.000	0.001	1.012	0.022	-0.001	0.000	0.000	0.001	0.000	0.022	0.000	-0.000	0.000	0.000	0.000
ACh3	0.004	0.003	0.278	1.020	0.001	0.002	0.001	0.000	0.001	0.278	0.001	0.001	0.001	0.001	0.001
A0&1	0.999	1.003	0.124	-0.000	0.999	0.120	-0.001	0.120	-0.002	-0.001	0.120	-0.001	-0.001	-0.002	-0.001
A0&2	1.002	0.016	1.026	0.036	0.013	0.998	0.035	0.014	-0.000	0.037	0.014	-0.001	0.035	-0.000	-0.001
A0&3	0.998	0.002	0.440	1.027	0.004	0.440	0.998	-0.000	0.003	0.442	-0.000	0.003	0.441	0.000	0.000
A1&2	0.012	1.002	1.016	0.035	0.010	0.010	0.001	1.001	0.035	0.035	0.010	0.001	0.001	0.035	0.001
A1&3	0.009	0.998	0.735	1.019	0.008	0.006	0.007	0.731	1.000	0.733	0.006	0.008	0.005	0.730	0.005
A2&3	0.003	0.005	1.038	1.035	0.000	0.001	0.001	0.003	0.002	1.005	0.000	0.001	0.001	0.002	0.001
A0&1&2	1.003	1.000	1.037	0.077	0.999	1.000	0.078	1.000	0.079	0.079	1.000	0.078	0.078	0.079	0.078
A0&1&3	1.004	1.000	0.862	1.022	1.000	0.860	1.000	0.860	1.000	0.862	0.860	1.000	0.860	0.860	0.860
A0&2&3	1.004	0.027	1.047	1.047	0.028	1.000	1.001	0.028	0.028	1.005	0.028	0.028	1.001	0.028	0.028
A1&2&3	0.014	1.005	1.050	1.044	0.016	0.016	0.016	1.001	1.001	1.005	0.016	0.016	0.016	1.001	0.016
A0&1&2&3	1.008	1.012	1.082	1.064	1.001	1.002	1.001	1.005	1.004	1.015	1.001	1.001	1.001	1.003	1.001
Injeção de 5kHz															
ACh0	1.000	0.003	0.000	-0.000	0.003	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
ACh1	0.003	0.999	0.050	-0.001	0.004	0.000	-0.000	0.049	-0.000	-0.000	0.000	-0.000	-0.000	-0.000	-0.000
ACh2	0.000	0.000	1.031	0.019	0.000	0.000	0.000	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000
ACh3	0.000	-0.000	0.284	1.030	0.000	0.000	0.000	-0.000	0.000	0.284	0.000	0.000	0.000	-0.000	0.000
A0&1	1.003	1.001	0.115	0.000	1.000	0.115	0.000	0.115	-0.000	0.000	0.115	0.000	0.000	-0.000	-0.000
A0&2	1.003	0.013	1.050	0.037	0.014	1.000	0.037	0.014	0.001	0.037	0.014	0.001	0.037	0.001	0.001
A0&3	1.002	0.007	0.446	1.040	0.006	0.445	1.000	0.004	0.006	0.447	0.004	0.006	0.445	0.004	0.004
A1&2	0.010	1.001	1.046	0.038	0.010	0.010	0.000	1.000	0.038	0.038	0.010	0.001	0.001	0.038	0.001
A1&3	0.008	1.001	0.735	1.041	0.008	0.007	0.008	0.734	1.000	0.738	0.006	0.008	0.006	0.734	0.006
A2&3	-0.000	0.000	1.062	1.057	-0.000	-0.000	0.000	0.000	0.000	1.003	-0.000	-0.000	0.000	0.001	-0.000
A0&1&2	1.004	1.003	1.070	0.078	1.000	1.000	0.078	1.000	0.078	0.079	1.000	0.078	0.078	0.078	0.078
A0&1&3	1.004	1.002	0.861	1.053	1.000	0.861	1.000	0.861	1.000	0.865	0.861	1.000	0.861	0.861	0.861
A0&2&3	1.004	0.029	1.097	1.077	0.029	1.001	1.001	0.030	0.029	1.008	0.029	0.029	1.001	0.029	0.029
A1&2&3	0.015	1.004	1.091	1.074	0.015	0.015	0.015	1.001	1.001	1.006	0.015	0.015	0.015	1.001	0.015
A0&1&2&3	1.005	1.008	1.135	1.097	1.001	1.001	1.001	1.004	1.004	1.014	1.001	1.001	1.001	1.003	1.001

Tabela A.2: Resultados da injeção de luz em 10kHz e 25kHz - Grupo A

Injeção de 10kHz														
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
ACh0	0.999	0.003	0.000	-0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ACh1	0.004	1.001	0.050	0.000	0.005	0.000	0.000	0.051	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ACh2	0.000	0.000	1.055	0.019	0.000	0.000	0.000	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000
ACh3	-0.000	-0.000	0.281	1.047	0.000	0.000	0.000	0.000	0.000	0.282	0.000	0.000	0.000	0.000	0.000
A0&1	1.003	1.002	0.118	-0.000	1.000	0.118	0.000	0.118	0.000	-0.000	0.118	0.000	-0.000	-0.000	-0.000
A0&2	1.003	0.014	1.087	0.036	0.014	1.000	0.036	0.014	0.001	0.036	0.014	0.001	0.036	0.001	0.001
A0&3	1.003	0.006	0.440	1.065	0.006	0.441	1.000	0.003	0.006	0.443	0.003	0.006	0.441	0.004	0.003
A1&2	0.009	1.002	1.081	0.039	0.009	0.009	0.001	1.000	0.039	0.039	0.009	0.001	0.001	0.039	0.001
A1&3	0.006	1.001	0.733	1.064	0.008	0.006	0.008	0.733	1.000	0.735	0.006	0.008	0.006	0.733	0.006
A2&3	0.000	0.000	1.102	1.088	-0.000	0.000	0.000	0.000	0.000	1.002	-0.000	-0.000	0.000	0.000	-0.000
A0&1&2	1.004	1.003	1.121	0.081	1.000	1.000	0.082	1.000	0.082	0.082	1.000	0.082	0.082	0.082	0.082
A0&1&3	1.004	1.003	0.859	1.085	1.000	0.859	1.000	0.859	1.000	0.863	0.858	1.000	0.859	0.859	0.859
A0&2&3	1.004	0.027	1.153	1.116	0.027	1.001	1.000	0.028	0.028	1.006	0.027	0.027	1.001	0.028	0.027
A1&2&3	0.015	1.004	1.140	1.111	0.015	0.015	0.015	1.001	1.001	1.005	0.015	0.015	0.015	1.001	0.015
A0&1&2&3	1.005	1.007	1.203	1.146	1.000	1.001	1.001	1.003	1.002	1.011	1.001	1.000	1.001	1.002	1.001
Injeção de 25kHz															
ACh0	0.993	0.003	0.000	0.000	0.003	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
ACh1	0.004	1.001	0.050	0.000	0.004	0.000	0.000	0.051	0.000	0.000	0.000	-0.000	-0.000	0.000	-0.000
ACh2	-0.000	0.000	1.087	0.019	0.000	0.000	0.000	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000
ACh3	0.000	-0.000	0.276	1.075	-0.000	0.000	0.000	-0.000	0.000	0.276	0.000	0.000	0.000	0.000	-0.000
A0&1	1.002	1.001	0.117	0.000	0.999	0.117	0.000	0.117	0.000	0.000	0.117	0.000	0.000	0.000	0.000
A0&2	1.002	0.014	1.141	0.036	0.014	1.001	0.037	0.014	0.001	0.037	0.014	0.001	0.037	0.001	0.001
A0&3	1.001	0.006	0.438	1.105	0.006	0.437	1.000	0.003	0.006	0.439	0.003	0.006	0.438	0.003	0.003
A1&2	0.009	1.002	1.132	0.037	0.009	0.009	0.001	1.001	0.037	0.038	0.009	0.001	0.001	0.038	0.001
A1&3	0.007	1.001	0.728	1.103	0.007	0.006	0.007	0.728	1.000	0.730	0.006	0.007	0.006	0.728	0.006
A2&3	0.000	0.000	1.161	1.138	-0.000	0.000	0.000	0.000	0.000	1.003	-0.000	-0.000	0.000	0.000	-0.000
A0&1&2	1.003	1.003	1.201	0.081	1.000	1.001	0.081	1.001	0.081	0.081	1.001	0.081	0.081	0.081	0.081
A0&1&3	1.003	1.002	0.855	1.136	1.000	0.855	1.000	0.855	1.000	0.858	0.855	1.000	0.855	0.855	0.855
A0&2&3	1.003	0.029	1.235	1.178	0.029	1.002	1.001	0.029	0.029	1.006	0.029	0.029	1.002	0.029	0.029
A1&2&3	0.016	1.002	1.221	1.177	0.016	0.017	0.017	1.002	1.001	1.006	0.016	0.016	0.016	1.002	0.016
A0&1&2&3	1.004	1.004	1.305	1.218	1.000	1.002	1.001	1.002	1.001	1.009	1.002	1.001	1.003	1.003	1.003

Tabela A.3: Resultados da injeção de luz em 50kHz e 100kHz - Grupo A

Injeção de 50kHz														
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
ACh0	0.992	0.003	-0.000	0.000	0.003	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
ACh1	0.005	1.000	0.050	-0.000	0.005	0.000	0.000	0.050	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ACh2	-0.000	0.000	1.076	0.019	-0.000	0.000	-0.000	0.000	-0.000	0.019	-0.000	-0.000	-0.000	-0.000	-0.000
ACh3	-0.000	-0.000	0.271	1.072	-0.000	-0.000	0.000	-0.000	0.000	0.272	-0.000	-0.000	-0.000	-0.000	-0.000
A0&1	0.999	1.000	0.119	0.000	0.999	0.119	0.000	0.119	0.000	-0.000	0.119	0.000	0.000	-0.000	-0.000
A0&2	0.999	0.014	1.122	0.038	0.014	1.000	0.038	0.014	0.001	0.038	0.014	0.001	0.038	0.001	0.001
A0&3	0.998	0.006	0.432	1.101	0.006	0.431	0.999	0.003	0.006	0.433	0.003	0.006	0.431	0.003	0.003
A1&2	0.009	1.001	1.114	0.037	0.009	0.009	0.001	1.002	0.037	0.038	0.009	0.001	0.001	0.037	0.001
A1&3	0.008	1.000	0.725	1.098	0.008	0.006	0.008	0.725	1.001	0.726	0.006	0.008	0.006	0.725	0.006
A2&3	-0.000	0.000	1.139	1.131	-0.000	0.000	0.000	0.000	0.000	1.004	-0.000	-0.000	0.000	0.000	-0.000
A0&1&2	1.001	1.001	1.170	0.083	0.999	1.002	0.083	1.002	0.083	0.083	1.002	0.083	0.083	0.083	0.083
A0&1&3	1.001	1.001	0.854	1.131	0.999	0.854	1.000	0.854	1.001	0.857	0.854	1.000	0.854	0.855	0.854
A0&2&3	1.000	0.029	1.202	1.167	0.029	1.002	1.001	0.030	0.030	1.007	0.030	0.029	1.004	0.030	0.030
A1&2&3	0.016	1.001	1.187	1.164	0.016	0.016	0.016	1.003	1.002	1.007	0.016	0.016	0.016	1.004	0.016
A0&1&2&3	1.001	1.001	1.256	1.196	0.999	1.004	1.002	1.003	1.002	1.010	1.003	1.001	1.004	1.005	1.004
Injeção de 100kHz															
ACh0	0.999	0.003	-0.000	0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	-0.000	-0.000	0.000	-0.000
ACh1	0.004	1.000	0.049	0.000	0.004	0.000	-0.000	0.049	0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000
ACh2	-0.000	0.000	1.026	0.019	0.000	0.000	-0.000	0.000	0.000	0.019	-0.000	-0.000	-0.000	0.000	-0.000
ACh3	0.000	-0.000	0.257	1.031	-0.000	0.000	0.000	0.000	0.000	0.257	0.000	-0.000	0.000	0.000	-0.000
A0&1	1.000	1.000	0.116	-0.000	1.000	0.116	0.000	0.116	0.000	-0.000	0.116	0.000	-0.000	0.000	-0.000
A0&2	0.999	0.014	1.044	0.038	0.014	1.001	0.038	0.014	0.001	0.038	0.014	0.001	0.038	0.001	0.001
A0&3	0.999	0.006	0.418	1.045	0.006	0.418	1.001	0.003	0.006	0.419	0.003	0.006	0.418	0.003	0.003
A1&2	0.009	1.000	1.041	0.038	0.009	0.009	0.001	1.001	0.038	0.038	0.009	0.001	0.001	0.038	0.001
A1&3	0.007	1.000	0.709	1.043	0.008	0.006	0.008	0.709	1.001	0.710	0.006	0.008	0.006	0.710	0.006
A2&3	-0.000	0.000	1.051	1.057	0.000	0.000	0.000	0.000	0.000	1.003	0.000	0.000	0.000	0.000	0.000
A0&1&2	1.000	1.000	1.063	0.084	1.000	1.002	0.084	1.002	0.084	0.084	1.002	0.084	0.084	0.084	0.084
A0&1&3	1.000	1.000	0.843	1.056	1.000	0.843	1.001	0.843	1.001	0.844	0.843	1.001	0.844	0.844	0.844
A0&2&3	1.000	0.031	1.075	1.073	0.031	1.002	1.002	0.031	0.031	1.004	0.031	0.031	1.003	0.031	0.031
A1&2&3	0.017	1.000	1.069	1.068	0.017	0.017	0.017	1.002	1.002	1.004	0.017	0.017	0.017	1.003	0.017
A0&1&2&3	1.000	1.000	1.098	1.062	1.000	1.002	1.001	1.003	1.001	1.005	1.002	1.001	1.004	1.004	1.004

Tabela A.4: Resultados da injeção de luz em 1kHz e 5kHz - Grupo B

Injeção de 1kHz														
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
BCh0	1.007	0.002	0.001	0.006	0.007	0.001	0.002	0.001	0.001	0.002	0.001	0.002	0.001	0.001	0.001
BCh1	0.010	0.997	0.003	0.003	0.003	0.001	0.002	0.002	0.002	0.002	0.001	0.002	0.002	0.001	0.001
BCh2	0.003	0.021	1.010	0.001	-0.001	0.000	-0.001	0.017	-0.000	0.004	-0.001	-0.001	-0.001	-0.000	-0.001
BCh3	0.003	0.004	0.002	1.003	0.002	0.001	0.002	0.001	0.001	0.002	0.001	0.001	0.001	0.001	0.001
B0&1	1.002	1.001	-0.000	0.003	0.999	-0.000	0.004	0.000	0.004	0.000	0.000	0.005	0.000	-0.000	0.000
B0&2	1.009	0.149	1.018	0.036	0.151	1.001	0.032	0.151	0.006	0.032	0.151	0.005	0.031	0.005	0.005
B0&3	1.004	0.005	-0.002	1.000	0.006	0.001	1.000	-0.001	0.006	0.001	-0.001	0.006	0.002	-0.000	-0.001
B1&2	0.008	1.009	1.013	0.018	0.010	0.010	0.002	1.000	0.019	0.019	0.011	0.001	0.001	0.019	0.001
B1&3	-0.003	1.000	0.006	1.002	0.002	-0.002	0.003	0.004	1.000	0.004	-0.001	0.003	-0.001	0.005	-0.001
B2&3	-0.001	0.038	1.015	1.004	-0.001	-0.000	0.000	0.043	0.044	1.000	-0.000	-0.000	-0.000	0.043	-0.000
B0&1&2	1.009	0.995	1.018	0.084	0.998	0.999	0.077	0.999	0.077	0.078	0.999	0.077	0.077	0.077	0.077
B0&1&3	1.006	1.000	0.011	1.003	0.999	0.007	0.999	0.007	0.999	0.008	0.007	0.999	0.007	0.007	0.007
B0&2&3	1.014	0.269	1.022	1.004	0.266	1.001	1.000	0.266	0.266	1.001	0.266	0.266	1.000	0.266	0.266
B1&2&3	0.017	1.008	1.022	1.004	0.016	0.018	0.017	1.001	1.000	1.001	0.017	0.017	0.017	1.000	0.017
B0&1&2&3	1.016	1.006	1.020	0.999	1.001	1.000	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Injeção de 5kHz															
BCh0	1.004	0.003	0.001	0.000	0.004	-0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
BCh1	0.002	1.000	-0.000	-0.000	0.002	-0.000	-0.000	0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000
BCh2	0.000	0.016	1.017	0.005	-0.000	-0.000	-0.000	0.016	0.000	0.005	-0.000	-0.000	-0.000	0.000	-0.000
BCh3	0.000	0.000	0.001	0.999	-0.000	-0.000	0.000	-0.000	0.000	0.001	-0.000	-0.000	-0.000	-0.000	-0.000
B0&1	1.009	1.001	-0.000	0.004	1.000	0.001	0.005	0.001	0.005	0.000	0.001	0.005	0.000	0.000	0.000
B0&2	1.008	0.146	1.024	0.029	0.147	1.000	0.029	0.147	0.005	0.029	0.147	0.005	0.029	0.005	0.005
B0&3	1.007	0.007	0.003	1.000	0.007	0.002	1.000	0.000	0.007	0.003	0.000	0.007	0.003	0.000	0.000
B1&2	0.011	1.001	1.022	0.019	0.010	0.011	0.000	1.000	0.019	0.019	0.010	0.000	0.000	0.019	0.000
B1&3	0.006	1.001	0.005	0.999	0.005	0.000	0.005	0.005	1.000	0.005	0.000	0.005	0.000	0.005	0.000
B2&3	0.001	0.043	1.024	0.999	0.000	0.000	0.000	0.043	0.043	1.000	0.000	0.000	0.000	0.043	0.000
B0&1&2	1.012	1.001	1.030	0.071	1.000	1.000	0.071	1.000	0.070	0.070	1.000	0.070	0.070	0.070	0.070
B0&1&3	1.010	1.000	0.011	1.000	1.000	0.011	1.000	0.010	1.000	0.011	0.010	1.000	0.010	0.010	0.010
B0&2&3	1.010	0.258	1.032	1.000	0.259	1.000	1.000	0.259	0.259	1.000	0.259	0.259	1.000	0.259	0.259
B1&2&3	0.020	1.001	1.031	1.001	0.019	0.019	0.019	1.000	1.000	1.001	0.019	0.020	0.019	1.000	0.019
B0&1&2&3	1.018	1.000	1.039	1.000	1.000	1.001	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Tabela A.5: Resultados da injeção de luz em 10kHz e 25kHz - Grupo B

Injeção de 10kHz															
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
BCh0	1.005	0.004	0.000	0.001	0.004	-0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
BCh1	0.001	1.000	0.000	0.000	0.002	-0.000	-0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	0.000	-0.000
BCh2	-0.000	0.015	1.020	0.005	-0.000	-0.000	-0.000	0.016	0.000	0.005	-0.000	-0.000	-0.000	0.000	-0.000
BCh3	0.000	-0.000	0.001	0.999	-0.000	0.000	0.000	-0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000
B0&1	1.010	1.000	0.000	0.005	1.000	0.000	0.005	0.000	0.004	-0.000	0.000	0.004	-0.000	-0.000	-0.000
B0&2	1.010	0.146	1.029	0.029	0.146	1.000	0.029	0.146	0.004	0.029	0.146	0.004	0.029	0.004	0.004
B0&3	1.009	0.007	0.003	1.000	0.007	0.003	1.000	-0.000	0.007	0.003	0.000	0.007	0.003	-0.000	-0.000
B1&2	0.010	1.000	1.028	0.019	0.010	0.010	0.000	1.000	0.020	0.020	0.010	0.000	0.000	0.020	0.000
B1&3	0.005	1.000	0.005	0.999	0.005	-0.000	0.005	0.005	1.000	0.005	-0.000	0.005	-0.000	0.005	-0.000
B2&3	0.001	0.042	1.031	0.999	-0.000	0.000	0.000	0.043	0.042	1.000	-0.000	-0.000	0.000	0.042	-0.000
B0&1&2	1.016	1.000	1.041	0.069	1.000	1.000	0.069	1.000	0.069	0.069	1.000	0.069	0.069	0.069	0.069
B0&1&3	1.014	1.000	0.011	0.999	1.000	0.010	1.000	0.010	1.000	0.010	0.010	1.000	0.010	0.010	0.010
B0&2&3	1.013	0.258	1.044	1.000	0.258	1.000	1.000	0.259	0.258	1.000	0.258	0.258	1.000	0.258	0.258
B1&2&3	0.019	1.000	1.040	1.000	0.019	0.019	0.019	1.000	1.000	1.000	0.019	0.019	0.019	1.000	0.019
B0&1&2&3	1.020	1.000	1.054	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Injeção de 25kHz															
BCh0	1.006	0.004	-0.000	0.000	0.004	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
BCh1	0.001	0.999	0.000	-0.000	0.002	-0.000	-0.000	0.000	0.000	-0.000	-0.000	0.000	-0.000	-0.000	-0.000
BCh2	-0.000	0.015	1.024	0.005	-0.000	0.000	-0.000	0.015	0.000	0.005	-0.000	-0.000	-0.000	0.000	-0.000
BCh3	0.000	-0.000	0.001	0.999	-0.000	-0.000	0.000	-0.000	0.000	0.001	-0.000	-0.000	-0.000	-0.000	-0.000
B0&1	1.011	0.999	0.000	0.012	1.000	0.000	0.012	0.000	0.012	0.000	0.000	0.012	0.000	0.000	-0.000
B0&2	1.010	0.148	1.038	0.029	0.148	1.000	0.029	0.148	0.005	0.029	0.148	0.005	0.029	0.005	0.005
B0&3	1.009	0.007	0.003	0.999	0.007	0.003	1.000	0.000	0.007	0.003	0.000	0.007	0.003	0.000	0.000
B1&2	0.010	1.000	1.035	0.019	0.010	0.010	0.000	1.000	0.019	0.019	0.010	0.000	0.000	0.019	0.000
B1&3	0.005	1.000	0.005	0.999	0.005	0.000	0.005	0.005	1.000	0.005	0.000	0.005	0.000	0.005	0.000
B2&3	0.000	0.043	1.041	1.000	0.000	0.000	0.000	0.043	0.043	1.000	0.000	0.000	0.000	0.043	0.000
B0&1&2	1.017	1.000	1.052	0.079	1.000	1.001	0.079	1.000	0.079	0.079	1.001	0.079	0.079	0.079	0.079
B0&1&3	1.016	1.000	0.010	1.000	1.000	0.010	1.000	0.010	1.000	0.010	0.010	1.000	0.010	0.010	0.010
B0&2&3	1.014	0.256	1.060	0.999	0.257	1.001	1.000	0.257	0.257	1.000	0.257	0.257	1.001	0.257	0.257
B1&2&3	0.019	1.000	1.054	0.999	0.019	0.019	0.019	1.000	1.000	1.000	0.019	0.019	0.019	1.000	0.019
B0&1&2&3	1.023	1.000	1.074	1.000	1.000	1.001	1.000	1.001	1.000	1.001	1.001	1.000	1.001	1.001	1.001

Tabela A.6: Resultados da injeção de luz em 50kHz e 100kHz - Grupo B

Injeção de 50kHz														
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
BCh0	1.003	0.004	0.000	0.001	0.004	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
BCh1	0.002	1.000	0.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
BCh2	-0.000	0.015	1.016	0.005	0.000	0.000	-0.000	0.015	0.000	0.005	-0.000	-0.000	-0.000	0.000	-0.000
BCh3	0.000	-0.000	0.001	0.999	0.000	-0.000	0.000	0.000	0.000	0.001	0.000	-0.000	0.000	-0.000	-0.000
B0&1	1.007	1.000	0.000	0.007	1.000	0.000	0.007	0.000	0.007	-0.000	0.000	0.007	-0.000	-0.000	-0.000
B0&2	1.006	0.144	1.027	0.028	0.144	1.000	0.028	0.144	0.004	0.028	0.144	0.004	0.028	0.004	0.004
B0&3	1.005	0.007	0.003	0.999	0.007	0.003	1.000	-0.000	0.007	0.003	-0.000	0.007	0.003	-0.000	-0.000
B1&2	0.010	0.999	1.024	0.018	0.010	0.010	0.000	1.000	0.018	0.018	0.010	0.000	0.000	0.018	0.000
B1&3	0.005	1.000	0.005	1.000	0.005	0.000	0.005	0.005	1.000	0.005	0.000	0.005	0.000	0.005	0.000
B2&3	-0.000	0.044	1.029	0.999	-0.000	0.000	0.000	0.044	0.044	1.000	0.000	0.000	0.000	0.044	0.000
B0&1&2	1.011	0.999	1.037	0.064	1.000	1.001	0.065	1.000	0.064	0.065	1.001	0.064	0.064	0.064	0.064
B0&1&3	1.010	1.000	0.010	1.000	1.000	0.010	1.000	0.010	1.000	0.010	0.010	1.000	0.010	0.010	0.010
B0&2&3	1.009	0.256	1.043	0.999	0.256	1.001	1.000	0.257	0.256	1.001	0.256	0.256	1.001	0.256	0.257
B1&2&3	0.019	1.000	1.038	0.999	0.019	0.020	0.020	1.001	1.000	1.001	0.020	0.020	0.020	1.001	0.020
B0&1&2&3	1.015	1.000	1.054	0.999	1.000	1.001	1.000	1.001	1.000	1.001	1.001	1.000	1.001	1.001	1.001
Injeção de 100kHz															
BCh0	1.000	0.003	0.000	0.001	0.003	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
BCh1	0.001	1.000	0.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
BCh2	-0.000	0.015	1.004	0.005	-0.000	0.000	0.000	0.015	0.000	0.005	-0.000	0.000	-0.000	0.000	-0.000
BCh3	0.000	0.000	0.001	1.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000
B0&1	1.001	1.000	0.000	0.034	1.000	0.000	0.034	0.000	0.034	0.000	0.000	0.034	0.000	-0.000	0.000
B0&2	1.001	0.139	1.007	0.028	0.139	1.000	0.028	0.139	0.004	0.028	0.139	0.004	0.028	0.004	0.004
B0&3	1.001	0.007	0.003	1.000	0.007	0.003	1.000	0.000	0.007	0.003	0.000	0.007	0.003	0.000	0.000
B1&2	0.009	1.000	1.006	0.018	0.009	0.009	0.000	1.000	0.018	0.018	0.009	0.000	0.000	0.018	0.000
B1&3	0.005	1.000	0.005	1.000	0.005	0.000	0.005	0.005	1.000	0.005	0.000	0.005	0.000	0.005	0.000
B2&3	0.000	0.045	1.008	1.000	0.000	0.000	0.000	0.045	0.045	1.000	0.000	0.000	0.000	0.045	0.000
B0&1&2	1.002	1.000	1.011	0.100	1.000	1.000	0.100	1.000	0.100	0.100	1.000	0.100	0.100	0.100	0.100
B0&1&3	1.002	1.000	0.010	1.000	1.000	0.010	1.000	0.010	1.000	0.010	0.010	1.000	0.010	0.010	0.010
B0&2&3	1.002	0.262	1.012	1.000	0.263	1.000	1.000	0.263	0.263	1.000	0.263	0.263	1.000	0.263	0.263
B1&2&3	0.019	1.000	1.011	1.000	0.019	0.019	0.019	1.000	1.000	1.000	0.019	0.019	0.019	1.000	0.019
B0&1&2&3	1.003	1.000	1.016	1.000	1.000	1.001	1.000	1.000	1.000	1.000	1.001	1.000	1.001	1.000	1.001

Tabela A.7: Resultados da injeção de luz em 1kHz e 5kHz - Grupo C

Injeção de 1kHz														
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
CCh0	1.006	0.002	0.000	0.002	0.001	0.002	0.001	0.001	0.002	0.001	0.001	0.001	0.001	0.001	0.001
CCh1	-0.004	1.006	0.001	-0.004	-0.000	0.000	-0.001	0.000	-0.000	-0.000	0.000	-0.000	-0.000	-0.000	-0.000
CCh2	-0.007	0.002	1.004	0.040	-0.001	-0.001	-0.001	0.000	-0.000	0.044	-0.001	-0.001	-0.001	-0.000	-0.001
CCh3	0.003	0.001	0.004	1.006	0.001	0.001	0.002	0.001	0.001	0.002	0.001	0.001	0.001	0.001	0.001
C0&1	1.007	1.011	0.000	0.006	1.002	0.000	0.001	0.001	0.001	0.000	0.000	0.001	0.000	0.000	0.000
C0&2	0.995	0.002	1.007	0.100	0.002	1.000	0.099	0.002	0.000	0.099	0.003	-0.000	0.099	-0.000	-0.000
C0&3	1.009	0.001	-0.001	0.998	-0.001	0.000	1.000	-0.001	-0.001	-0.000	-0.000	-0.000	0.000	-0.000	-0.000
C1&2	0.001	1.006	1.001	0.108	0.002	0.003	-0.000	0.998	0.107	0.107	0.003	-0.000	-0.000	0.107	-0.000
C1&3	0.005	1.008	0.006	1.002	0.001	-0.001	0.001	0.002	0.999	0.002	-0.001	0.001	-0.001	0.002	-0.001
C2&3	-0.002	0.010	1.006	1.003	0.000	-0.000	0.001	0.002	0.003	1.000	-0.000	-0.000	0.000	0.003	0.000
C0&1&2	1.005	1.019	1.001	0.180	1.000	1.000	0.178	1.000	0.178	0.179	1.000	0.178	0.178	0.178	0.178
C0&1&3	1.005	1.009	0.003	1.001	1.000	0.005	1.001	0.005	1.001	0.005	0.005	1.000	0.005	0.005	0.005
C0&2&3	1.003	0.005	1.009	1.017	0.006	1.000	1.000	0.006	0.006	1.000	0.006	0.006	1.000	0.006	0.007
C1&2&3	0.010	1.014	1.009	1.009	0.006	0.006	0.006	0.999	0.999	1.000	0.006	0.006	0.006	0.999	0.006
C0&1&2&3	1.002	1.031	1.020	1.016	0.999	0.999	0.998	1.004	1.006	1.004	0.999	0.999	0.999	1.002	0.999
Injeção de 5kHz															
CCh0	1.001	0.000	0.000	0.001	-0.000	-0.000	-0.000	-0.000	-0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000
CCh1	-0.000	1.007	0.000	0.000	0.001	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
CCh2	0.000	0.001	1.001	0.045	0.000	-0.000	-0.000	0.000	-0.000	0.046	-0.000	-0.000	-0.000	-0.000	-0.000
CCh3	-0.001	-0.000	-0.000	1.001	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
C0&1	1.002	1.014	-0.001	0.002	1.000	-0.000	0.000	-0.000	0.000	0.000	0.000	0.000	-0.000	-0.000	-0.000
C0&2	1.003	0.004	1.006	0.100	0.004	1.000	0.101	0.004	0.001	0.101	0.004	0.001	0.101	0.001	0.001
C0&3	1.002	-0.000	0.000	1.005	-0.000	-0.000	1.000	-0.000	-0.000	-0.000	-0.000	-0.000	0.000	-0.000	-0.000
C1&2	0.002	1.011	1.005	0.109	0.003	0.003	0.000	0.999	0.109	0.109	0.003	0.000	0.000	0.109	0.000
C1&3	0.002	1.010	0.001	1.003	0.003	0.000	0.002	0.001	1.000	0.002	-0.000	0.002	-0.000	0.002	-0.000
C2&3	-0.001	0.002	1.008	1.009	-0.000	0.000	0.000	0.002	0.002	1.000	-0.000	-0.000	0.000	0.002	-0.000
C0&1&2	1.003	1.023	1.009	0.193	1.000	1.000	0.194	1.000	0.194	0.194	1.000	0.194	0.194	0.194	0.194
C0&1&3	1.003	1.020	0.003	1.009	1.000	0.004	1.000	0.004	1.000	0.004	0.004	1.000	0.004	0.004	0.004
C0&2&3	1.003	0.007	1.012	1.014	0.007	1.000	1.000	0.007	0.007	1.000	0.007	0.007	1.000	0.007	0.007
C1&2&3	0.007	1.019	1.014	1.015	0.007	0.007	0.007	1.002	1.002	1.002	0.007	0.007	0.007	1.001	0.007
C0&1&2&3	1.005	1.041	1.023	1.023	1.000	1.000	1.000	1.004	1.004	1.004	1.000	1.000	1.000	1.002	1.000

Tabela A.8: Resultados da injeção de luz em 10kHz e 25kHz - Grupo C

Injeção de 10kHz															
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
CCh0	1.001	0.000	-0.000	-0.000	0.000	0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
CCh1	0.000	1.009	-0.000	-0.000	0.001	-0.000	-0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
CCh2	-0.000	0.000	1.002	0.046	0.000	0.000	0.000	0.001	0.000	0.046	0.000	0.000	0.000	0.000	0.000
CCh3	0.000	0.001	0.000	1.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
C0&1	1.001	1.020	0.000	-0.000	1.000	0.000	0.000	0.000	0.000	-0.000	0.000	0.000	0.000	0.000	0.000
C0&2	1.002	0.004	1.007	0.099	0.003	1.000	0.099	0.003	0.001	0.099	0.003	0.001	0.099	0.001	0.001
C0&3	1.001	0.000	0.000	1.006	-0.000	0.000	1.000	-0.000	-0.000	0.000	-0.000	-0.000	0.000	-0.000	-0.000
C1&2	0.003	1.016	1.009	0.110	0.004	0.004	0.001	1.000	0.111	0.111	0.004	0.001	0.001	0.111	0.001
C1&3	0.003	1.013	0.002	1.006	0.002	-0.000	0.002	0.001	1.000	0.001	-0.000	0.002	-0.000	0.001	-0.000
C2&3	-0.000	0.002	1.008	1.010	-0.000	-0.000	0.000	0.002	0.002	1.000	-0.000	-0.000	0.000	0.002	-0.000
C0&1&2	1.002	1.031	1.012	0.194	1.000	1.000	0.194	1.000	0.195	0.195	1.000	0.194	0.194	0.194	0.194
C0&1&3	1.002	1.029	0.004	1.012	1.000	0.004	1.000	0.004	1.000	0.004	0.004	1.000	0.004	0.004	0.004
C0&2&3	1.001	0.007	1.015	1.020	0.007	1.000	1.000	0.007	0.007	1.001	0.007	0.007	1.000	0.007	0.007
C1&2&3	0.007	1.024	1.016	1.019	0.007	0.007	0.007	1.001	1.001	1.001	0.007	0.007	0.007	1.001	0.007
C0&1&2&3	1.002	1.048	1.025	1.028	1.000	1.000	1.000	1.003	1.003	1.002	1.000	1.000	1.000	1.002	1.000
Injeção de 25kHz															
CCh0	1.000	0.000	-0.000	-0.000	-0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
CCh1	0.001	1.010	-0.000	0.000	0.001	-0.000	-0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
CCh2	0.000	0.001	1.001	0.046	-0.000	0.000	0.000	0.001	0.000	0.046	0.000	0.000	0.000	0.000	0.000
CCh3	-0.000	-0.000	0.000	1.003	-0.000	-0.000	0.000	-0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000
C0&1	1.001	1.023	0.000	-0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
C0&2	1.001	0.003	1.008	0.100	0.003	1.000	0.100	0.003	0.001	0.100	0.003	0.001	0.100	0.001	0.001
C0&3	1.001	0.000	0.000	1.008	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.000
C1&2	0.004	1.018	1.009	0.110	0.004	0.004	0.001	1.000	0.111	0.111	0.004	0.001	0.001	0.111	0.001
C1&3	0.002	1.015	0.002	1.006	0.002	0.000	0.002	0.002	1.000	0.002	0.000	0.002	0.000	0.002	0.000
C2&3	-0.000	0.002	1.009	1.013	0.000	0.000	0.000	0.002	0.002	1.000	-0.000	0.000	0.000	0.002	-0.000
C0&1&2	1.001	1.039	1.014	0.194	1.000	1.000	0.194	1.000	0.194	0.194	1.000	0.194	0.194	0.194	0.194
C0&1&3	1.001	1.033	0.004	1.012	1.000	0.004	1.000	0.004	1.000	0.004	0.004	1.000	0.004	0.004	0.004
C0&2&3	1.001	0.007	1.015	1.021	0.007	1.000	1.000	0.007	0.007	1.000	0.007	0.007	1.000	0.007	0.007
C1&2&3	0.007	1.027	1.016	1.020	0.007	0.007	0.007	1.000	1.000	1.000	0.007	0.007	0.007	1.000	0.007
C0&1&2&3	1.001	1.054	1.025	1.029	1.000	1.000	1.000	1.001	1.001	1.001	1.000	1.000	1.000	1.001	1.001

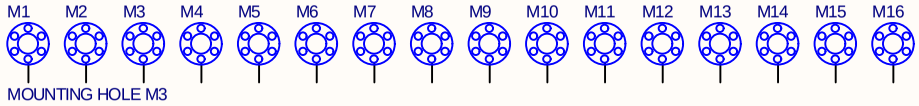
Tabela A.9: Resultados da injeção de luz em 50kHz e 100kHz - Grupo C

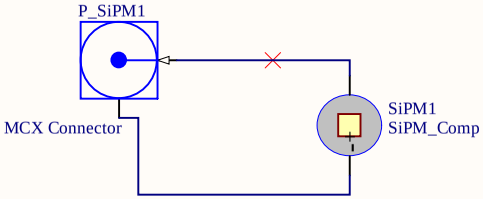
Injeção de 50kHz															
InjectedCh	ACh0	ACh1	ACh2	ACh3	A0&1	A0&2	A0&3	A1&2	A1&3	A2&3	A0&1&2	A0&1&3	A0&2&3	A1&2&3	A0&1&2&3
CCh0	1.000	0.000	-0.000	-0.000	0.000	0.000	0.000	-0.000	-0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000
CCh1	0.001	1.006	-0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CCh2	-0.000	0.001	1.001	0.047	-0.000	0.000	0.000	0.001	0.000	0.047	0.000	0.000	0.000	0.000	0.000
CCh3	-0.000	0.000	0.000	1.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
C0&1	1.000	1.017	0.000	-0.000	1.000	0.000	0.000	0.000	0.000	-0.000	0.000	0.000	-0.000	-0.000	-0.000
C0&2	1.000	0.003	1.005	0.104	0.003	1.000	0.104	0.003	0.001	0.104	0.003	0.001	0.104	0.001	0.001
C0&3	1.000	-0.000	0.000	1.004	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
C1&2	0.004	1.013	1.005	0.105	0.004	0.004	0.001	1.000	0.105	0.105	0.004	0.001	0.001	0.105	0.001
C1&3	0.002	1.010	0.002	1.004	0.002	-0.000	0.002	0.002	1.000	0.002	-0.000	0.002	-0.000	0.002	-0.000
C2&3	-0.000	0.002	1.005	1.008	0.000	0.000	0.000	0.002	0.002	1.000	-0.000	0.000	0.000	0.002	-0.000
C0&1&2	1.000	1.030	1.009	0.192	1.000	1.000	0.193	1.000	0.193	0.193	1.000	0.193	0.193	0.193	0.193
C0&1&3	1.000	1.025	0.004	1.008	1.000	0.004	1.000	0.004	1.000	0.004	0.004	1.000	0.004	0.004	0.004
C0&2&3	1.000	0.008	1.010	1.015	0.007	1.000	1.000	0.007	0.008	1.000	0.007	0.008	1.000	0.008	0.008
C1&2&3	0.007	1.019	1.010	1.014	0.007	0.007	0.007	1.000	1.000	1.000	0.007	0.007	0.007	1.001	0.007
C0&1&2&3	1.000	1.040	1.016	1.019	1.000	1.000	1.000	1.001	1.001	1.000	1.001	1.001	1.000	1.001	1.001
Injeção de 100kHz															
CCh0	0.999	-0.000	-0.000	-0.000	0.000	0.000	0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
CCh1	0.001	1.001	0.000	0.000	0.001	-0.000	0.000	0.000	0.000	0.000	-0.000	0.000	0.000	-0.000	-0.000
CCh2	0.000	0.001	1.000	0.048	0.000	0.000	0.000	0.001	0.000	0.048	0.000	0.000	0.000	0.000	0.000
CCh3	-0.000	-0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
C0&1	0.999	1.004	0.000	0.000	1.000	0.000	0.000	0.000	0.000	-0.000	0.000	0.000	0.000	-0.000	0.000
C0&2	0.999	0.003	1.001	0.101	0.003	1.000	0.101	0.003	0.001	0.101	0.004	0.001	0.101	0.001	0.001
C0&3	0.999	0.000	0.000	1.001	0.000	0.000	1.000	-0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
C1&2	0.004	1.003	1.001	0.111	0.004	0.004	0.001	1.000	0.111	0.111	0.004	0.001	0.001	0.111	0.001
C1&3	0.002	1.002	0.002	1.000	0.002	0.000	0.002	0.002	1.000	0.002	0.000	0.002	0.000	0.002	0.000
C2&3	-0.000	0.002	1.001	1.002	-0.000	0.000	0.000	0.002	0.002	1.000	-0.000	0.000	0.000	0.002	0.000
C0&1&2	0.999	1.007	1.002	0.201	1.000	1.000	0.201	1.000	0.201	0.201	1.000	0.201	0.201	0.201	0.201
C0&1&3	0.999	1.006	0.004	1.002	1.000	0.004	1.000	0.004	1.000	0.004	0.004	1.000	0.004	0.004	0.004
C0&2&3	0.999	0.007	1.002	1.004	0.008	1.000	1.000	0.008	0.008	1.000	0.008	0.008	1.000	0.008	0.008
C1&2&3	0.007	1.005	1.002	1.003	0.007	0.007	0.007	1.000	1.000	1.000	0.007	0.007	0.007	1.000	0.007
C0&1&2&3	1.000	1.011	1.004	1.005	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.001	1.001

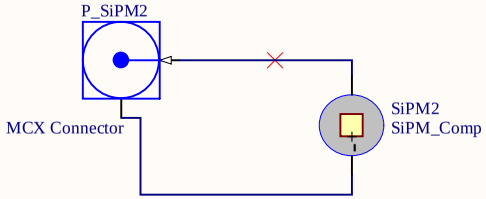
Apêndice B

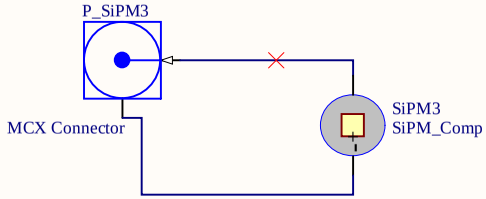
Projeto da placa adaptadora SiPM/LED

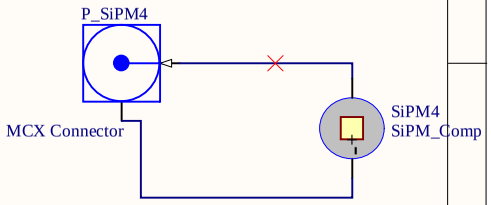
Este apêndice é composto pelo projeto da placa adaptadora SiPM/LED, desenvolvido no *software Altium Designer*. A placa é utilizada em conjunto com a mecânica óptica do experimento. O apêndice contém tanto o esquemático quanto o *layout*, desenvolvido em uma camada, nesta ordem.

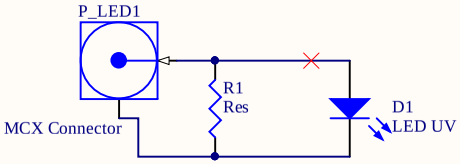


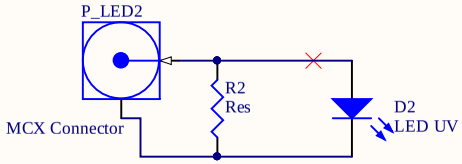


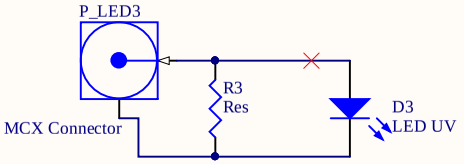


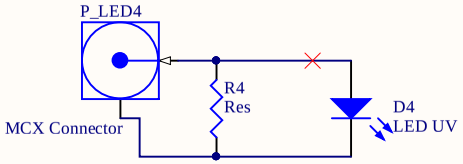




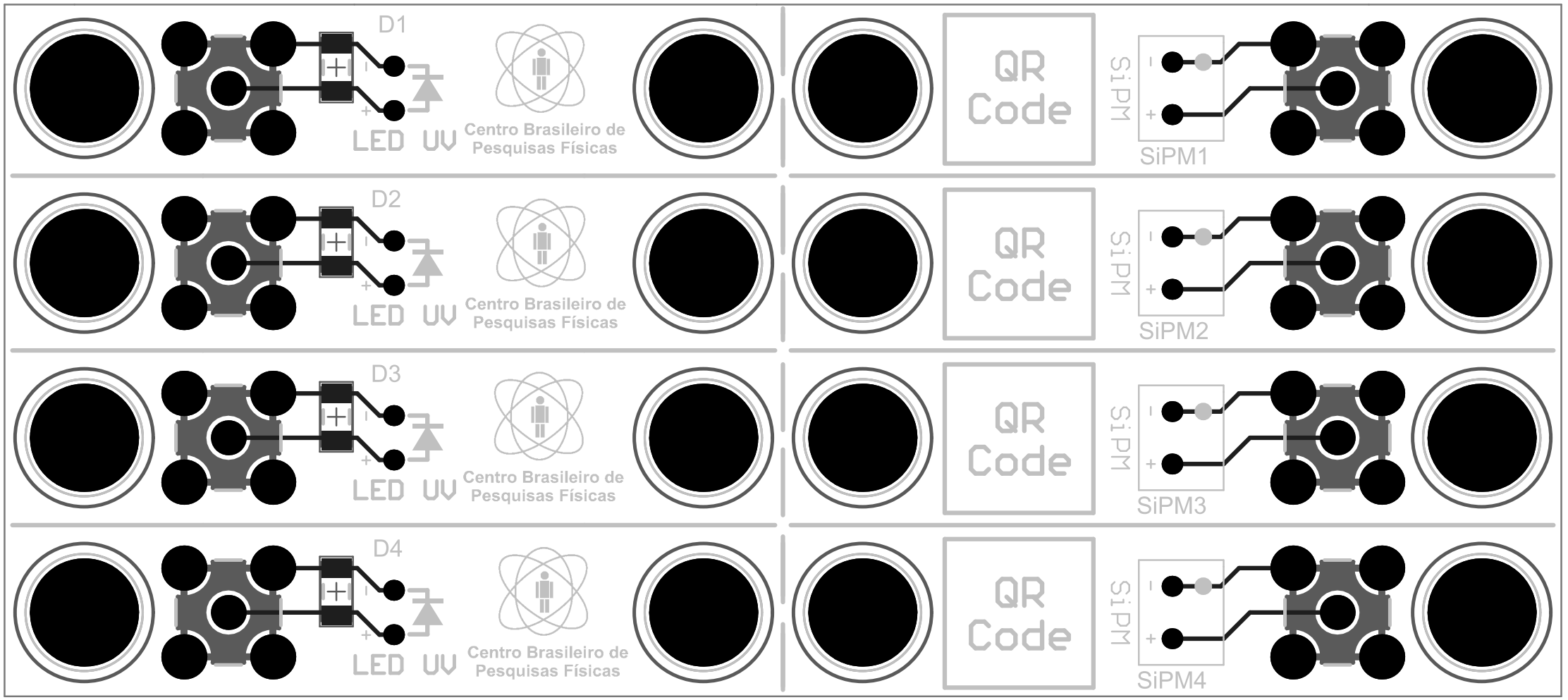








Title	**SciTile Mechanical Board**		*Centro Brasileiro de Pesquisas Físicas COHEP Rio de Janeiro - Brasil*
Size: A4	Number: 1	Revision: 1.0	*Diogo Ayres Rocha*
Date: 19/09/2023	Time: 17:23:57	Sheet 1 of 3	
File: E:\CERNBox\CBPF\Mestrado\SciTile_SiPMAdapter\TOP_SCH.schdoc			


D1
LED UV
Centro Brasileiro de Pesquisas Físicas
QR Code
SiPM
SiPM1
D2
LED UV
Centro Brasileiro de Pesquisas Físicas
QR Code
SiPM
SiPM2
D3
LED UV
Centro Brasileiro de Pesquisas Físicas
QR Code
SiPM
SiPM3
D4
LED UV
Centro Brasileiro de Pesquisas Físicas
QR Code
SiPM
SiPM4

Apêndice C

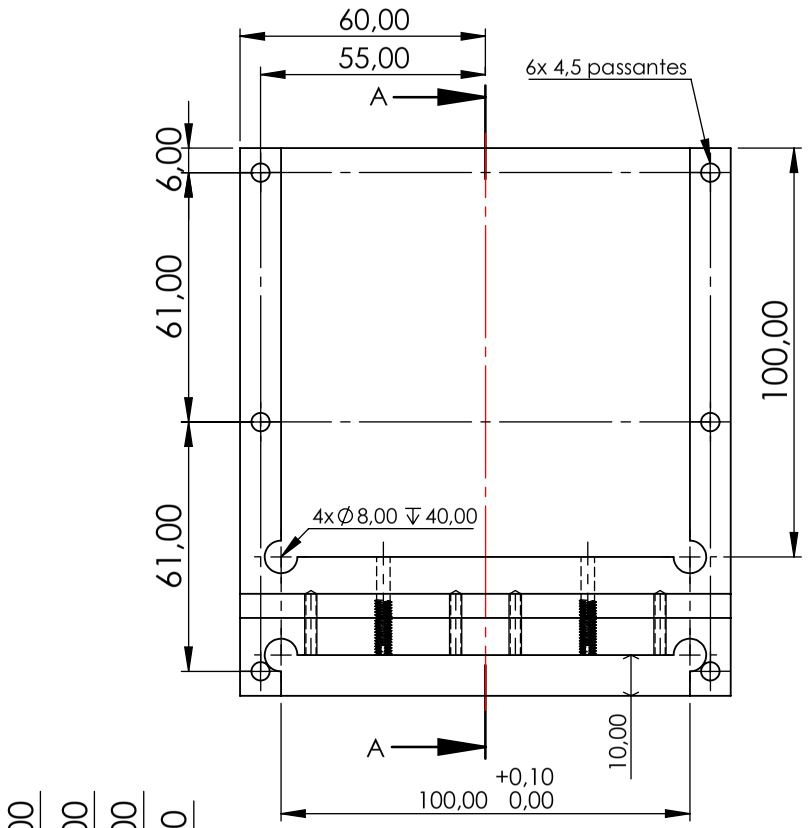
Projetos das peças mecânicas da óptica do experimento

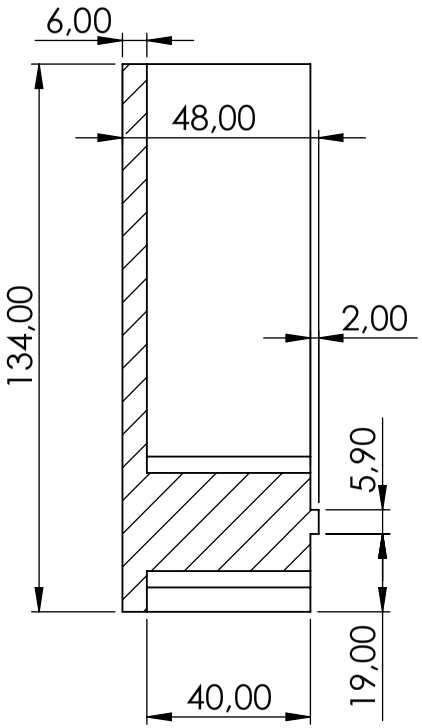
Este apêndice é composto pelos projetos das peças que compõem o sistema óptico do experimento.

Os projetos estão na seguinte ordem:

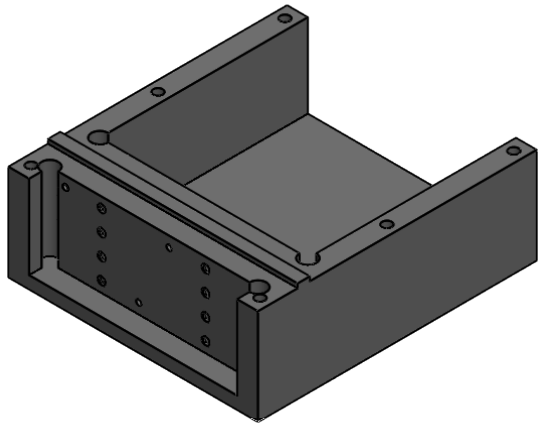
1. Extensor frontal;
2. Tampa do extensor frontal;
3. Extensor traseiro;
4. Tampa do extensor traseiro;
5. Holder;
6. Parafuso óptico.

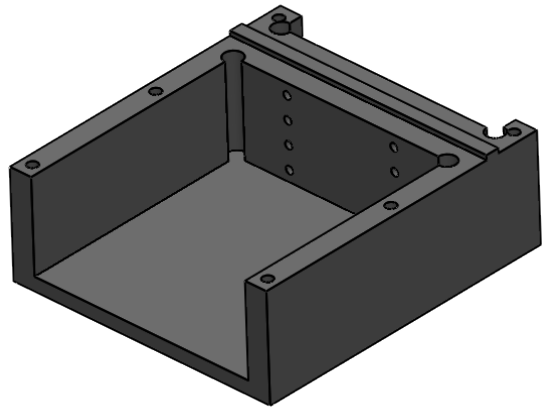
Extensor FRONTAL 2023 desenho

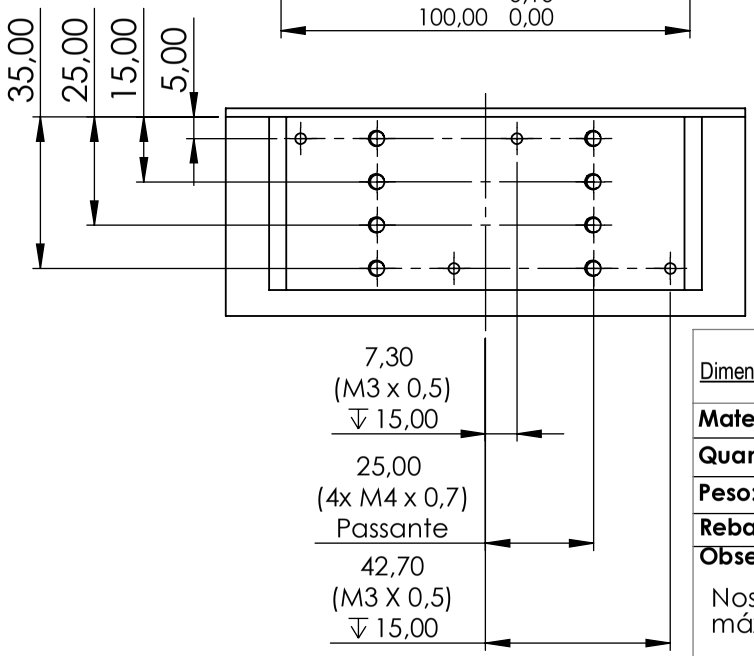




SECTION A-A
SCALE 1 : 1.8





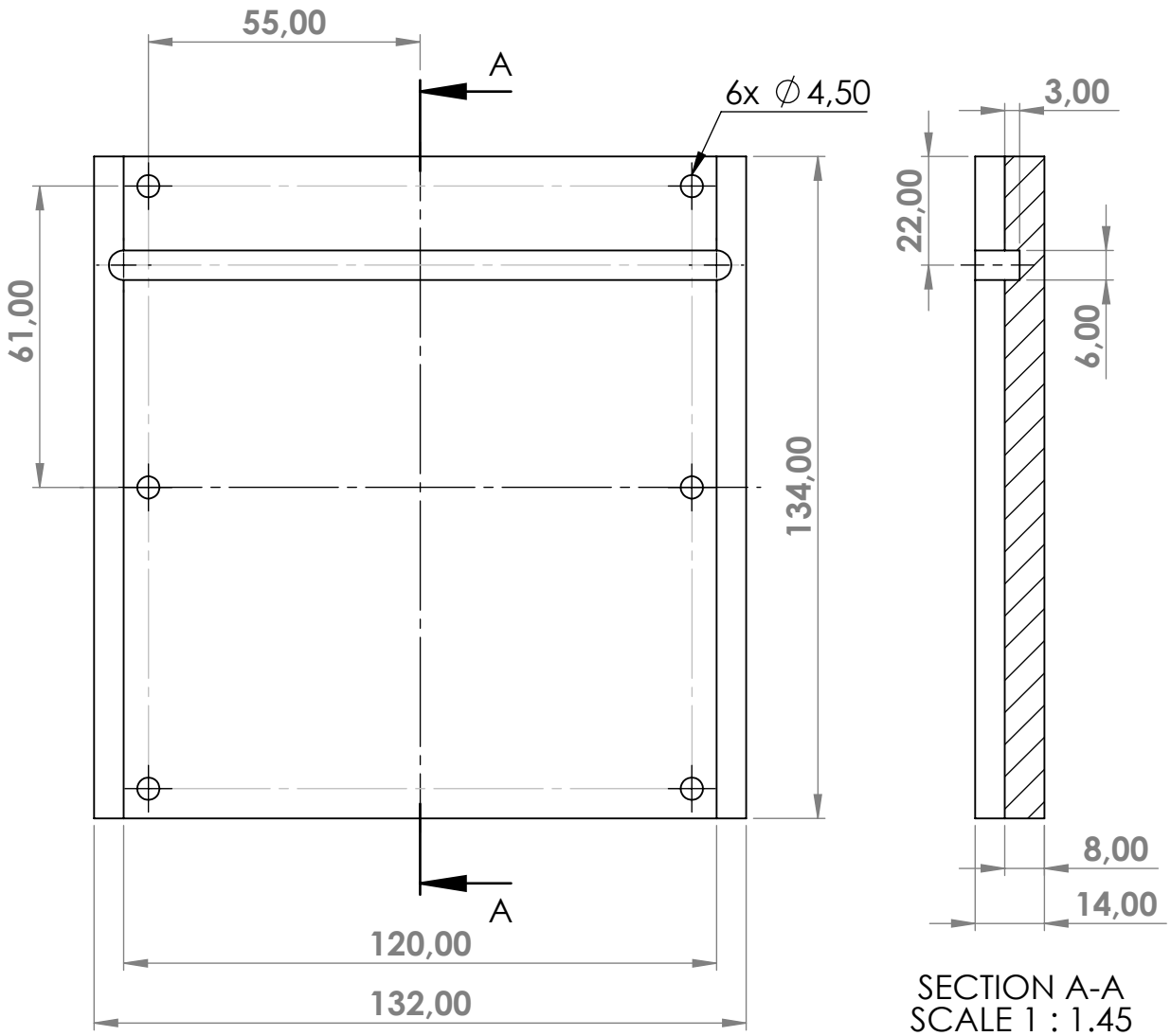


Dimensões em milímetros	3º diedro	**COTEC - LITMEC** Laboratório de Instrumentação e Tecnologia Mecânica	
Material: Polímero		**Projeto:** Extensor FRONTAL 2023 VERSÃO 3	
Quantidade:	**Escala:** 1:1.7		
Peso: g	**Tolerância:** ± 0,1	Solicitante: André Massafferri	Dep: COHEP
Rebarbar e Quebrar arestas agudas			
Observação: Nos furos, as roscas devem ir até o máximo que a ferramenta permitir		N° O.S: 0063/23	**Data:** 25/07/2023
		Versão: **Folha:** 1/1	**Desenhista:** Barbara / José Eduardo / Hiago

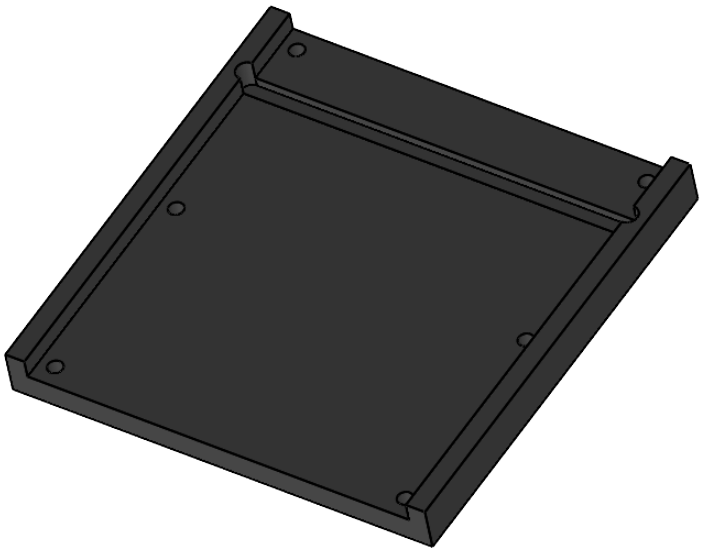
Z:\2023\NUCLEO DE PROJETOS MECANICOS -NPM\COHEP\PROF ANDRE MASSA\Projeto CRE4AT - 2023\Sistema Optico V3

Tampa EXTENSOR FRONTAL 2023 desenho

VERSÃO FINAL



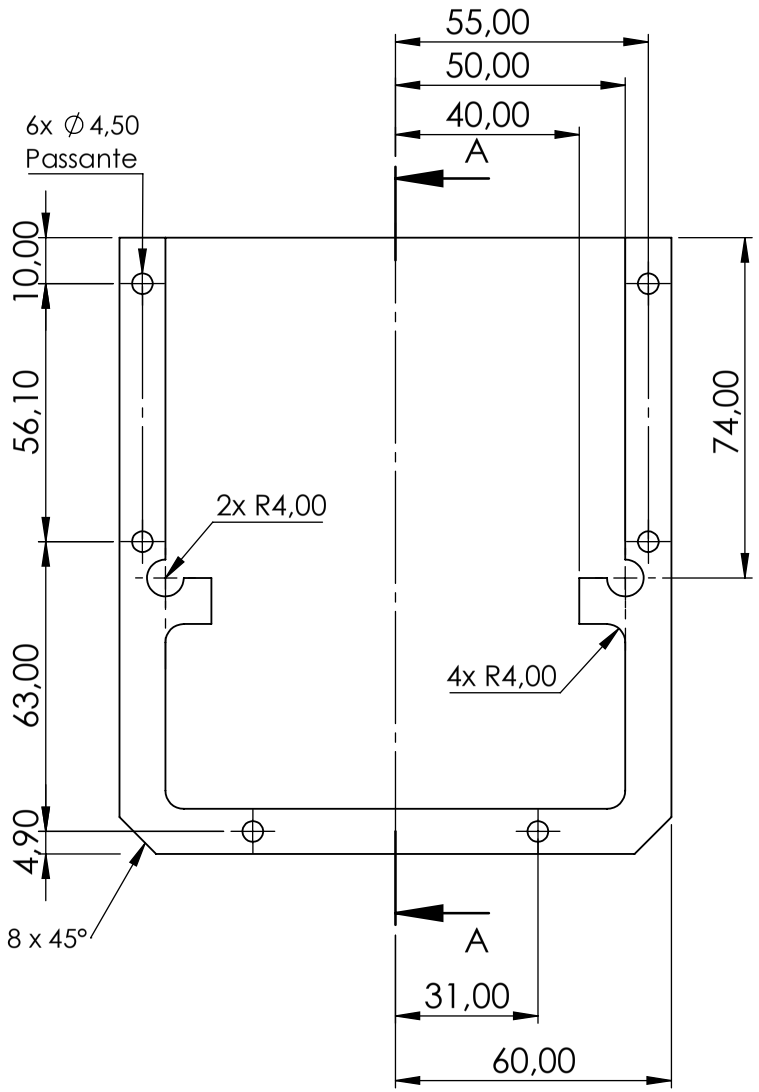
SECTION A-A
SCALE 1 : 1.45

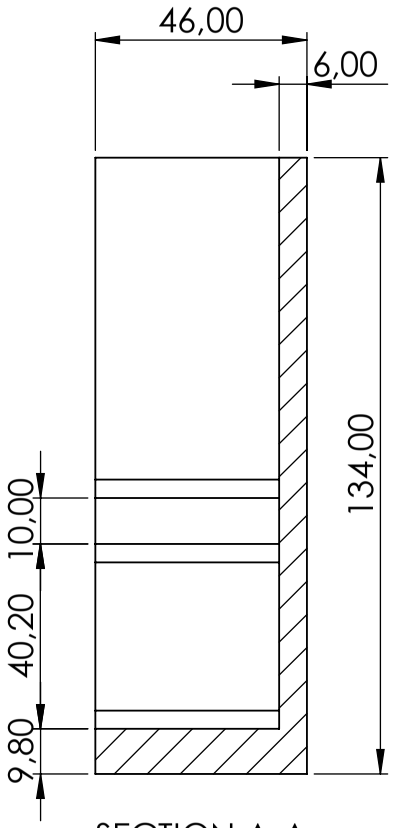


Dimensões em milímetros	3° diedro	**COTEC - LITMEC** Laboratório de Instrumentação e Tecnologia Mecânica	
Material: Polímero		**Projeto:** Tampa extensor frontal 2023	
Quantidade:	**Escala:** 1:1.45		
Peso: g	**Tolerância:** ± 0,1	Solicitante: André Massafferri	Dep: COHEP
Rebarbar e Quebrar arestas agudas			
Observação:		N° O.S: ???????	**Data:** 03/03/2023
		Versão: **Folha:** 1/1	**Desenhista:** Barbara / José Eduardo / Hiago

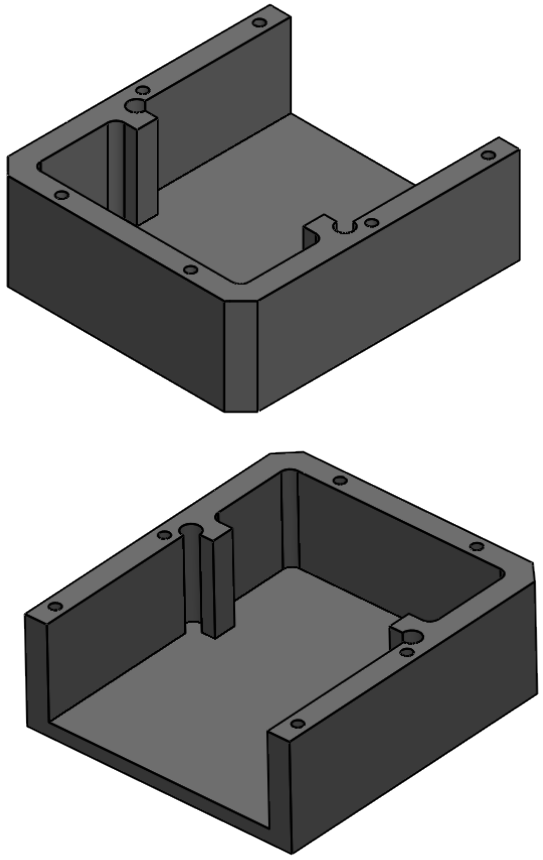
L:\2023\NUCLEO DE PROJETOS MECANICOS -NPM\COHEP\PROF ANDRE MASSA\Projeto CRE4AT - 2023\Sistema Optico- ARQUIVOS REVISADOS 2023\

Extensor traseiro 2023 desenho





SECTION A-A
SCALE 1 : 1.6

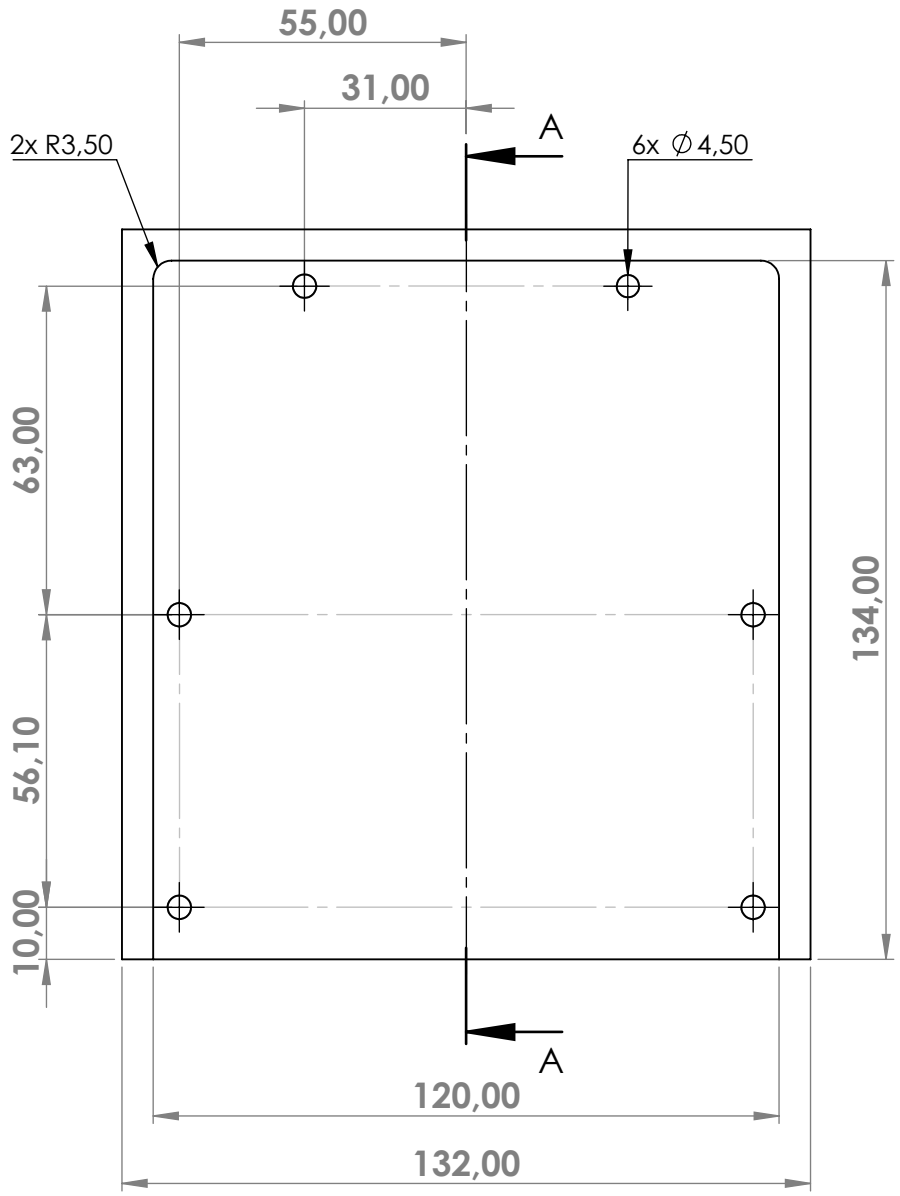


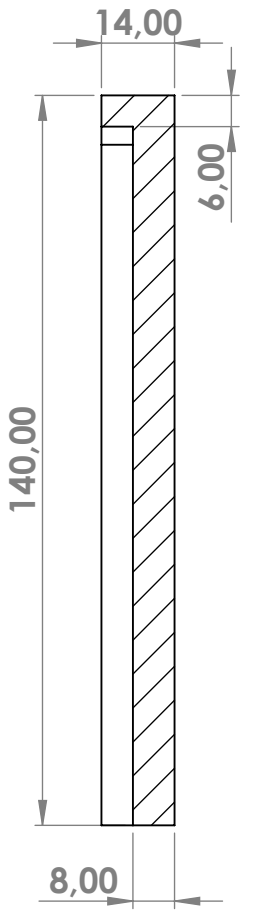
Dimensões em milímetros	3º diedro	**COTEC - LITMEC** Laboratório de Instrumentação e Tecnologia Mecânica	
Material: Polímero		**Projeto:** Extensor traseiro 2023 VERSÃO 3	
Quantidade:	**Escala:** 1:1.6		
Peso: g	**Tolerância:** ± 0,1	Solicitante: André Massafferri	Dep: COHEP
Rebarbar e Quebrar arestas agudas			
Observação: Atenção para a data do desenho		Nº O.S: 0063/23	**Data:** 25/07/2023
		Versão: **Folha:** 1/1	**Desenhista:** Barbara / josé Eduardo / Hiago

Z:\2023\NUCLEO DE PROJETOS MECANICOS -NPM\COHEP\PROF ANDRE MASSA\Projeto CRE4AT - 2023\Sistema Optico V3

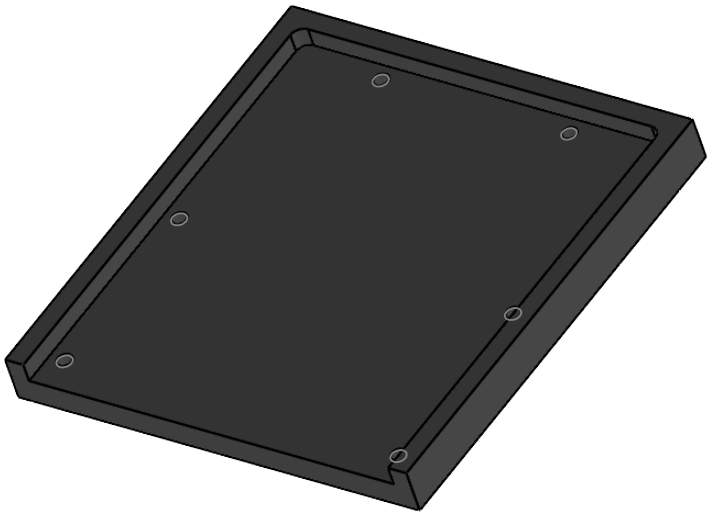
Tampa Extensor Traseiro 2023 desenho

VERSÃO FINAL





SECTION A-A
SCALE 1 : 1.45

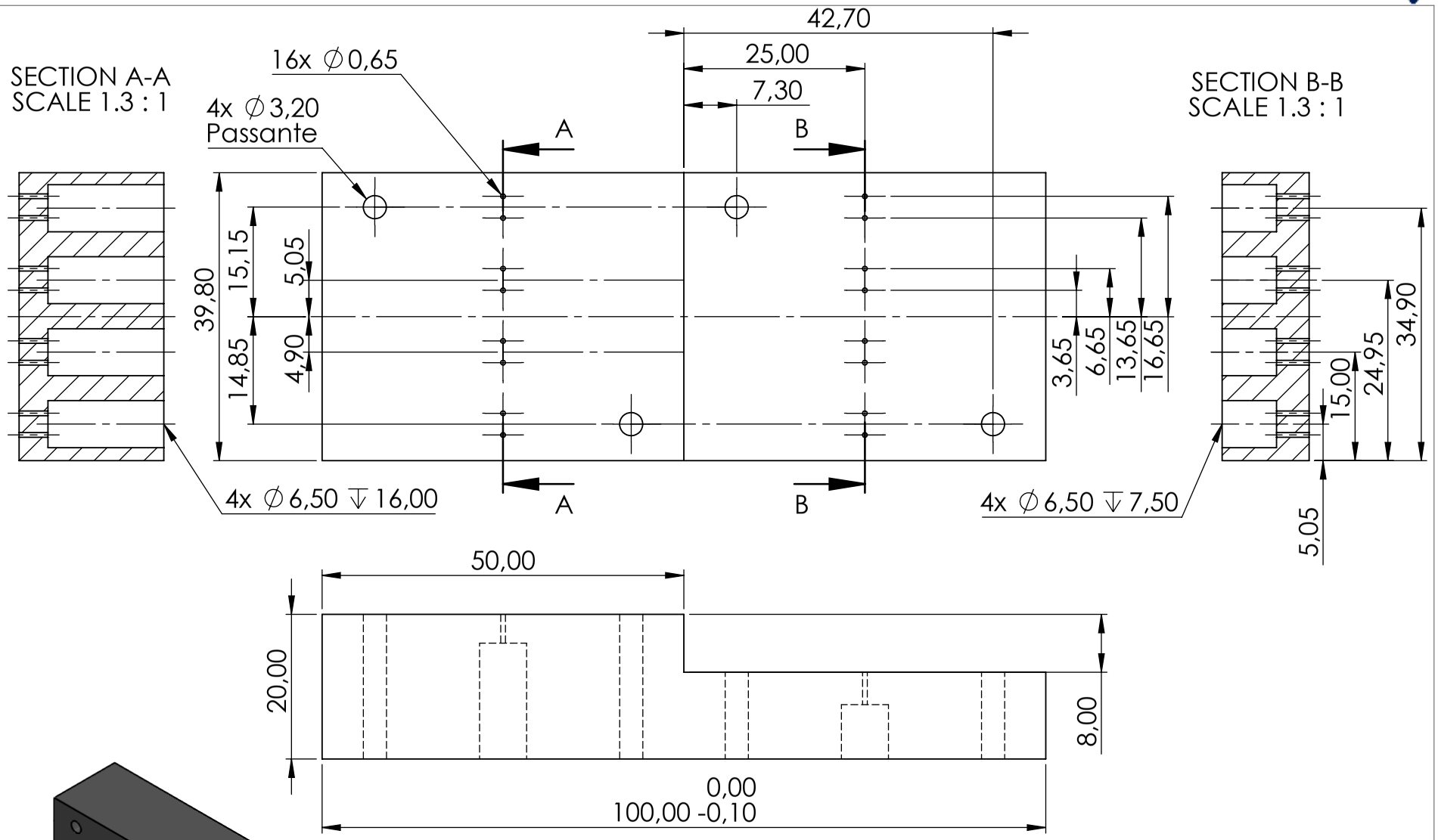


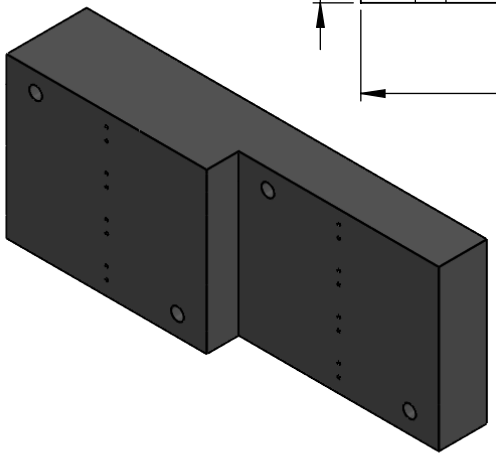
Dimensões em milímetros	3° diedro	**COTEC - LITMEC** Laboratório de Instrumentação e Tecnologia Mecânica	
Material: Polímero		**Projeto:** Tampa extensor traseiro 2023	
Quantidade:	**Escala:** 1.45		
Peso: g	**Tolerância:** ± 0,1	Solicitante: André Massafferri	Dep: COHEP
Rebarbar e Quebrar arestas agudas			
Observação:		N° O.S: ????????	**Data:** 03/03/2023
		Versão: **Folha:** 1/1	**Desenhista:** Barbara / José Eduardo / Hiago

L:\2023\NUCLEO DE PROJETOS MECANICOS -NPM\COHEP\PROF ANDRE MASSA\Projeto CRE4AT - 2023\Sistema Optico- ARQUIVOS REVISADOS 2023\

Holder 8 em 1 2023 desenho

VERSÃO FINAL

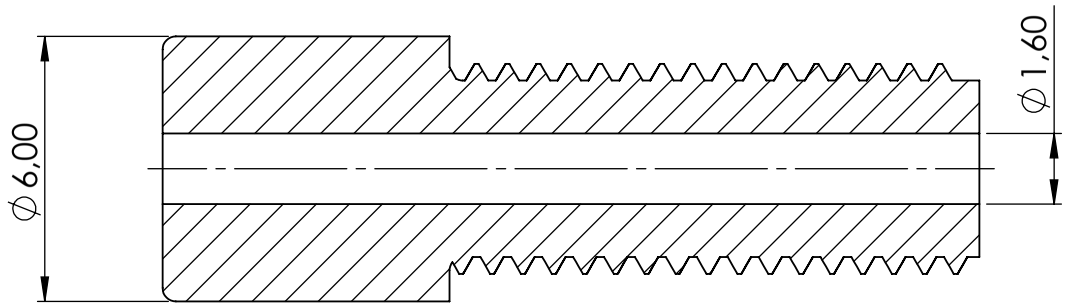




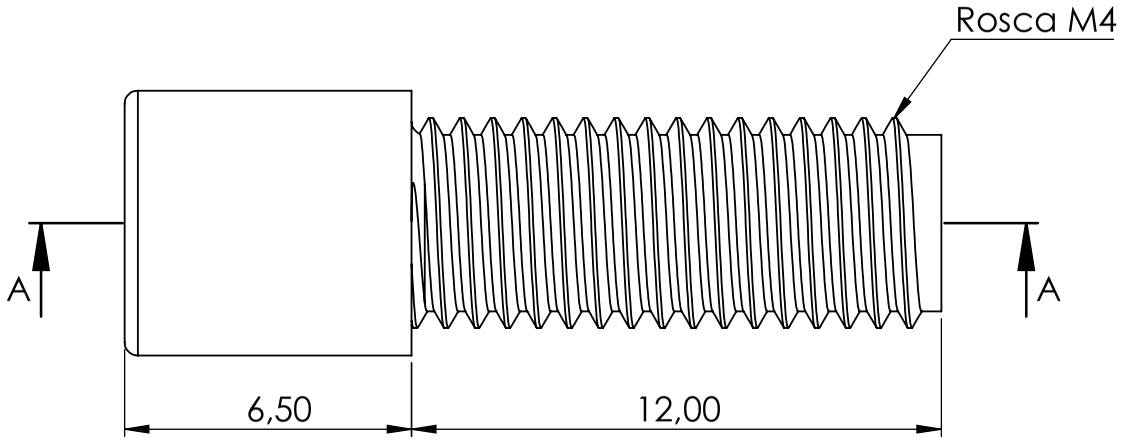
Dimensões em milímetros	3º diedro	**COTEC - LITMEC** Laboratório de Instrumentação e Tecnologia Mecânica	
Material: Polímero		**Projeto:** Holder 8 em 1 2023	
Quantidade:	**Escala:** 1.3:1		
Peso: g	**Tolerância:** ± 0,1	Solicitante: André Massafferri	Dep: COHEP
Rebarbar e Quebrar arestas agudas			
Observação:		N° O.S: ????????	**Data:** 06/03/2023
		Versão: **Folha:** 1/1	**Desenhista:** Barbara / José Eduardo / Hiago

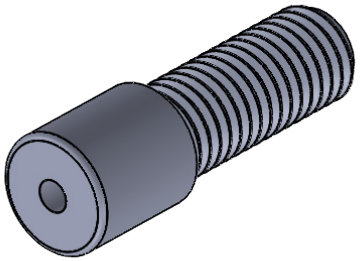
Z:\2023\NUCLEO DE PROJETOS MECANICOS -NPM\COHEP\PROF ANDRE MASSA\Projeto CRE4AT - 2023\Sistema Optico- ARQUIVOS REVISADOS 2023

parafuso optico 2023



SEÇÃO A-A
ESCALA 6 : 1





Dimensões em milímetros	3º diedro	**COTEC - LITMEC** Laboratório de Instrumentação e Tecnologia Mecânica	
Material: Acetal		**Projeto:** Parafuso Optico 2023	
Quantidade: 35	**Escala:** 6:1		
Peso: g	**Tolerância:** ± 0,1	Solicitante: André Massafferri	Dep: COHEP
Rebarbar e Quebrar arestas agudas			
Observação:		N° O.S: ??????????	**Data:** 26/05/2023
		Versão: **Folha:** 1/1	**Desenhista:** Barbara / José Eduardo / Hiago

Z:\2023\NUCLEO DE PROJETOS MECANICOS -NPM\COHEP\PROF ANDRE MASSA\Projeto CRE4AT - 2023\Sistema Optico- ARQUIVOS F

Apêndice D

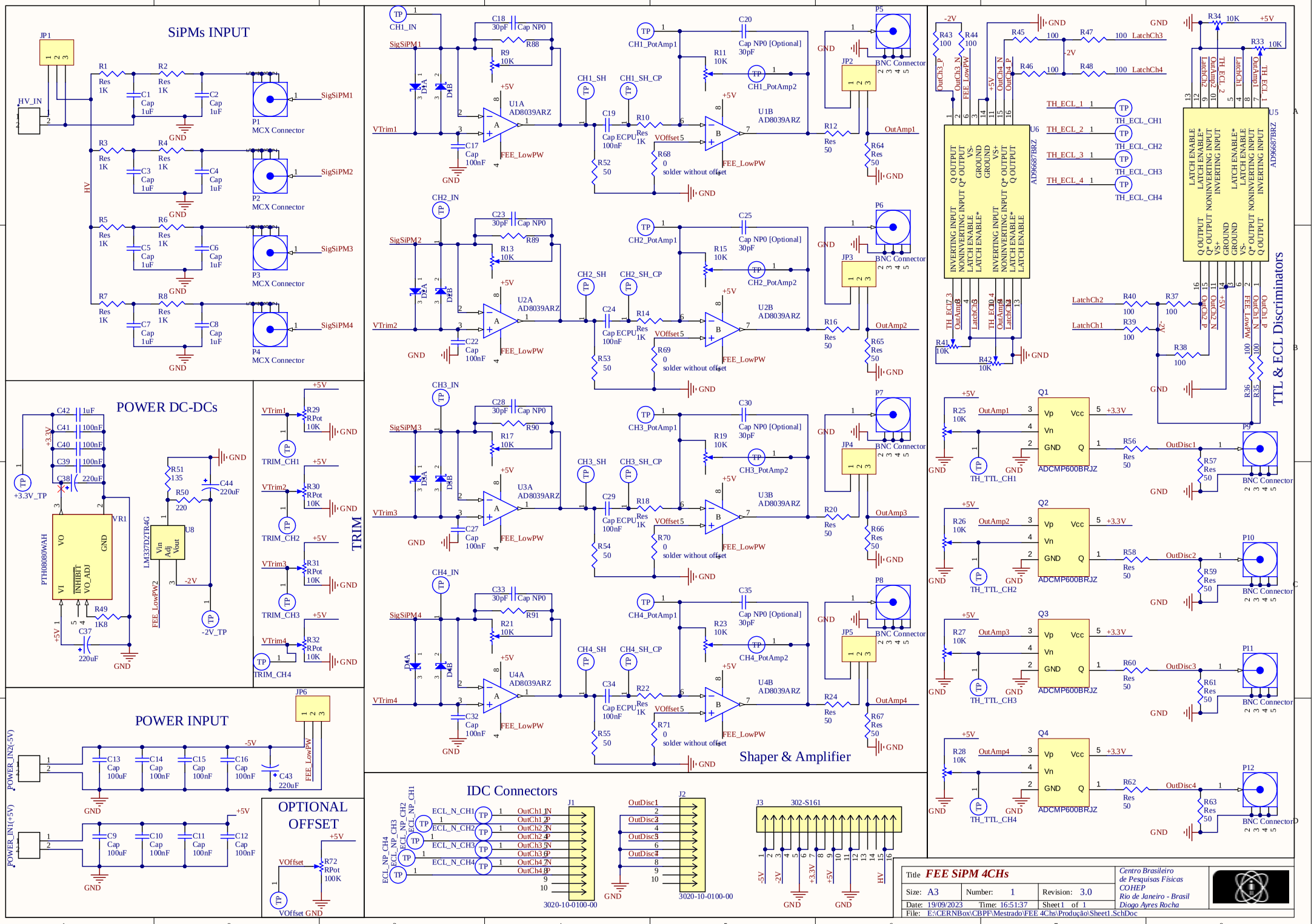
Projeto da eletrônica de front-end

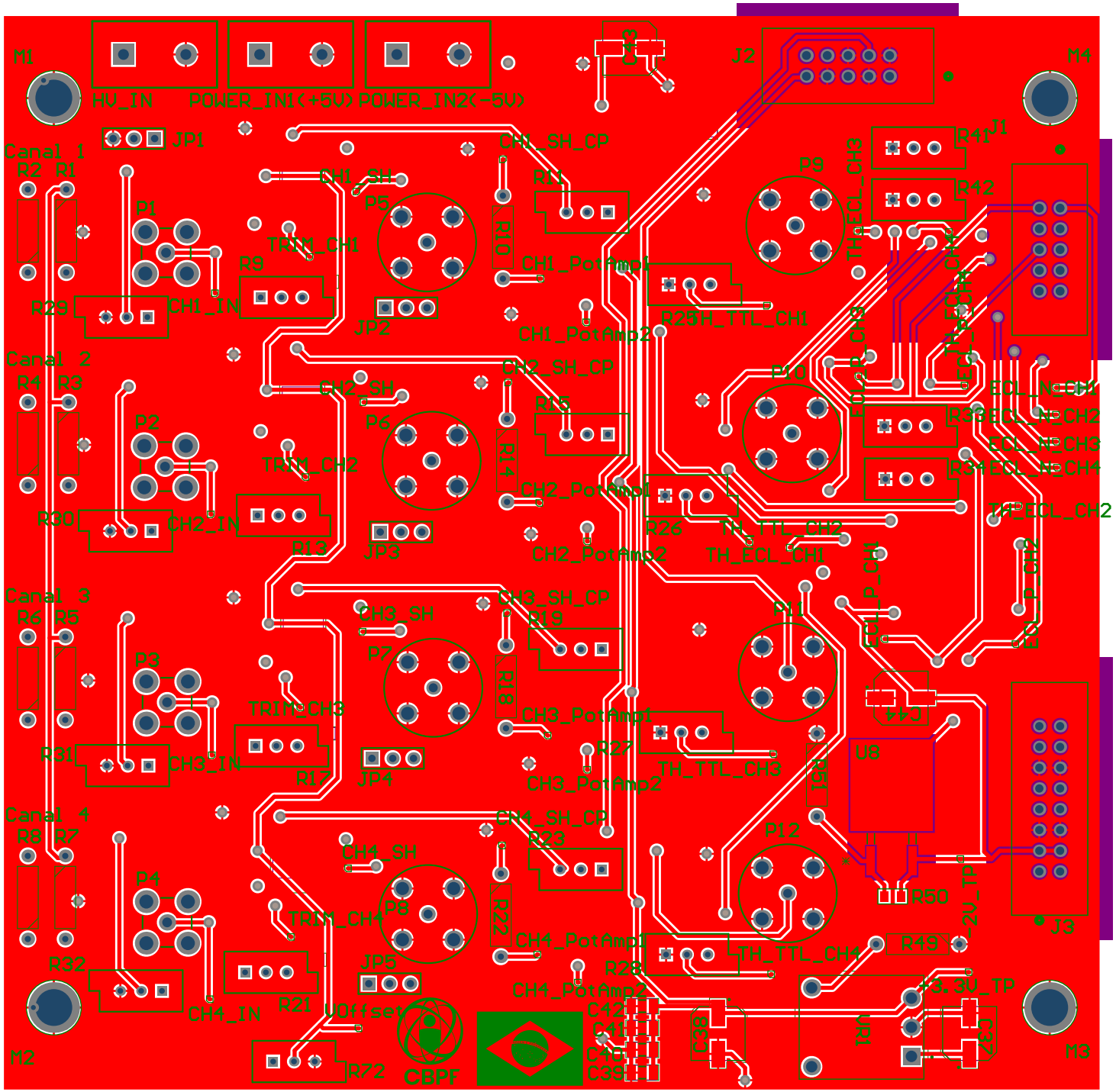
Este apêndice é composto pelo projeto da eletrônica de *front-end* de 4 canais, desenvolvido no *software Altium Designer*. O *layout* foi projetado com 4 camadas, sendo elas as seguintes:

1. Camada de sinais superior;
2. Plano +5V;
3. Plano -5V;
4. Camada de sinais inferior.

O projeto, neste apêndice, será mostrado na seguinte ordem:

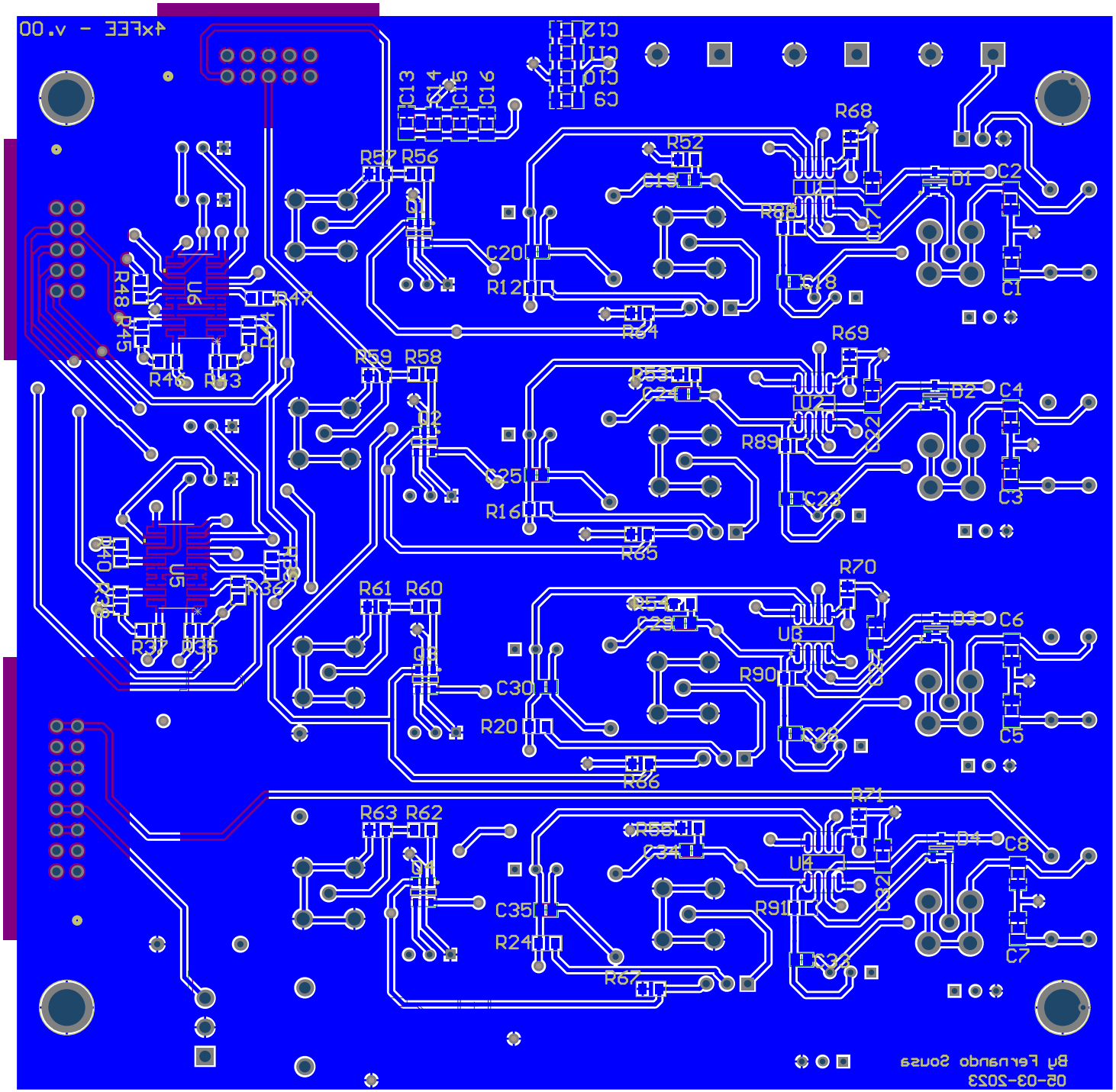
1. Esquemático da placa;
2. Layout camada de sinais superior;
3. Layout camada de sinais inferior.


SiPMs INPUT
POWER DC-DCs
POWER INPUT
OPTIONAL OFFSET
TRIM
Shaper & Amplifier
IDC Connectors
TTL & ECL Discriminators
FEE SiPM 4CHs
Centro Brasileiro de Pesquisas Físicas
COHEP
Rio de Janeiro - Brasil
Diogo Ayres Rocha
Date: 19/09/2023
Time: 16:51:37
Sheet1 of 1
File: E:\CERNBox\CBPF\Mestrado\FEE 4Chs\Produção\Sheet1.SchDoc



Centro Brasileiro de Pesquisas Fisicas - CBPF	
Data: 05/03/2023	Rev: v.00
Autores:	Esquematico - Diogo Ayres
	PCB - Fernando Sousa

Centro Brasileiro de Pesquisas Fisicas - CBPF	
Data: 05/03/2023	Rev: v.00
Autores:	Esquematico - Diogo Ayres
	PCB - Fernando Sousa



Apêndice E

Projeto da placa do DAQ

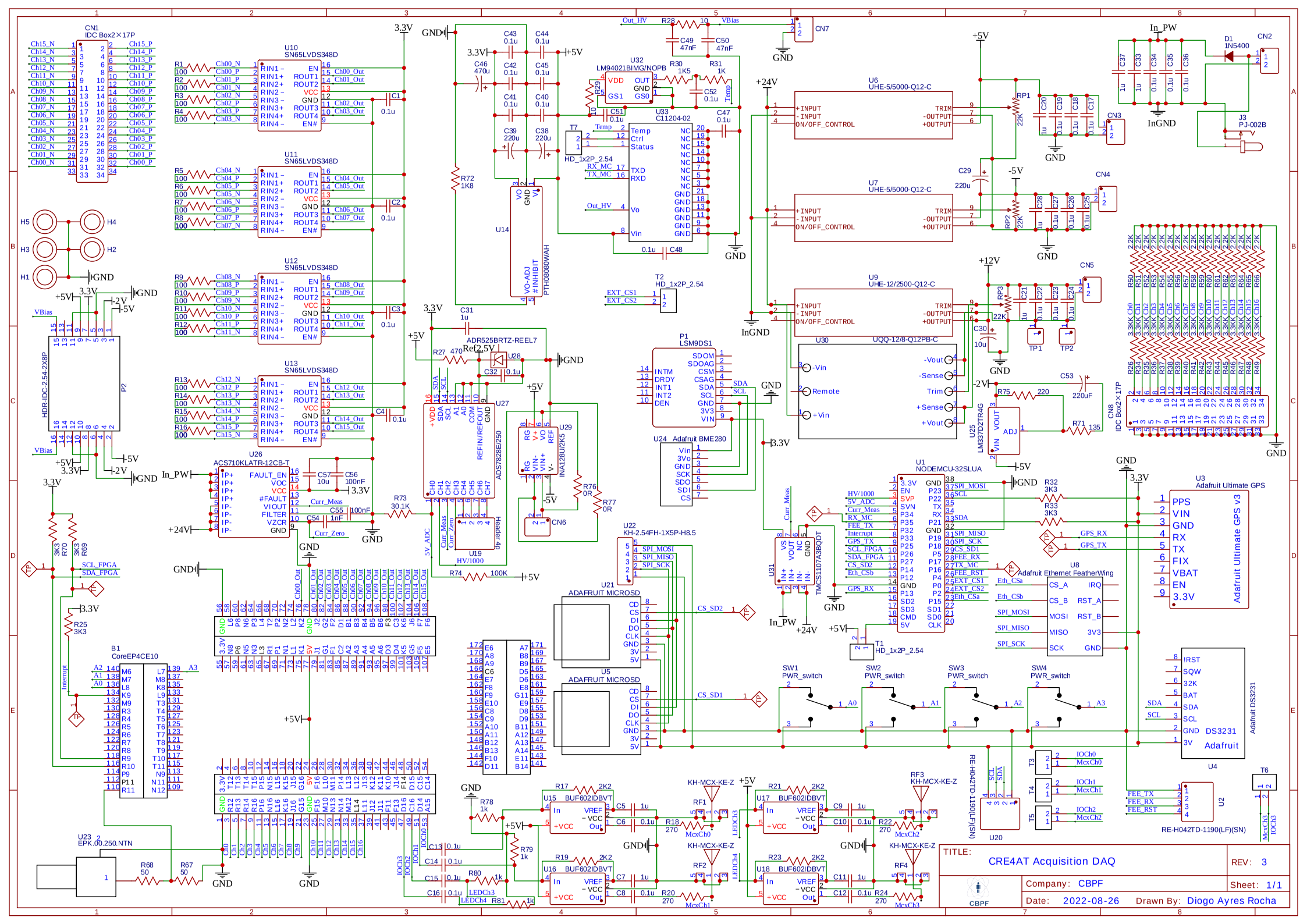
Este apêndice é composto pelo projeto da placa do DAQ, desenvolvido no *software EasyEDA*. O *layout* do DAQ foi feito em 6 camadas:

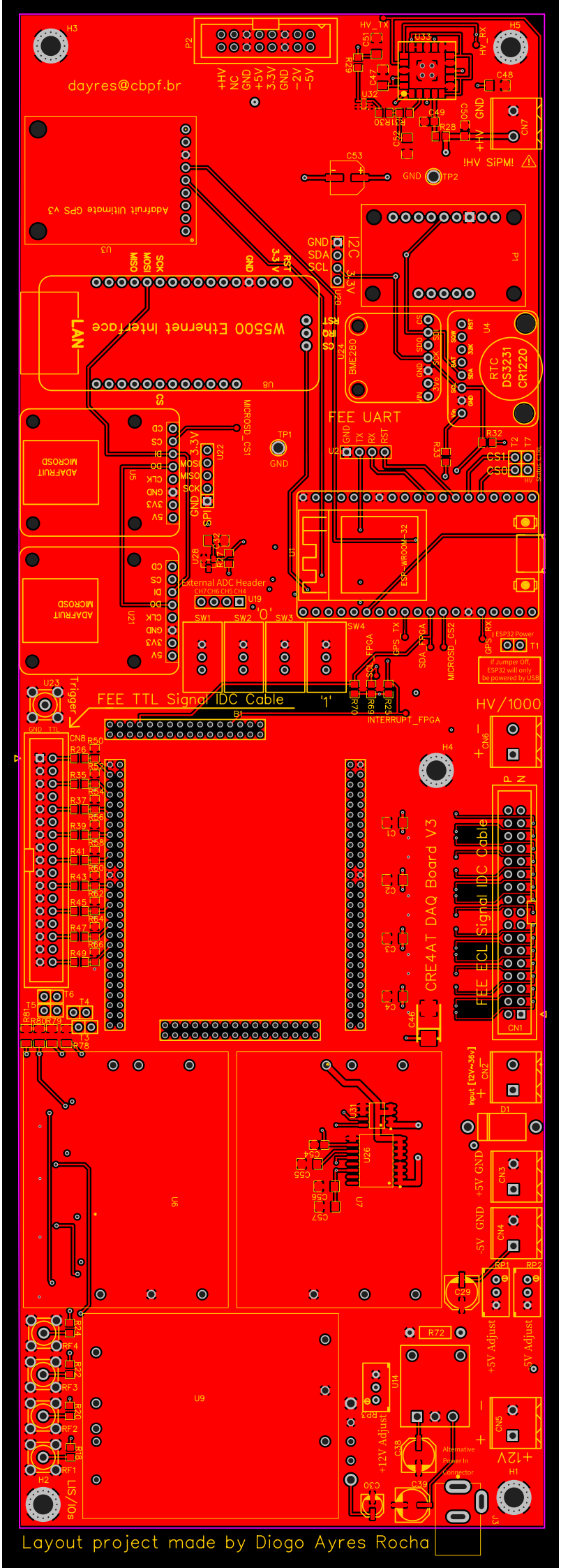
1. Camada de sinais superior;
2. Plano +5V;
3. Plano -5V;
4. Camada de sinais interna;
5. Plano +3.3V;
6. Camada de sinais inferior.

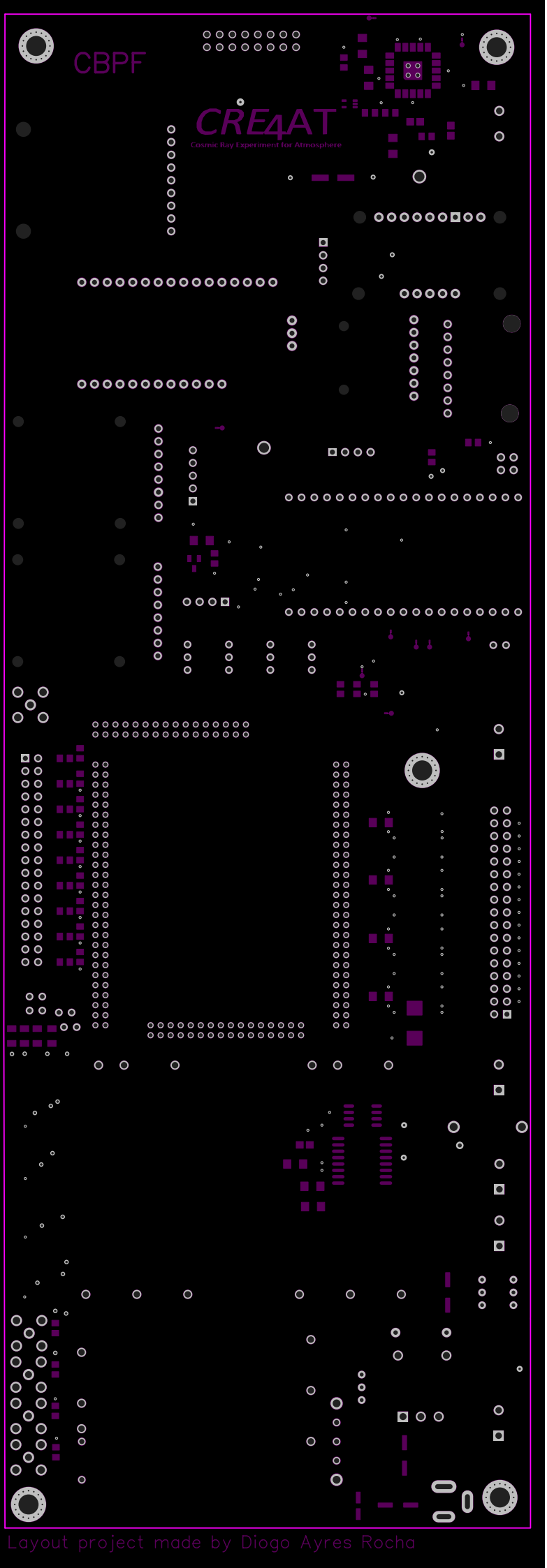
O projeto, neste apêndice, será mostrado na seguinte ordem:

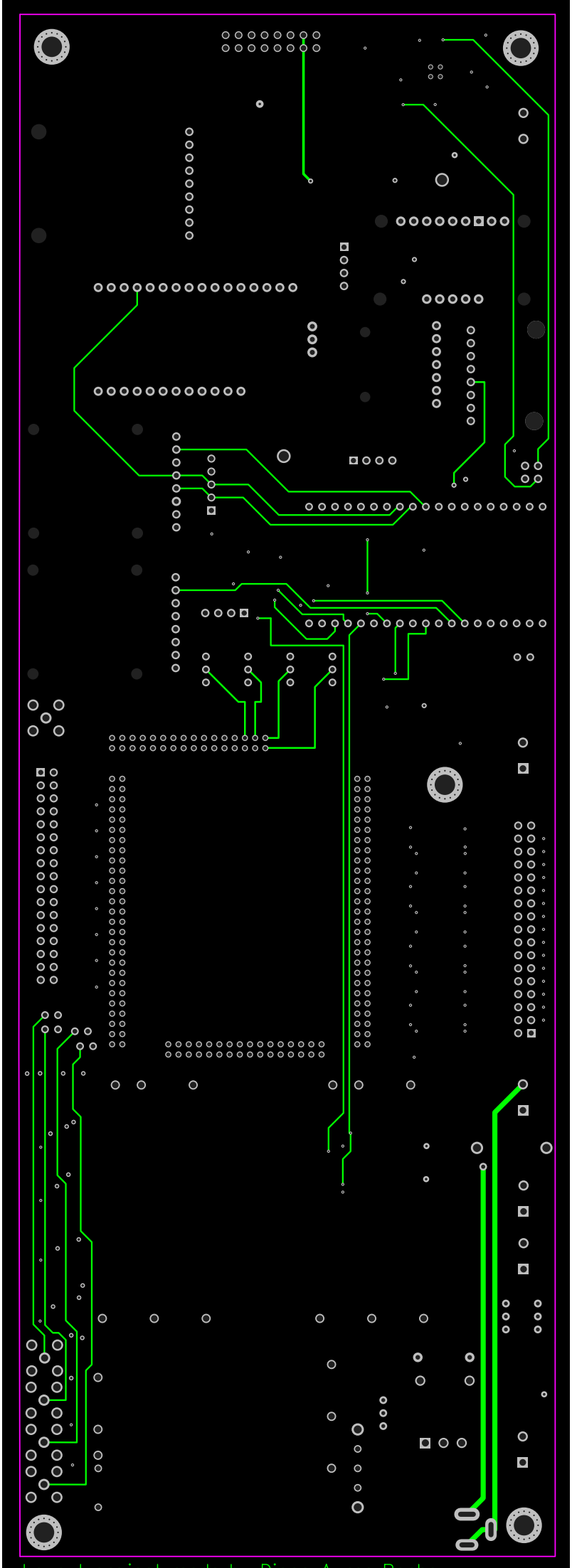
1. Esquemático da placa;
2. Layout camada de sinais superior;
3. Layout máscara de solda superior;
4. Layout camada de sinais interna;
5. Layout camada de sinais inferior;
6. Layout máscara de solda inferior.

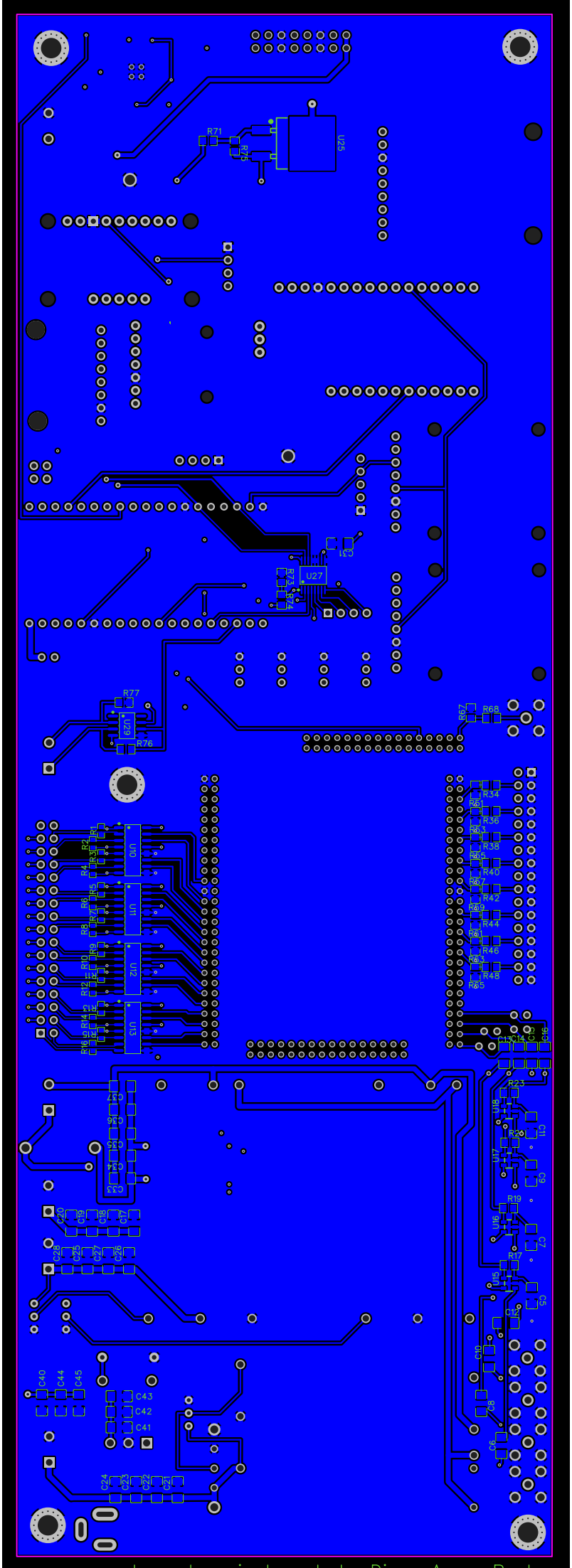
A camada inferior e a máscara de solda inferior foram espelhadas para melhor entendimento do projeto.

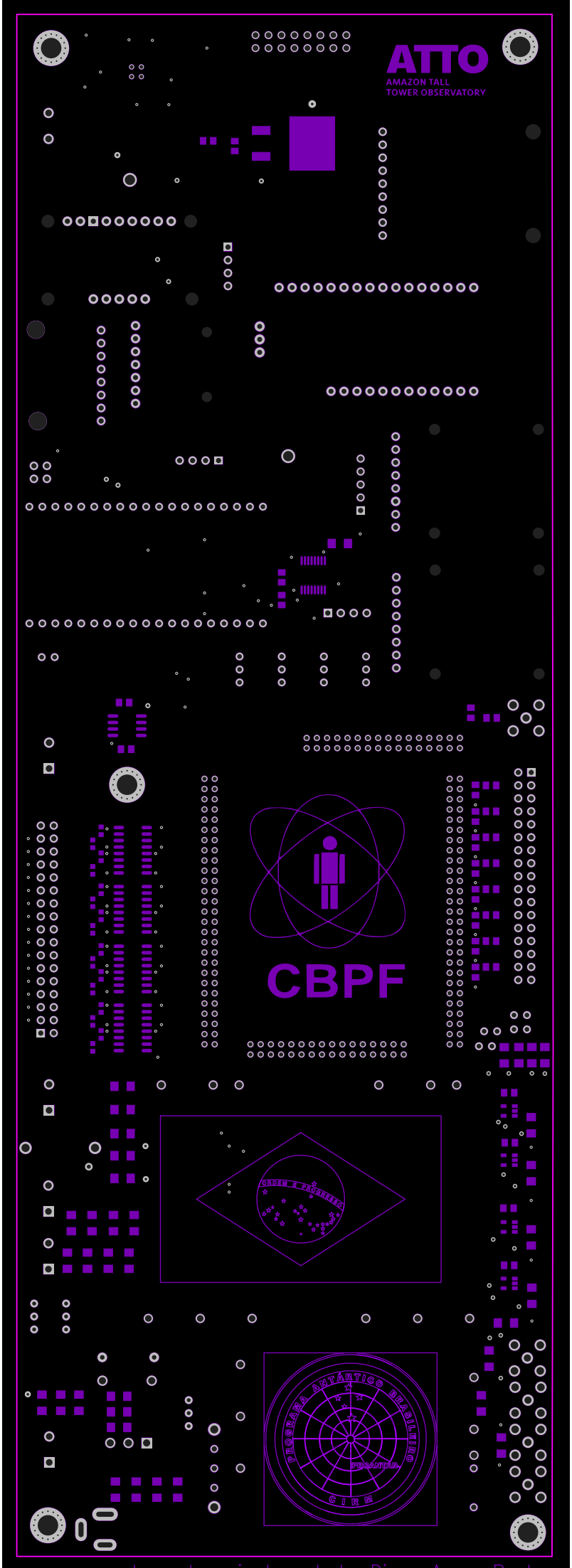

CN1
IDC Box2×17P
U10
SN65LVDS348D
U11
SN65LVDS348D
U12
SN65LVDS348D
U13
SN65LVDS348D
U14
PTH08080WAH
U26
ACS710KLATR-12CB-T
U27
ADS7828E/250
U28
ADR525BRTZ-REEL7
U29
INA128U/2K5
U32
LM94021BIMG/NOPB
U33
C11204-02
U6
UHE-5/5000-Q12-C
U7
UHE-5/5000-Q12-C
U9
UHE-12/2500-Q12-C
U30
UQQ-12/8-Q12PB-C
U24
Adafruit BME280
P1
LSM9DS1
U1
NODEMCU-32SLUA
U3
Adafruit Ultimate GPS
U8
Adafruit Ethernet FeatherWing
U21
ADAFRUIT MICROSD
U5
ADAFRUIT MICROSD
U22
KH-2.54FH-1X5P-H8.5
U31
TMCS1107A3BQDT
U25
LM337D2TR4G
U23
EPK.00.250.NTN
B1
CoreEP4CE10
U4
DS3231 Adafruit
U20
RE-H042TD-1190(LF)(SN)
CN2
CN3
CN4
CN5
CN6
CN7
CN8
IDC Box2×17P
J3
PJ-002B
D1
1N5400
T1
T2
T3
T4
T5
T6
SW1
PWR_switch
SW2
PWR_switch
SW3
PWR_switch
SW4
PWR_switch
TITLE: CRE4AT Acquisition DAQ
REV: 3
Company: CBPF
Sheet: 1/1
Date: 2022-08-26
Drawn By: Diogo Ayres Rocha
CBPF


dayres@cbpf.br
CRE4AT DAQ Board V3
FEE ECL Signal IDC Cable
FEE TTL Signal IDC Cable
FEE UART
HV/1000
HV SiPM!
W5500 Ethernet Interface
Adafruit Ultimate GPS v3
RTC DS3231 CR1220
BME280
ESP-WROOM-32
External ADC Header
Trigger
INTERRUPT_FPGA
Layout project made by Diogo Ayres Rocha


CBPF
CRE4AT
Cosmic Ray Experiment for Atmosphere
Layout project made by Diogo Ayres Rocha


Layout project made by Diogo Ayres Rocha


Layout project made by Diogo Ayres Rocha



Layout project made by Diogo Ayres Rocha

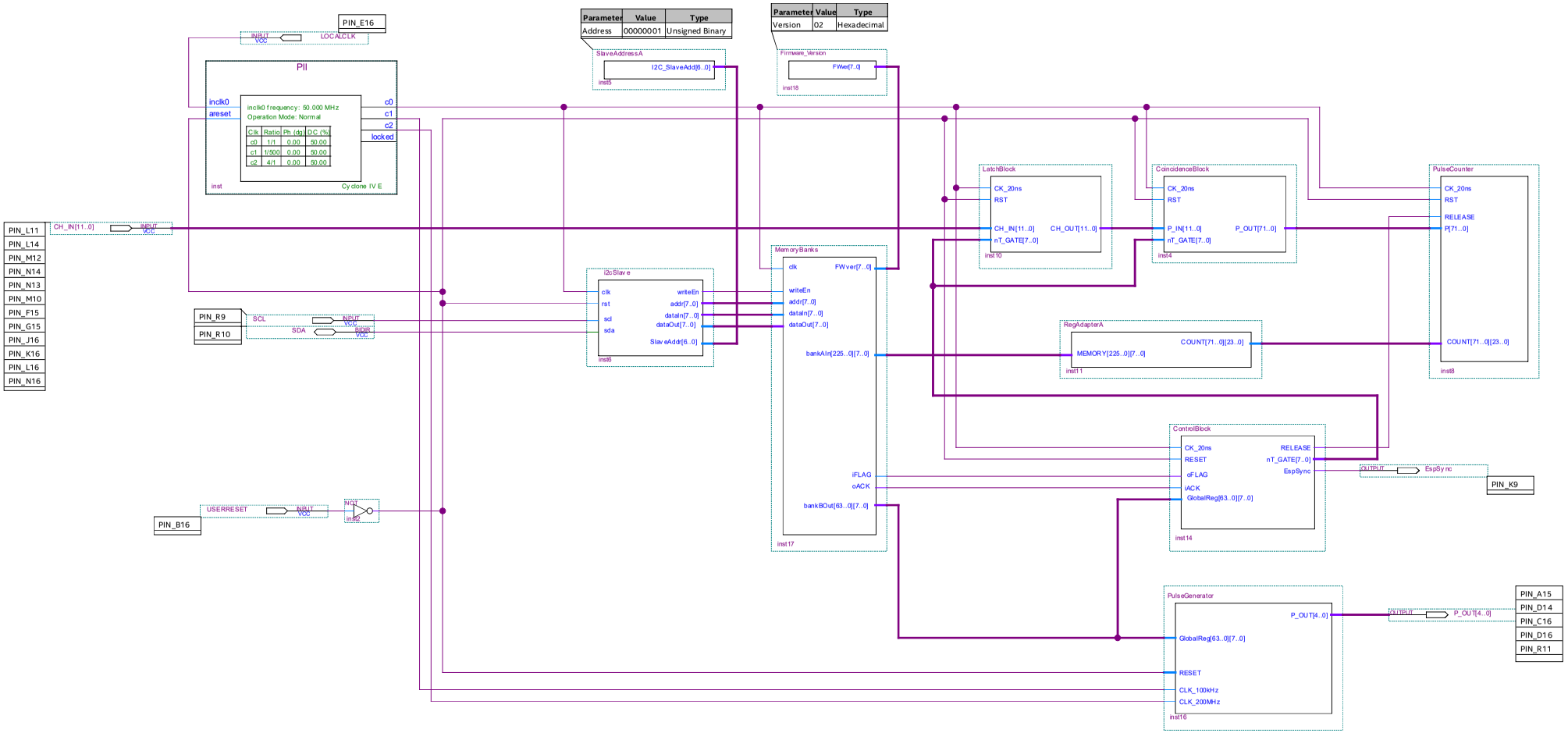
Apêndice F

Projeto da firmware do FPGA do DAQ

Este apêndice é composto pelo projeto do *firmware* do FPGA da placa do DAQ. O *firmware* foi desenvolvido, utilizando-se o software Quartus Prime 18.1 Lite Edition.

Os seguintes arquivos são necessários para a correta compilação deste projeto e serão inseridos nessa ordem:

1. FW_CREAT.bdf [Top Level];
2. ClkGen.vhd;
3. CoincidenceBlock.vhd;
4. ControlBlock.vhd;
5. CREATLibrary.vhd;
6. Firmware_Version.vhd;
7. FW_CREAT.qsf;
8. GroupCoincidence.vhd;
9. LatchBlock.vhd;
10. LEDClksGen.vhd;
11. MemoryBanks.vhd;
12. MemoryLibrary.vhd;
13. Pll.vhd;
14. PulseCounter.vhd;
15. PulseGenerator.vhd;
16. RegAdapterA.vhd;
17. SlaveAddressA;
18. WidthLEDControl;
19. i2cSlave/i2cSlave.v;
20. i2cSlave/i2cSlave_define.v;
21. i2cSlave/i2cSlaveTop.v;
22. i2cSlave/registerInterface.v;
23. i2cSlave/serialInterface.v;
24. i2cSlave/timescale.v.


PIN_E16
INPUT
VCC
LOCALCLK
PII
inclk0
areset
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk | Ratio | Ph (dg) | DC (%)
c0 | 1/1 | 0.00 | 50.00
c1 | 1/500 | 0.00 | 50.00
c2 | 4/1 | 0.00 | 50.00
c0
c1
c2
locked
inst
Cyclone IV E
Parameter | Value | Type
Address | 00000001 | Unsigned Binary
SlaveAddressA
I2C_SlaveAdd[6..0]
inst5
Parameter | Value | Type
Version | 02 | Hexadecimal
Firmware_Version
FWer[7..0]
inst18
PIN_L11
PIN_L14
PIN_M12
PIN_N14
PIN_N13
PIN_M10
PIN_F15
PIN_G15
PIN_J16
PIN_K16
PIN_L16
PIN_N16
CH_IN[11..0]
INPUT
VCC
PIN_R9
PIN_R10
SCL
SDA
INOUT
PIN_B16
USERRESET
NOT
inst2
i2cSlave
clk
rst
scl
sda
writeEn
addr[7..0]
dataIn[7..0]
dataOut[7..0]
SlaveAddr[6..0]
inst6
MemoryBanks
clk
writeEn
addr[7..0]
dataIn[7..0]
dataOut[7..0]
FWver[7..0]
bankAIn[225..0][7..0]
iFLAG
oACK
bankBOut[63..0][7..0]
inst17
LatchBlock
CK_20ns
RST
CH_IN[11..0]
nT_GATE[7..0]
CH_OUT[11..0]
inst10
CoincidenceBlock
CK_20ns
RST
P_IN[11..0]
nT_GATE[7..0]
P_OUT[71..0]
inst4
PulseCounter
CK_20ns
RST
RELEASE
P[71..0]
COUNT[71..0][23..0]
inst8
RegAdapterA
COUNT[71..0][23..0]
MEMORY[225..0][7..0]
inst11
ControlBlock
CK_20ns
RESET
oFLAG
iACK
GlobalReg[63..0][7..0]
RELEASE
nT_GATE[7..0]
EspSync
inst14
OUTPUT
EspSync
PIN_K9
PulseGenerator
P_OUT[4..0]
GlobalReg[63..0][7..0]
RESET
CLK_100kHz
CLK_200MHz
inst16
OUTPUT
P_OUT[4..0]
PIN_A15
PIN_D14
PIN_C16
PIN_D16
PIN_R11

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              ClkGen
-- Description:         CRE4AT 1Hz Clock Generation
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author             Rev.    Comments
-- 14 May 22  Diogo Ayres Rocha  0001    File Created
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity ClkGen is
port(
      CK_20ns     : in  std_logic;
      RST           : in  std_logic;
      CKOUT_1s    : out std_logic
);
end ClkGen;

architecture fClkGen of ClkGen is
      signal CK_1s  : std_logic := '0';
begin
      Clock1s : process(CK_20ns)
            variable count_20ns : integer range 0 to 25000000 := 0;
      begin
            if RST = '1' then
                  CK_1s       <= '0';
                  count_20ns :=  0 ;
            elsif rising_edge(CK_20ns) then
                  if count_20ns >= 24999999 then
                        CK_1s     <= not CK_1s;
                        count_20ns := 0;
                  else
                        count_20ns := count_20ns + 1;
                  end if;
            end if;
      end process;
      CKOUT_1s  <= CK_1s;
end fClkGen;
```

ClkGen.vhd

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              CoincidenceBlock
-- Description:         CRE4AT Groups Coincidence
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author              Rev.   Comments
-- 14 May 22  Diogo Ayres Rocha   0001   File Created
-- 09 Ago 23  Diogo Ayres Rocha   0002   Add Coincidences Between
Groups
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity CoincidenceBlock is
port(
      CK_20ns : in  std_logic;
      RST     : in  std_logic;
      nT_GATE : in  unsigned         ( 7 downto 0);
      P_IN    : in  std_logic_vector (11 downto 0);
      P_OUT   : out std_logic_vector (71 downto 0)
);
end CoincidenceBlock;

architecture fCoincidenceBlock of CoincidenceBlock is

      component GroupCoincidence is
      port(
            CK_20ns : in std_logic;
            RST     : in std_logic;
            nT_GATE : in unsigned          (7  downto 0);
            P       : in  std_logic_vector (3  downto 0);
            CoGroup : out std_logic_vector (10 downto 0)
      );
      end component;

      signal APulseIn   : std_logic_vector (3  downto 0);
      signal AGroupOut  : std_logic_vector (10 downto 0);
      signal CoinA01    : std_logic;
      signal CoinA12    : std_logic;
      signal CoinA23    : std_logic;

      signal BPulseIn   : std_logic_vector (3  downto 0);
      signal BGroupOut  : std_logic_vector (10 downto 0);
      signal CoinB01    : std_logic;
      signal CoinB12    : std_logic;
      signal CoinB23    : std_logic;

      signal CPulseIn   : std_logic_vector (3  downto 0);
      signal CGroupOut  : std_logic_vector (10 downto 0);
```

CoincidenceBlock.vhd

```
        signal CoinC01    : std_logic;
        signal CoinC12    : std_logic;
        signal CoinC23    : std_logic;

        signal DPulseIn   : std_logic_vector (3  downto 0);
        signal DGroupOut  : std_logic_vector (10 downto 0);
        signal CoinD01    : std_logic;
        signal CoinD12    : std_logic;
        signal CoinD23    : std_logic;

        signal nFindGroups: std_logic_vector (26 downto 0);


begin

        APulseIn(3 downto 0) <= P_IN( 3 downto 0);
        BPulseIn(3 downto 0) <= P_IN( 7 downto 4);
        CPulseIn(3 downto 0) <= P_IN(11 downto 8);

        AGroup : GroupCoincidence port map (CK_20ns, RST, nT_GATE,
APulseIn, AGroupOut);
        BGroup : GroupCoincidence port map (CK_20ns, RST, nT_GATE,
BPulseIn, BGroupOut);
        CGroup : GroupCoincidence port map (CK_20ns, RST, nT_GATE,
CPulseIn, CGroupOut);

        CoinA01 <= AGroupOut(0);
        CoinA12 <= AGroupOut(3);
        CoinA23 <= AGroupOut(5);
        CoinB01 <= BGroupOut(0);
        CoinB12 <= BGroupOut(3);
        CoinB23 <= BGroupOut(5);
        CoinC01 <= CGroupOut(0);
        CoinC12 <= CGroupOut(3);
        CoinC23 <= CGroupOut(5);

        nFindGroups(00) <= CoinA01 and CoinB01;
        nFindGroups(01) <= CoinA01 and CoinB12;
        nFindGroups(02) <= CoinA01 and CoinB23;
        nFindGroups(03) <= CoinA12 and CoinB01;
        nFindGroups(04) <= CoinA12 and CoinB12;
        nFindGroups(05) <= CoinA12 and CoinB23;
        nFindGroups(06) <= CoinA23 and CoinB01;
        nFindGroups(07) <= CoinA23 and CoinB12;
        nFindGroups(08) <= CoinA23 and CoinB23;
        nFindGroups(09) <= CoinA01 and CoinC01;
        nFindGroups(10) <= CoinA01 and CoinC12;
        nFindGroups(11) <= CoinA01 and CoinC23;
        nFindGroups(12) <= CoinA12 and CoinC01;
        nFindGroups(13) <= CoinA12 and CoinC12;
        nFindGroups(14) <= CoinA12 and CoinC23;
        nFindGroups(15) <= CoinA23 and CoinC01;
        nFindGroups(16) <= CoinA23 and CoinC12;
        nFindGroups(17) <= CoinA23 and CoinC23;
        nFindGroups(18) <= CoinB01 and CoinC01;
        nFindGroups(19) <= CoinB01 and CoinC12;
        nFindGroups(20) <= CoinB01 and CoinC23;
        nFindGroups(21) <= CoinB12 and CoinC01;
        nFindGroups(22) <= CoinB12 and CoinC12;
        nFindGroups(23) <= CoinB12 and CoinC23;
        nFindGroups(24) <= CoinB23 and CoinC01;
```

CoincidenceBlock.vhd

```
            nFindGroups(25) <= CoinB23 and CoinC12;
            nFindGroups(26) <= CoinB23 and CoinC23;


            P_OUT(11 downto  0)     <= P_IN(11 downto 0);
            P_OUT(22 downto 12)     <= AGRoupOut(10 downto 0);
            P_OUT(33 downto 23)     <= BGRoupOut(10 downto 0);
            P_OUT(44 downto 34)     <= CGRoupOut(10 downto 0);
            P_OUT(71 downto 45)     <= nFindGroups(26 downto 0);


end fCoincidenceBlock;
```

CoincidenceBlock.vhd

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              ControlBlock
-- Description:         CRE4AT Counters Time Control
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author              Rev.    Comments
-- 14 May 22  Diogo Ayres Rocha   0001    File Created
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

LIBRARY work;
USE work.my_types_pkg.all;
USE work.reg_addresses.all;
USE work.pulse_count_pkg.all;

entity ControlBlock is
port(
      CK_20ns     : in  std_logic;
      RESET       : in  std_logic;
      GlobalReg   : in  ByteArray(63 downto 0);
      RELEASE     : out std_logic;
      EspSync     : buffer std_logic;
      iACK        : in  std_logic;
      oFLAG       : out std_logic;
      nT_GATE     : out unsigned (7 downto 0)
);
end ControlBlock;

architecture fControlBlock of ControlBlock is

      component ClkGen is
      port(
            CK_20ns           : in  std_logic;
            RST               : in  std_logic;
            CKOUT_1s    : out std_logic
      );
      end component;

      signal CK_1s        : std_logic;
      signal TAQ          : unsigned (15 downto 0);
      signal sRELEASE     : std_logic := '0';
      signal sRELEASEprev : std_logic := '0';
      signal FLAG           : std_logic := '0';
      signal ACK          : std_logic := '0';
      signal configFlag   : std_logic := '0';
      -- !!! switches must be started inverted !!!
      signal switchA      : std_logic := '1';
      signal switchB      : std_logic := '0';
```

ControlBlock.vhd

```
begin

	RELEASE    <= sRELEASE;
	oFLAG      <= FLAG;
	clockModule : ClkGen port map (CK_20ns, RESET, CK_1s);
	sRELEASE   <= switchA xor switchB;
	ACK        <= iACK;

	BlockTiming : process(CK_1s, RESET)
		variable time_sec : unsigned (15 downto 0) := X"0000";
	begin
		if RESET = '1' then
			switchA  <= '1';
			time_sec :=  X"0000";
		elsif rising_edge(CK_1s) then
			time_sec := time_sec + 1;
			if time_sec >= TAQ then
				switchA  <= not switchA;
				time_sec := X"0000";
			end if;
		end if;
	end process;

	ReleasePullup : process (CK_20ns, sRELEASE, RESET)
	begin
		if RESET = '1' then
			switchB <= '0';
		elsif rising_edge(CK_20ns) then
			if sRELEASE = '0' then
				switchB <= not switchB;
			end if;
		end if;
	end process;

	UpdateConfig : process (CK_20ns, sRELEASE, RESET)
	begin
		if rising_edge(CK_20ns) then
			nT_GATE                  <= GlobalReg(nT_GATEAddr);
			TAQ(15 downto 8) <= GlobalReg(TAQAddr);
			TAQ( 7 downto 0) <= GlobalReg(TAQAddr + 1);
		end if;
	end process;

	TriggerConfig : process(CK_20ns)
	begin
		if rising_edge(CK_20ns) then
			if sRELEASEprev = '0' and sRELEASE = '1' then
				configFlag <= '1';
			else
				configFlag <= '0';
			end if;
		end if;
	end process;

	UpdateFlag : process(CK_20ns, sRELEASE, RESET, ACK)
	variable count : integer range 0 to 30 := 0;
	begin
		if RESET = '1' or ACK = '1' then
			FLAG <= '0';
			EspSync <= 'Z';
```

ControlBlock.vhd

```
                count := 0;
            elsif sRELEASEprev = '0' and sRELEASE = '1' then
                EspSync <= '0';
                FLAG <= '1';
            elsif rising_edge(CK_20ns) then
                if count >= 25 then
                    EspSync <= 'Z';
                    count := 0;
                elsif EspSync = '0' then
                    count := count + 1;
                end if;
            end if;
        end process;

        UpdatePrev : process(RESET, CK_20ns)
        begin
            if RESET = '1' then
                sRELEASEprev <= '1';
            elsif rising_edge(CK_20ns) then
                sRELEASEprev <= sRELEASE;
            end if;
        end process;

end fControlBlock;
```

ControlBlock.vhd

```vhdl
-- *******************************************************************
-- Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:            CRE4AT
-- Model:              CRE4AT DAQ V3
-- FPGA Proj. Name:    FW_CREAT
-- Device:             ALTERA EP4CE10F17C8
-- Author:             Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:               14-05-2022
-- -------------------------------------------------------------------
-- Module:             CRE4ATLibrary
-- Description:        CRE4AT Vector types definition
-- *******************************************************************

-- ###################################################################
-- Revision History:
--   Date      Author             Rev.   Comments
-- 14 May 22  Diogo Ayres Rocha  0001   File Created
-- 09 Ago 23  Diogo Ayres Rocha  0002   Vectors Dimensions Change
-- ###################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

package pulse_count_pkg is

      -- LATCH_ASINC
      constant max_nT_gate : integer := 255; -- maximum number of
clock periods (50MHz) in one latched pulse (1 Byte count)
      type     int8b_array is array (natural range <>) of integer
range 0 to max_nT_gate;
      -- PULSE_COUNT
      constant reg16b_range : integer :=    65535;
      constant vec24b_zero  : unsigned (23 downto 0) := (others =>
'0');
      constant reg24b_range : integer := 16777215; -- maximum number
to which a 24-bit register can count
      type vec24b is array (natural range <>) of unsigned (23 downto
0);
      type int24b is array (natural range <>) of integer range 0 to
reg24b_range;
      constant VEC24bx72_ZERO : vec24b(71 downto 0)   := (vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero, vec24b_zero,
vec24b_zero);

      constant INT24bx72_ZERO : int24b(71 downto 0)   := (0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);

end package;
```

```
-- WARNING: Do NOT edit the input and output ports in this file in a
text
-- editor if you plan to continue editing the block that represents it
in
-- the Block Editor! File corruption is VERY likely to occur.

-- Copyright (C) 2016  Intel Corporation. All rights reserved.
-- Your use of Intel Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- associated documentation or information are expressly subject
-- to the terms and conditions of the Intel Program License
-- Subscription Agreement, the Intel Quartus Prime License Agreement,
-- the Intel MegaCore Function License Agreement, or other
-- applicable license agreement, including, without limitation,
-- that your use is for the sole purpose of programming logic
-- devices manufactured by Intel and sold by Intel or its
-- authorized distributors.  Please refer to the applicable
-- agreement for further details.


-- Generated by Quartus Prime Version 16.1 (Build Build 196
10/24/2016)
-- Created on Thu May 04 15:58:21 2017

LIBRARY ieee;
USE ieee.std_logic_1164.all;


--  Entity Declaration

ENTITY Firmware_Version IS
-- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
GENERIC(Version : STD_LOGIC_VECTOR(7 DOWNTO 0) := X"03");
PORT
(
FWver : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
);
-- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

END Firmware_Version;


--  Architecture Body

ARCHITECTURE Firmware_Version_architecture OF Firmware_Version IS


BEGIN

FWver <= Version;

END Firmware_Version_architecture;
```

Firmware_Version.vhd

```
# -------------------------------------------------------------------------- #
#
# Copyright (C) 2018  Intel Corporation. All rights reserved.
# Your use of Intel Corporation's design tools, logic functions
# and other software and tools, and its AMPP partner logic
# functions, and any output files from any of the foregoing
# (including device programming or simulation files), and any
# associated documentation or information are expressly subject
# to the terms and conditions of the Intel Program License
# Subscription Agreement, the Intel Quartus Prime License Agreement,
# the Intel FPGA IP License Agreement, or other applicable license
# agreement, including, without limitation, that your use is for
# the sole purpose of programming logic devices manufactured by
# Intel and sold by Intel or its authorized distributors.  Please
# refer to the applicable agreement for further details.
#
# -------------------------------------------------------------------------- #
#
# Quartus Prime
# Version 18.1.0 Build 625 09/12/2018 SJ Lite Edition
# Date created = 22:02:32  November 19, 2020
#
# -------------------------------------------------------------------------- #
#
# Notes:
#
# 1) The default values for assignments are stored in the file:
#                FW_CREAT_assignment_defaults.qdf
#   If this file doesn't exist, see file:
#                assignment_defaults.qdf
#
# 2) Altera recommends that you do not modify this file. This
#   file is updated automatically by the Quartus Prime software
#   and any changes you make may be lost or overwritten.
#
# -------------------------------------------------------------------------- #


set_global_assignment -name FAMILY "Cyclone IV E"
set_global_assignment -name DEVICE EP4CE10F17C8
set_global_assignment -name TOP_LEVEL_ENTITY FW_CREAT
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 18.1.0
set_global_assignment -name PROJECT_CREATION_TIME_DATE "22:02:32  NOVEMBER 19, 2020"
set_global_assignment -name LAST_QUARTUS_VERSION "18.1.0 Lite Edition"
set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
set_global_assignment -name NOMINAL_CORE_SUPPLY_VOLTAGE 1.2V
set_global_assignment -name POWER_PRESET_COOLING_SOLUTION "23 MM HEAT SINK WITH 200 LFPM
AIRFLOW"
set_global_assignment -name POWER_BOARD_THERMAL_MODEL "NONE (CONSERVATIVE)"
```

FW_CREAT.qsf

```
set_location_assignment PIN_L11 -to CH_IN[0]
set_location_assignment PIN_L14 -to CH_IN[1]
set_location_assignment PIN_A15 -to P_OUT[0]
set_location_assignment PIN_D14 -to P_OUT[1]
set_location_assignment PIN_C16 -to P_OUT[2]
set_location_assignment PIN_D16 -to P_OUT[3]
set_location_assignment PIN_R10 -to SDA
set_location_assignment PIN_R9 -to SCL
set_location_assignment PIN_E16 -to LOCALCLK
set_location_assignment PIN_B16 -to USERRESET
set_global_assignment -name EDA_SIMULATION_TOOL "<None>"
set_global_assignment -name TIMING_ANALYZER_MULTICORNER_ANALYSIS ON
set_global_assignment -name SMART_RECOMPILE ON
set_global_assignment -name NUM_PARALLEL_PROCESSORS 2
set_global_assignment -name ALLOW_ANY_RAM_SIZE_FOR_RECOGNITION ON
set_global_assignment -name ALLOW_ANY_ROM_SIZE_FOR_RECOGNITION ON
set_global_assignment -name ALLOW_ANY_SHIFT_REGISTER_SIZE_FOR_RECOGNITION ON
set_location_assignment PIN_K9 -to EspSync
set_global_assignment -name ENABLE_SIGNALTAP ON
set_global_assignment -name USE_SIGNALTAP_FILE output_files/stp1.stp
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[11]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[10]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[9]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[8]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[7]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[6]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[5]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[4]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[3]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[2]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[1]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[0]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LOCALCLK
set_instance_assignment -name IO_STANDARD "2.5 V" -to P_OUT[3]
set_instance_assignment -name IO_STANDARD "2.5 V" -to P_OUT[2]
set_instance_assignment -name IO_STANDARD "2.5 V" -to P_OUT[1]
set_instance_assignment -name IO_STANDARD "2.5 V" -to P_OUT[0]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to SCL
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to SDA
set_instance_assignment -name WEAK_PULL_UP_RESISTOR ON -to SDA
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to USERRESET
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to EspSync
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[15]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[14]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[13]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN[12]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CH_IN
set_location_assignment PIN_M12 -to CH_IN[2]
set_location_assignment PIN_N14 -to CH_IN[3]
set_location_assignment PIN_N13 -to CH_IN[4]
set_location_assignment PIN_M10 -to CH_IN[5]
set_location_assignment PIN_F15 -to CH_IN[6]
```

FW_CREAT.qsf

```
set_location_assignment PIN_G15 -to CH_IN[7]
set_location_assignment PIN_J16 -to CH_IN[8]
set_location_assignment PIN_K16 -to CH_IN[9]
set_location_assignment PIN_L16 -to CH_IN[10]
set_location_assignment PIN_N16 -to CH_IN[11]
set_location_assignment PIN_P16 -to CH_IN[12]
set_location_assignment PIN_R16 -to CH_IN[13]
set_location_assignment PIN_R14 -to CH_IN[14]
set_location_assignment PIN_R13 -to CH_IN[15]
set_global_assignment -name VERILOG_FILE i2cSlave/timescale.v
set_global_assignment -name VERILOG_FILE i2cSlave/serialInterface.v
set_global_assignment -name VERILOG_FILE i2cSlave/i2cSlave_define.v
set_global_assignment -name VERILOG_FILE i2cSlave/i2cSlave.v
set_global_assignment -name SOURCE_FILE Pll.cmp
set_global_assignment -name VHDL_FILE MemoryLibrary.vhd
set_global_assignment -name VHDL_FILE PulseGenerator.vhd
set_global_assignment -name VHDL_FILE SlaveAddressA.vhd
set_global_assignment -name VHDL_FILE RegAdapterA.vhd
set_global_assignment -name BDF_FILE FW_CREAT.bdf
set_global_assignment -name VHDL_FILE Firmware_Version.vhd
set_global_assignment -name VHDL_FILE CREATLibrary.vhd
set_global_assignment -name VHDL_FILE CoincidenceBlock.vhd
set_global_assignment -name VHDL_FILE ClkGen.vhd
set_global_assignment -name VHDL_FILE RegAdapterB.vhd
set_global_assignment -name VHDL_FILE LedTestOutput.vhd
set_global_assignment -name VHDL_FILE MemoryBanks.vhd
set_global_assignment -name VHDL_FILE GroupCoincidence.vhd
set_global_assignment -name VHDL_FILE ControlBlock.vhd
set_global_assignment -name QIP_FILE Pll.qip
set_global_assignment -name VHDL_FILE ClkGen2.vhd
set_global_assignment -name VHDL_FILE PulseCounter.vhd
set_global_assignment -name BSF_FILE CoincidenceBlock.bsf
set_global_assignment -name VHDL_FILE LatchBlock.vhd
set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -
section_id Top
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
set_location_assignment PIN_R11 -to P_OUT[4]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to P_OUT[4]
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
```

FW_CREAT.qsf

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              GroupCoincidence
-- Description:         CRE4AT Inside Group Coincidences
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author              Rev.    Comments
-- 14 May 22  Diogo Ayres Rocha   0001    File Created
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity GroupCoincidence is
port(
      CK_20ns : in std_logic;
      RST     : in std_logic;
      nT_GATE : in unsigned          (7  downto 0);
      P       : in  std_logic_vector (3  downto 0);
      CoGroup : out std_logic_vector (10 downto 0)
);
end GroupCoincidence;

architecture fGroupCoincidence of GroupCoincidence is

begin

      CoGroup(0)  <= P(0) and P(1);
      CoGroup(1)  <= P(0) and P(2);
      CoGroup(2)  <= P(0) and P(3);
      CoGroup(3)  <= P(1) and P(2);
      CoGroup(4)  <= P(1) and P(3);
      CoGroup(5)  <= P(2) and P(3);
      CoGroup(6)  <= P(0) and P(1) and P(2);
      CoGroup(7)  <= P(0) and P(1) and P(3);
      CoGroup(8)  <= P(0) and P(2) and P(3);
      CoGroup(9)  <= P(1) and P(2) and P(3);
      CoGroup(10) <= P(0) and P(1) and P(2) and P(3);


end fGroupCoincidence;
```

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              LatchBlock
-- Description:         Define pulse width control for all channels
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author             Rev.   Comments
-- 14 May 22  Diogo Ayres Rocha  0001   File Created
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

LIBRARY work;
USE work.pulse_count_pkg.all;

entity LatchBlock is
port(
      CK_20ns : in  std_logic;
      RST     : in  std_logic;
      nT_GATE : in  unsigned              ( 7 downto 0);
      CH_IN   : in  std_logic_vector (11 downto 0);
      CH_OUT  : out std_logic_vector (11 downto 0)
);
end LatchBlock;

architecture fLatchBlock of LatchBlock is
      -- !!! CH_SWT and LT_SWT must necessarily be initialyzed with
inverted corresponding bits !!!
      signal CH_SWT : std_logic_vector (11 downto 0) :=
"111111111111";
      signal LT_SWT : std_logic_vector (11 downto 0) :=
"000000000000";
      signal CH_LAT : std_logic_vector (11 downto 0) :=
"000000000000";
      signal SET_LT : std_logic_vector (11 downto 0) :=
"000000000000";

begin

Ch0_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
      if RST = '1' then
            CH_SWT(0) <= '1';
      else
            if rising_edge(CH_IN(0)) and SET_LT(0) = '0' and CH_LAT(0)
= '0' then
                  CH_SWT(0) <= not CH_SWT(0);
            end if;
      end if;
end process;
```

LatchBlock.vhd

```
Ch1_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(1) <= '1';
	else
		if rising_edge(CH_IN(1)) and SET_LT(1) = '0' and CH_LAT(1)
= '0' then
			CH_SWT(1) <= not CH_SWT(1);
		end if;
	end if;
end process;

Ch2_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(2) <= '1';
	else
		if rising_edge(CH_IN(2)) and SET_LT(2) = '0' and CH_LAT(2)
= '0' then
			CH_SWT(2) <= not CH_SWT(2);
		end if;
	end if;
end process;

Ch3_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(3) <= '1';
	else
		if rising_edge(CH_IN(3)) and SET_LT(3) = '0' and CH_LAT(3)
= '0' then
			CH_SWT(3) <= not CH_SWT(3);
		end if;
	end if;
end process;

Ch4_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(4) <= '1';
	else
		if rising_edge(CH_IN(4)) and SET_LT(4) = '0' and CH_LAT(4)
= '0' then
			CH_SWT(4) <= not CH_SWT(4);
		end if;
	end if;
end process;

Ch5_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(5) <= '1';
	else
		if rising_edge(CH_IN(5)) and SET_LT(5) = '0' and CH_LAT(5)
= '0' then
			CH_SWT(5) <= not CH_SWT(5);
		end if;
	end if;
end process;
```

```
Ch6_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(6) <= '1';
	else
		if rising_edge(CH_IN(6)) and SET_LT(6) = '0' and CH_LAT(6)
= '0' then
			CH_SWT(6) <= not CH_SWT(6);
		end if;
	end if;
end process;

Ch7_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(7) <= '1';
	else
		if rising_edge(CH_IN(7)) and SET_LT(7) = '0' and CH_LAT(7)
= '0' then
			CH_SWT(7) <= not CH_SWT(7);
		end if;
	end if;
end process;

Ch8_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(8) <= '1';
	else
		if rising_edge(CH_IN(8)) and SET_LT(8) = '0' and CH_LAT(8)
= '0' then
			CH_SWT(8) <= not CH_SWT(8);
		end if;
	end if;
end process;

Ch9_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(9) <= '1';
	else
		if rising_edge(CH_IN(9)) and SET_LT(9) = '0' and CH_LAT(9)
= '0' then
			CH_SWT(9) <= not CH_SWT(9);
		end if;
	end if;
end process;

Ch10_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
	if RST = '1' then
		CH_SWT(10) <= '1';
	else
		if rising_edge(CH_IN(10)) and SET_LT(10) = '0' and
CH_LAT(10) = '0' then
			CH_SWT(10) <= not CH_SWT(10);
		end if;
	end if;
end process;

Ch11_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
```

```
begin
	if RST = '1' then
		CH_SWT(11) <= '1';
	else
		if rising_edge(CH_IN(11)) and SET_LT(11) = '0' and
CH_LAT(11) = '0' then
			CH_SWT(11) <= not CH_SWT(11);
		end if;
	end if;
end process;

Latch : process(CK_20ns, RST, SET_LT, CH_LAT)
	variable num_ticks : int8b_array(11 downto 0) := (0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0);
begin
	if RST = '1' then
		LT_SWT    <= "000000000000";
		CH_LAT    <= "000000000000";
		num_ticks := (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);

	elsif falling_edge(CK_20ns) then
		for i in 0 to 11 loop
			if SET_LT(i) = '1' then
				LT_SWT(i)    <= not LT_SWT(i);
				CH_LAT(i)    <= '1';
				num_ticks(i) :=  0 ;

			elsif CH_LAT(i) = '1' then
				num_ticks(i) := num_ticks(i) + 1;

				if num_ticks(i) >= to_integer(nT_GATE) then
					CH_LAT(i)    <= '0';
					num_ticks(i) :=  0 ;
				end if;

			end if;
		end loop;
	end if;
end process;

	SET_LT <= CH_SWT xnor LT_SWT;
	CH_OUT <= CH_LAT;

end fLatchBlock;
```

LatchBlock.vhd

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                09-08-2023
-- ------------------------------------------------------------------
-- Module:              LEDClksGen
-- Description:         Generate LED Pulse Clocks from 100kHz in Clock
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author              Rev.   Comments
-- 09 Ago 23  Diogo Ayres Rocha   0001   File Created
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity LEDClksGen is
port(
     CLK_100kHz     : in  std_logic;
     RST              : in  std_logic;
     CLKOUT_50kHz   : out std_logic;
     CLKOUT_25kHz   : out std_logic;
     CLKOUT_10kHz   : out std_logic;
     CLKOUT_5kHz    : out std_logic;
     CLKOUT_1kHz    : out std_logic
);
end LEDClksGen;

architecture fLEDClksGen of LEDClksGen is
    signal Clk_50kHz   : std_logic := '0';
    signal Clk_25kHz   : std_logic := '0';
    signal Clk_10kHz   : std_logic := '0';
    signal Clk_5kHz    : std_logic := '0';
    signal Clk_1kHz    : std_logic := '0';
begin

Clock1s : process(CLK_100kHz, RST)
    variable count_50kHz : integer range 0 to 3    := 0;
    variable count_25kHz : integer range 0 to 2    := 0;
    variable count_10kHz : integer range 0 to 5    := 0;
    variable count_5kHz  : integer range 0 to 10   := 0;
    variable count_1kHz  : integer range 0 to 50   := 0;
begin
      if RST = '1' then
        Clk_50kHz   <= '0';
        Clk_25kHz   <= '0';
        Clk_10kHz   <= '0';
        Clk_5kHz    <= '0';
        Clk_1kHz    <= '0';
        count_25kHz :=  0 ;
        count_10kHz :=  0 ;
        count_5kHz  :=  0 ;
        count_1kHz  :=  0 ;
      elsif rising_edge(CLK_100kHz) then
```

LEDClksGen.vhd

```
            Clk_50kHz   <= not Clk_50kHz;
            if count_25kHz >= 1 then
                  Clk_25kHz   <= not Clk_25kHz;
                  count_25kHz := 0;
            else
                  count_25kHz := count_25kHz + 1;
            end if;
         if count_10kHz >= 4 then
                  Clk_10kHz   <= not Clk_10kHz;
                  count_10kHz := 0;
            else
                  count_10kHz := count_10kHz + 1;
            end if;
         if count_5kHz >= 9 then
                  Clk_5kHz   <= not Clk_5kHz;
                  count_5kHz := 0;
            else
                  count_5kHz := count_5kHz + 1;
            end if;
         if count_1kHz >= 49 then
                  Clk_1kHz   <= not Clk_1kHz;
                  count_1kHz := 0;
            else
                  count_1kHz := count_1kHz + 1;
            end if;
      end if;

end process;

    CLKOUT_50kHz <= Clk_50kHz;
    CLKOUT_25kHz <= Clk_25kHz;
    CLKOUT_10kHz <= Clk_10kHz;
    CLKOUT_5kHz  <= Clk_5kHz;
    CLKOUT_1kHz  <= Clk_1kHz;

end fLEDClksGen;
```

LEDClksGen.vhd

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              MemoryBanks
-- Description:         I2C Memories Control
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author              Rev.   Comments
-- 14 May 22  Diogo Ayres Rocha   0001   File Created
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

LIBRARY work;
USE work.my_types_pkg.all;
USE work.reg_addresses.all;

entity MemoryBanks is
port (
      clk        : in     std_logic;
      addr       : in     unsigned (7 downto 0);
      dataIn     : in     unsigned (7  downto 0);
      dataOut    : buffer unsigned (7  downto 0);
      writeEn    : in     std_logic;
      iFLAG      : in     std_logic;
      oACK       : out    std_logic;
      bankAIn    : in     ByteArray(225 downto 0);
      bankBOut   : out    ByteArray(63 downto 0);
      FWver      : in     unsigned (7   downto 0)
);
end entity;

architecture fMemoryBanks of MemoryBanks is

      signal bankA            : ByteArray(225 downto 0);
      signal bankB            : ByteArray(63 downto 0);
      signal iaddr            : integer range 0 to 255;
      signal FLAG           : std_logic := '0';
      signal ACK              : std_logic := '0';
      signal bankSelect : std_logic := '1'; --defaults to Bank B
      signal DAC_Rst          : std_logic := '1';

begin

      bankA     <= bankAIn;
      bankBOut  <= bankB;
      iaddr     <= to_integer(addr(7 downto 0));
      FLAG      <= iFLAG;
      oACK      <= ACK;

      ---------------------------------
```

MemoryBanks.vhd

```
---------- MEMORY MUX ----------
----------------------------------

BankSelection : process(clk)
begin
      if rising_edge(clk) then
            if writeEn = '1' and iaddr = bankSelectAddr then
                  bankSelect <= dataIn(0);
            end if;
      end if;
end process;

MemoryMux : process(clk)
begin
      if rising_edge(clk) then
            -- Common Addresses Reading
            if iaddr = FWverAddr then
                  dataOut <= FWver;

            elsif iaddr = bankSelectAddr then
                  dataOut <= "0000000" & bankSelect;

            elsif iaddr = FLAGAddr then
                  dataOut <= "0000000" & FLAG;

            elsif iaddr = ACKAddr then
                  dataOut <= "0000000" & ACK;

            -- Bank Addresses Reading
            elsif bankSelect = '0' then
                  dataOut <= bankA(iaddr);

            elsif bankSelect = '1' then
                  dataOut <= bankB(iaddr);
            end if;
      end if;
end process;

----------------------------------
--------- BANK B Write -----------
----------------------------------

WriteBankB : process(clk)
begin
      if rising_edge(clk) then
            if writeEn = '1' then
                  bankB(iaddr) <= dataIn;
            end if;
      end if;
end process;

----------------------------------
--------- Flag Function ----------
----------------------------------

FlagManaging : process(clk, FLAG)
begin
      if rising_edge(clk) then
            if writeEn = '1' and iaddr = ACKAddr then
                  ACK <= '1';
            elsif FLAG = '0' then
```

MemoryBanks.vhd

```
                        ACK <= '0';
                    end if;
                end if;
            end process;

end fMemoryBanks;
```

MemoryBanks.vhd

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              MemoryLibrary
-- Description:         CRE4AT I2C Memory Types and regs definition
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author              Rev.   Comments
-- 14 May 22  Diogo Ayres Rocha 0001   File Created
-- 09 Ago 23  Diogo Ayres Rocha 0002   File Changes LED Injector Regs
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

package my_types_pkg is
  type ByteArray is array (natural range <>) of unsigned(7 downto 0);
end package;

package reg_addresses is

    ---------------------------------------
    ---------- Common Addresses ----------
    ---------------------------------------
    constant ACKAddr          : integer := 251; -- Count Blocks
Retrieval Ack register
    constant FLAGAddr         : integer := 252; -- Count Blocks Ready
Flag register
    constant bankSelectAddr   : integer := 253; -- Memory Bank A or B
selection register
    constant softwareRstAddr  : integer := 254; -- Software Reset
    constant FWverAddr        : integer := 255; -- Firmware Version
    ---------------------------------------
    ------ Bank B (I2C Write enable) ------
    ---------------------------------------
    ---------------------------------------
    -- GlobalReg range 0 to 63
    ---------------------------------------
    constant GlobalRegAddr    : integer :=   0;
    -------- LEDInj range 0 to 10 ---------
    -------- ledEnableMask    addresses    0
    constant LED0InjFreqSel   : integer :=   1;
    constant LED1InjFreqSel   : integer :=   2;
    constant LED2InjFreqSel   : integer :=   3;
    constant LED3InjFreqSel   : integer :=   4;
    constant TriggerFreqSel   : integer :=   5;
    constant LED0InjWidthSel  : integer :=   6;
    constant LED1InjWidthSel  : integer :=   7;
    constant LED2InjWidthSel  : integer :=   8;
    constant LED3InjWidthSel  : integer :=   9;
    constant TriggerWidthSel  : integer :=   10;
    constant MinimumWidthSel  : integer :=   11;
```

```
    ----------------------------------------------------
    ----- CREAT PulseCount range 12 to 63 -----
    constant nT_GATEAddr      : integer :=  12;
    constant TAQAddr          : integer :=  13; --(13 and 14) 2 bytes
    --constant LedOutAddr     : integer :=  15;
    ----------------------------------------------------
end package;
```

```
-- megafunction wizard: %ALTPLL%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: altpll

-- ============================================================
-- File Name: Pll.vhd
-- Megafunction Name(s):
--               altpll
--
-- Simulation Library Files(s):
--               altera_mf
-- ============================================================
-- ************************************************************
-- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
--
-- 18.1.0 Build 625 09/12/2018 SJ Lite Edition
-- ************************************************************


--Copyright (C) 2018  Intel Corporation. All rights reserved.
--Your use of Intel Corporation's design tools, logic functions
--and other software and tools, and its AMPP partner logic
--functions, and any output files from any of the foregoing
--(including device programming or simulation files), and any
--associated documentation or information are expressly subject
--to the terms and conditions of the Intel Program License
--Subscription Agreement, the Intel Quartus Prime License Agreement,
--the Intel FPGA IP License Agreement, or other applicable license
--agreement, including, without limitation, that your use is for
--the sole purpose of programming logic devices manufactured by
--Intel and sold by Intel or its authorized distributors.  Please
--refer to the applicable agreement for further details.


LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.all;

ENTITY Pll IS
      PORT
      (
            areset            : IN STD_LOGIC  := '0';
            inclk0            : IN STD_LOGIC  := '0';
            c0          : OUT STD_LOGIC ;
            c1          : OUT STD_LOGIC ;
            c2          : OUT STD_LOGIC ;
            locked            : OUT STD_LOGIC
      );
END Pll;


ARCHITECTURE SYN OF pll IS

      SIGNAL sub_wire0  : STD_LOGIC ;
      SIGNAL sub_wire1  : STD_LOGIC_VECTOR (1 DOWNTO 0);
      SIGNAL sub_wire2_bv     : BIT_VECTOR (0 DOWNTO 0);
      SIGNAL sub_wire2  : STD_LOGIC_VECTOR (0 DOWNTO 0);
      SIGNAL sub_wire3  : STD_LOGIC_VECTOR (4 DOWNTO 0);
```

Pll.vhd

```
SIGNAL sub_wire4  : STD_LOGIC ;
SIGNAL sub_wire5  : STD_LOGIC ;
SIGNAL sub_wire6  : STD_LOGIC ;
SIGNAL sub_wire7  : STD_LOGIC ;


COMPONENT altpll
GENERIC (
      bandwidth_type          : STRING;
      clk0_divide_by          : NATURAL;
      clk0_duty_cycle         : NATURAL;
      clk0_multiply_by        : NATURAL;
      clk0_phase_shift        : STRING;
      clk1_divide_by          : NATURAL;
      clk1_duty_cycle         : NATURAL;
      clk1_multiply_by        : NATURAL;
      clk1_phase_shift        : STRING;
      clk2_divide_by          : NATURAL;
      clk2_duty_cycle         : NATURAL;
      clk2_multiply_by        : NATURAL;
      clk2_phase_shift        : STRING;
      compensate_clock        : STRING;
      inclk0_input_frequency        : NATURAL;
      intended_device_family        : STRING;
      lpm_hint          : STRING;
      lpm_type          : STRING;
      operation_mode          : STRING;
      pll_type          : STRING;
      port_activeclock        : STRING;
      port_areset       : STRING;
      port_clkbad0            : STRING;
      port_clkbad1            : STRING;
      port_clkloss            : STRING;
      port_clkswitch          : STRING;
      port_configupdate       : STRING;
      port_fbin         : STRING;
      port_inclk0       : STRING;
      port_inclk1       : STRING;
      port_locked       : STRING;
      port_pfdena       : STRING;
      port_phasecounterselect       : STRING;
      port_phasedone          : STRING;
      port_phasestep          : STRING;
      port_phaseupdown        : STRING;
      port_pllena       : STRING;
      port_scanaclr           : STRING;
      port_scanclk            : STRING;
      port_scanclkena         : STRING;
      port_scandata           : STRING;
      port_scandataout        : STRING;
      port_scandone           : STRING;
      port_scanread           : STRING;
      port_scanwrite          : STRING;
      port_clk0         : STRING;
      port_clk1         : STRING;
      port_clk2         : STRING;
      port_clk3         : STRING;
      port_clk4         : STRING;
      port_clk5         : STRING;
      port_clkena0            : STRING;
```

```
            port_clkena1            : STRING;
            port_clkena2            : STRING;
            port_clkena3            : STRING;
            port_clkena4            : STRING;
            port_clkena5            : STRING;
            port_extclk0            : STRING;
            port_extclk1            : STRING;
            port_extclk2            : STRING;
            port_extclk3            : STRING;
            self_reset_on_loss_lock       : STRING;
            width_clock       : NATURAL
      );
      PORT (
                  areset      : IN STD_LOGIC ;
                  inclk : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
                  clk   : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);
                  locked      : OUT STD_LOGIC
      );
      END COMPONENT;

BEGIN
      sub_wire2_bv(0 DOWNTO 0) <= "0";
      sub_wire2    <= To_stdlogicvector(sub_wire2_bv);
      sub_wire0    <= inclk0;
      sub_wire1    <= sub_wire2(0 DOWNTO 0) & sub_wire0;
      sub_wire6    <= sub_wire3(2);
      sub_wire5    <= sub_wire3(1);
      sub_wire4    <= sub_wire3(0);
      c0    <= sub_wire4;
      c1    <= sub_wire5;
      c2    <= sub_wire6;
      locked    <= sub_wire7;

      altpll_component : altpll
      GENERIC MAP (
            bandwidth_type => "AUTO",
            clk0_divide_by => 1,
            clk0_duty_cycle => 50,
            clk0_multiply_by => 1,
            clk0_phase_shift => "0",
            clk1_divide_by => 500,
            clk1_duty_cycle => 50,
            clk1_multiply_by => 1,
            clk1_phase_shift => "0",
            clk2_divide_by => 1000,
            clk2_duty_cycle => 50,
            clk2_multiply_by => 1,
            clk2_phase_shift => "0",
            compensate_clock => "CLK0",
            inclk0_input_frequency => 20000,
            intended_device_family => "Cyclone IV E",
            lpm_hint => "CBX_MODULE_PREFIX=Pll",
            lpm_type => "altpll",
            operation_mode => "NORMAL",
            pll_type => "AUTO",
            port_activeclock => "PORT_UNUSED",
            port_areset => "PORT_USED",
            port_clkbad0 => "PORT_UNUSED",
            port_clkbad1 => "PORT_UNUSED",
            port_clkloss => "PORT_UNUSED",
            port_clkswitch => "PORT_UNUSED",
```

```
		port_configupdate => "PORT_UNUSED",
		port_fbin => "PORT_UNUSED",
		port_inclk0 => "PORT_USED",
		port_inclk1 => "PORT_UNUSED",
		port_locked => "PORT_USED",
		port_pfdena => "PORT_UNUSED",
		port_phasecounterselect => "PORT_UNUSED",
		port_phasedone => "PORT_UNUSED",
		port_phasestep => "PORT_UNUSED",
		port_phaseupdown => "PORT_UNUSED",
		port_pllena => "PORT_UNUSED",
		port_scanaclr => "PORT_UNUSED",
		port_scanclk => "PORT_UNUSED",
		port_scanclkena => "PORT_UNUSED",
		port_scandata => "PORT_UNUSED",
		port_scandataout => "PORT_UNUSED",
		port_scandone => "PORT_UNUSED",
		port_scanread => "PORT_UNUSED",
		port_scanwrite => "PORT_UNUSED",
		port_clk0 => "PORT_USED",
		port_clk1 => "PORT_USED",
		port_clk2 => "PORT_USED",
		port_clk3 => "PORT_UNUSED",
		port_clk4 => "PORT_UNUSED",
		port_clk5 => "PORT_UNUSED",
		port_clkena0 => "PORT_UNUSED",
		port_clkena1 => "PORT_UNUSED",
		port_clkena2 => "PORT_UNUSED",
		port_clkena3 => "PORT_UNUSED",
		port_clkena4 => "PORT_UNUSED",
		port_clkena5 => "PORT_UNUSED",
		port_extclk0 => "PORT_UNUSED",
		port_extclk1 => "PORT_UNUSED",
		port_extclk2 => "PORT_UNUSED",
		port_extclk3 => "PORT_UNUSED",
		self_reset_on_loss_lock => "ON",
		width_clock => 5
	)
	PORT MAP (
		areset => areset,
		inclk => sub_wire1,
		clk => sub_wire3,
		locked => sub_wire7
	);


END SYN;

-- ============================================================
-- CNX file retrieval info
-- ============================================================
-- Retrieval info: PRIVATE: ACTIVECLK_CHECK STRING "0"
-- Retrieval info: PRIVATE: BANDWIDTH STRING "1.000"
-- Retrieval info: PRIVATE: BANDWIDTH_FEATURE_ENABLED STRING "1"
-- Retrieval info: PRIVATE: BANDWIDTH_FREQ_UNIT STRING "MHz"
-- Retrieval info: PRIVATE: BANDWIDTH_PRESET STRING "Low"
-- Retrieval info: PRIVATE: BANDWIDTH_USE_AUTO STRING "1"
-- Retrieval info: PRIVATE: BANDWIDTH_USE_PRESET STRING "0"
-- Retrieval info: PRIVATE: CLKBAD_SWITCHOVER_CHECK STRING "0"
-- Retrieval info: PRIVATE: CLKLOSS_CHECK STRING "0"
```

```
-- Retrieval info: PRIVATE: CLKSWITCH_CHECK STRING "0"
-- Retrieval info: PRIVATE: CNX_NO_COMPENSATE_RADIO STRING "0"
-- Retrieval info: PRIVATE: CREATE_CLKBAD_CHECK STRING "0"
-- Retrieval info: PRIVATE: CREATE_INCLK1_CHECK STRING "0"
-- Retrieval info: PRIVATE: CUR_DEDICATED_CLK STRING "c0"
-- Retrieval info: PRIVATE: CUR_FBIN_CLK STRING "c0"
-- Retrieval info: PRIVATE: DEVICE_SPEED_GRADE STRING "8"
-- Retrieval info: PRIVATE: DIV_FACTOR0 NUMERIC "1"
-- Retrieval info: PRIVATE: DIV_FACTOR1 NUMERIC "1"
-- Retrieval info: PRIVATE: DIV_FACTOR2 NUMERIC "1"
-- Retrieval info: PRIVATE: DUTY_CYCLE0 STRING "50.00000000"
-- Retrieval info: PRIVATE: DUTY_CYCLE1 STRING "50.00000000"
-- Retrieval info: PRIVATE: DUTY_CYCLE2 STRING "50.00000000"
-- Retrieval info: PRIVATE: EFF_OUTPUT_FREQ_VALUE0 STRING "50.000000"
-- Retrieval info: PRIVATE: EFF_OUTPUT_FREQ_VALUE1 STRING "0.100000"
-- Retrieval info: PRIVATE: EFF_OUTPUT_FREQ_VALUE2 STRING "0.050000"
-- Retrieval info: PRIVATE: EXPLICIT_SWITCHOVER_COUNTER STRING "0"
-- Retrieval info: PRIVATE: EXT_FEEDBACK_RADIO STRING "0"
-- Retrieval info: PRIVATE: GLOCKED_COUNTER_EDIT_CHANGED STRING "1"
-- Retrieval info: PRIVATE: GLOCKED_FEATURE_ENABLED STRING "0"
-- Retrieval info: PRIVATE: GLOCKED_MODE_CHECK STRING "0"
-- Retrieval info: PRIVATE: GLOCK_COUNTER_EDIT NUMERIC "1048575"
-- Retrieval info: PRIVATE: HAS_MANUAL_SWITCHOVER STRING "1"
-- Retrieval info: PRIVATE: INCLK0_FREQ_EDIT STRING "50.000"
-- Retrieval info: PRIVATE: INCLK0_FREQ_UNIT_COMBO STRING "MHz"
-- Retrieval info: PRIVATE: INCLK1_FREQ_EDIT STRING "100.000"
-- Retrieval info: PRIVATE: INCLK1_FREQ_EDIT_CHANGED STRING "1"
-- Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_CHANGED STRING "1"
-- Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_COMBO STRING "MHz"
-- Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone IV
E"
-- Retrieval info: PRIVATE: INT_FEEDBACK__MODE_RADIO STRING "1"
-- Retrieval info: PRIVATE: LOCKED_OUTPUT_CHECK STRING "1"
-- Retrieval info: PRIVATE: LONG_SCAN_RADIO STRING "1"
-- Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE STRING "Not Available"
-- Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE_DIRTY NUMERIC "0"
-- Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNIT0 STRING "deg"
-- Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNIT1 STRING "deg"
-- Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNIT2 STRING "ps"
-- Retrieval info: PRIVATE: MIG_DEVICE_SPEED_GRADE STRING "Any"
-- Retrieval info: PRIVATE: MIRROR_CLK0 STRING "0"
-- Retrieval info: PRIVATE: MIRROR_CLK1 STRING "0"
-- Retrieval info: PRIVATE: MIRROR_CLK2 STRING "0"
-- Retrieval info: PRIVATE: MULT_FACTOR0 NUMERIC "1"
-- Retrieval info: PRIVATE: MULT_FACTOR1 NUMERIC "8"
-- Retrieval info: PRIVATE: MULT_FACTOR2 NUMERIC "1"
-- Retrieval info: PRIVATE: NORMAL_MODE_RADIO STRING "1"
-- Retrieval info: PRIVATE: OUTPUT_FREQ0 STRING "100.00000000"
-- Retrieval info: PRIVATE: OUTPUT_FREQ1 STRING "0.10000000"
-- Retrieval info: PRIVATE: OUTPUT_FREQ2 STRING "0.05000000"
-- Retrieval info: PRIVATE: OUTPUT_FREQ_MODE0 STRING "0"
-- Retrieval info: PRIVATE: OUTPUT_FREQ_MODE1 STRING "1"
-- Retrieval info: PRIVATE: OUTPUT_FREQ_MODE2 STRING "1"
-- Retrieval info: PRIVATE: OUTPUT_FREQ_UNIT0 STRING "MHz"
-- Retrieval info: PRIVATE: OUTPUT_FREQ_UNIT1 STRING "MHz"
-- Retrieval info: PRIVATE: OUTPUT_FREQ_UNIT2 STRING "MHz"
-- Retrieval info: PRIVATE: PHASE_RECONFIG_FEATURE_ENABLED STRING "1"
-- Retrieval info: PRIVATE: PHASE_RECONFIG_INPUTS_CHECK STRING "0"
-- Retrieval info: PRIVATE: PHASE_SHIFT0 STRING "0.00000000"
-- Retrieval info: PRIVATE: PHASE_SHIFT1 STRING "0.00000000"
-- Retrieval info: PRIVATE: PHASE_SHIFT2 STRING "0.00000000"
```

```
-- Retrieval info: PRIVATE: PHASE_SHIFT_STEP_ENABLED_CHECK STRING "0"
-- Retrieval info: PRIVATE: PHASE_SHIFT_UNIT0 STRING "deg"
-- Retrieval info: PRIVATE: PHASE_SHIFT_UNIT1 STRING "deg"
-- Retrieval info: PRIVATE: PHASE_SHIFT_UNIT2 STRING "ps"
-- Retrieval info: PRIVATE: PLL_ADVANCED_PARAM_CHECK STRING "0"
-- Retrieval info: PRIVATE: PLL_ARESET_CHECK STRING "1"
-- Retrieval info: PRIVATE: PLL_AUTOPLL_CHECK NUMERIC "1"
-- Retrieval info: PRIVATE: PLL_ENHPLL_CHECK NUMERIC "0"
-- Retrieval info: PRIVATE: PLL_FASTPLL_CHECK NUMERIC "0"
-- Retrieval info: PRIVATE: PLL_FBMIMIC_CHECK STRING "0"
-- Retrieval info: PRIVATE: PLL_LVDS_PLL_CHECK NUMERIC "0"
-- Retrieval info: PRIVATE: PLL_PFDENA_CHECK STRING "0"
-- Retrieval info: PRIVATE: PLL_TARGET_HARCOPY_CHECK NUMERIC "0"
-- Retrieval info: PRIVATE: PRIMARY_CLK_COMBO STRING "inclk0"
-- Retrieval info: PRIVATE: RECONFIG_FILE STRING "Pll.mif"
-- Retrieval info: PRIVATE: SACN_INPUTS_CHECK STRING "0"
-- Retrieval info: PRIVATE: SCAN_FEATURE_ENABLED STRING "1"
-- Retrieval info: PRIVATE: SELF_RESET_LOCK_LOSS STRING "1"
-- Retrieval info: PRIVATE: SHORT_SCAN_RADIO STRING "0"
-- Retrieval info: PRIVATE: SPREAD_FEATURE_ENABLED STRING "0"
-- Retrieval info: PRIVATE: SPREAD_FREQ STRING "50.000"
-- Retrieval info: PRIVATE: SPREAD_FREQ_UNIT STRING "KHz"
-- Retrieval info: PRIVATE: SPREAD_PERCENT STRING "0.500"
-- Retrieval info: PRIVATE: SPREAD_USE STRING "0"
-- Retrieval info: PRIVATE: SRC_SYNCH_COMP_RADIO STRING "0"
-- Retrieval info: PRIVATE: STICKY_CLK0 STRING "1"
-- Retrieval info: PRIVATE: STICKY_CLK1 STRING "1"
-- Retrieval info: PRIVATE: STICKY_CLK2 STRING "1"
-- Retrieval info: PRIVATE: SWITCHOVER_COUNT_EDIT NUMERIC "1"
-- Retrieval info: PRIVATE: SWITCHOVER_FEATURE_ENABLED STRING "1"
-- Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
-- Retrieval info: PRIVATE: USE_CLK0 STRING "1"
-- Retrieval info: PRIVATE: USE_CLK1 STRING "1"
-- Retrieval info: PRIVATE: USE_CLK2 STRING "1"
-- Retrieval info: PRIVATE: USE_CLKENA0 STRING "0"
-- Retrieval info: PRIVATE: USE_CLKENA1 STRING "0"
-- Retrieval info: PRIVATE: USE_CLKENA2 STRING "0"
-- Retrieval info: PRIVATE: USE_MIL_SPEED_GRADE NUMERIC "0"
-- Retrieval info: PRIVATE: ZERO_DELAY_RADIO STRING "0"
-- Retrieval info: LIBRARY: altera_mf
altera_mf.altera_mf_components.all
-- Retrieval info: CONSTANT: BANDWIDTH_TYPE STRING "AUTO"
-- Retrieval info: CONSTANT: CLK0_DIVIDE_BY NUMERIC "1"
-- Retrieval info: CONSTANT: CLK0_DUTY_CYCLE NUMERIC "50"
-- Retrieval info: CONSTANT: CLK0_MULTIPLY_BY NUMERIC "1"
-- Retrieval info: CONSTANT: CLK0_PHASE_SHIFT STRING "0"
-- Retrieval info: CONSTANT: CLK1_DIVIDE_BY NUMERIC "500"
-- Retrieval info: CONSTANT: CLK1_DUTY_CYCLE NUMERIC "50"
-- Retrieval info: CONSTANT: CLK1_MULTIPLY_BY NUMERIC "1"
-- Retrieval info: CONSTANT: CLK1_PHASE_SHIFT STRING "0"
-- Retrieval info: CONSTANT: CLK2_DIVIDE_BY NUMERIC "1000"
-- Retrieval info: CONSTANT: CLK2_DUTY_CYCLE NUMERIC "50"
-- Retrieval info: CONSTANT: CLK2_MULTIPLY_BY NUMERIC "1"
-- Retrieval info: CONSTANT: CLK2_PHASE_SHIFT STRING "0"
-- Retrieval info: CONSTANT: COMPENSATE_CLOCK STRING "CLK0"
-- Retrieval info: CONSTANT: INCLK0_INPUT_FREQUENCY NUMERIC "20000"
-- Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "Cyclone IV
E"
-- Retrieval info: CONSTANT: LPM_TYPE STRING "altpll"
-- Retrieval info: CONSTANT: OPERATION_MODE STRING "NORMAL"
-- Retrieval info: CONSTANT: PLL_TYPE STRING "AUTO"
```

Pll.vhd

```
-- Retrieval info: CONSTANT: PORT_ACTIVECLOCK STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_ARESET STRING "PORT_USED"
-- Retrieval info: CONSTANT: PORT_CLKBAD0 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_CLKBAD1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_CLKLOSS STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_CLKSWITCH STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_CONFIGUPDATE STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_FBIN STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_INCLK0 STRING "PORT_USED"
-- Retrieval info: CONSTANT: PORT_INCLK1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_LOCKED STRING "PORT_USED"
-- Retrieval info: CONSTANT: PORT_PFDENA STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PHASECOUNTERSELECT STRING
"PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PHASEDONE STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PHASESTEP STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PHASEUPDOWN STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_PLLENA STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANACLR STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANCLK STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANCLKENA STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANDATA STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANDATAOUT STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANDONE STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANREAD STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_SCANWRITE STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clk0 STRING "PORT_USED"
-- Retrieval info: CONSTANT: PORT_clk1 STRING "PORT_USED"
-- Retrieval info: CONSTANT: PORT_clk2 STRING "PORT_USED"
-- Retrieval info: CONSTANT: PORT_clk3 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clk4 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clk5 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena0 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena2 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena3 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena4 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_clkena5 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_extclk0 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_extclk1 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_extclk2 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: PORT_extclk3 STRING "PORT_UNUSED"
-- Retrieval info: CONSTANT: SELF_RESET_ON_LOSS_LOCK STRING "ON"
-- Retrieval info: CONSTANT: WIDTH_CLOCK NUMERIC "5"
-- Retrieval info: USED_PORT: @clk 0 0 5 0 OUTPUT_CLK_EXT VCC
"@clk[4..0]"
-- Retrieval info: USED_PORT: @inclk 0 0 2 0 INPUT_CLK_EXT VCC
"@inclk[1..0]"
-- Retrieval info: USED_PORT: areset 0 0 0 0 INPUT GND "areset"
-- Retrieval info: USED_PORT: c0 0 0 0 0 OUTPUT_CLK_EXT VCC "c0"
-- Retrieval info: USED_PORT: c1 0 0 0 0 OUTPUT_CLK_EXT VCC "c1"
-- Retrieval info: USED_PORT: c2 0 0 0 0 OUTPUT_CLK_EXT VCC "c2"
-- Retrieval info: USED_PORT: inclk0 0 0 0 0 INPUT_CLK_EXT GND
"inclk0"
-- Retrieval info: USED_PORT: locked 0 0 0 0 OUTPUT GND "locked"
-- Retrieval info: CONNECT: @areset 0 0 0 0 areset 0 0 0 0
-- Retrieval info: CONNECT: @inclk 0 0 1 1 GND 0 0 0 0
-- Retrieval info: CONNECT: @inclk 0 0 1 0 inclk0 0 0 0 0
-- Retrieval info: CONNECT: c0 0 0 0 0 @clk 0 0 1 0
-- Retrieval info: CONNECT: c1 0 0 0 0 @clk 0 0 1 1
-- Retrieval info: CONNECT: c2 0 0 0 0 @clk 0 0 1 2
```

Pll.vhd

```
-- Retrieval info: CONNECT: locked 0 0 0 0 @locked 0 0 0 0
-- Retrieval info: GEN_FILE: TYPE_NORMAL Pll.vhd TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL Pll.ppf TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL Pll.inc FALSE
-- Retrieval info: GEN_FILE: TYPE_NORMAL Pll.cmp TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL Pll.bsf TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL Pll_inst.vhd FALSE
-- Retrieval info: LIB_FILE: altera_mf
-- Retrieval info: CBX_MODULE_PREFIX: ON
```

Pll.vhd

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              PulseCounter
-- Description:         SingleCh and Coincidences Pulse Counter
-- ******************************************************************

-- ##################################################################
-- Revision History:
--    Date      Author              Rev.    Comments
-- 14 May 22   Diogo Ayres Rocha   0001    File Created
-- 09 Ago 23   Diogo Ayres Rocha      0002    Added Counter Reset on
Release
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

LIBRARY work;
USE work.pulse_count_pkg.all;

entity PulseCounter is
port(
      CK_20ns : in  std_logic;
      RST     : in  std_logic;
      P       : in  std_logic_vector (71 downto 0);
      RELEASE : in  std_logic;
      COUNT   : out vec24b(71 downto 0)
);
end PulseCounter;

architecture fPulseCounter of PulseCounter is

      signal P_PREV       : std_logic_vector (71 downto 0)       :=
(others => '0');
      signal RELEASE_PREV : std_logic
            := '0';
      signal S_COUNT      : vec24b(71 downto 0)
      := VEC24bx72_ZERO; -- array 72 counts x 24-bits

begin

REdgeCount : process(CK_20ns, RST)
      variable num_pulse : vec24b(71 downto 0)
      := VEC24bx72_ZERO; -- array 72 counts x 24-bits

begin
      if RST = '1' then
            num_pulse := VEC24bx72_ZERO;
            S_COUNT   <= VEC24bx72_ZERO;
      elsif rising_edge(CK_20ns) then
            for i in 0 to 71 loop
               if RELEASE_PREV = '1' and RELEASE = '0' then
                        num_pulse(i) := vec24b_zero;
```

PulseCounter.vhd

```
                  end if;
                  if P_PREV(i) = '0' and P(i) = '1' and num_pulse(i) <
reg24b_range then
                        num_pulse(i) := num_pulse(i) + 1;
                  end if;
                  S_COUNT(i) <= num_pulse(i);
            end loop;
      end if;
end process;

CountOutUpdate : process(CK_20ns, RST)
begin
      if RST = '1' then
            COUNT <= VEC24bx72_ZERO;

      elsif rising_edge(CK_20ns) then
            if RELEASE_PREV = '1' and RELEASE = '0' then
                  COUNT <= S_COUNT;
            end if;
      end if;
end process;

PrevUpdate : process(CK_20ns, RST)
begin
      if RST = '1' then
            P_PREV       <= (others => '0');
            RELEASE_PREV <= '0';

      elsif rising_edge(CK_20ns) then
            P_PREV       <= P;
            RELEASE_PREV <= RELEASE;

      end if;
end process;

end fPulseCounter;
```

PulseCounter.vhd

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha
-- Date:                10-07-2023
-- ------------------------------------------------------------------
-- Module:              PulseGenerator
-- Description:         LED Light Injector Pulse Generator
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author            Rev.   Comments
-- 10 Jul 23 Diogo Ayres Rocha  0001   File created
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

LIBRARY work;
USE work.my_types_pkg.all;
USE work.reg_addresses.all;

entity PulseGenerator is
 port(
  --Pulses Input/Output:
  CLK_100kHz    : in  std_logic;
  CLK_200MHz    : in  std_logic;
  RESET              : in  std_logic;
  GlobalReg     : in  ByteArray(63 downto 0);          -- Global Config
Memory Registers
  P_OUT         : out std_logic_vector (4 downto 0)  -- Pulse Output
 );
end PulseGenerator;

architecture circuit of PulseGenerator is

      component LEDClksGen is
      port(
            CLK_100kHz    : in  std_logic;
            RST                 : in  std_logic;
            CLKOUT_50kHz  : out std_logic;
            CLKOUT_25kHz  : out std_logic;
            CLKOUT_10kHz  : out std_logic;
            CLKOUT_5kHz   : out std_logic;
            CLKOUT_1kHz   : out std_logic
      );
      end component;

      component WidthLEDControl is
      port(
            CK_5ns      : in  std_logic;
            RST         : in  std_logic;
            nT_GATE : in  unsigned (7 downto 0);
            CH_IN   : in  std_logic;
            CH_OUT  : out std_logic
      );
```

PulseGenerator.vhd

```
    end component;


    signal LED0ClkSelected     : std_logic := '0';
    signal LED1ClkSelected     : std_logic := '0';
    signal LED2ClkSelected     : std_logic := '0';
    signal LED3ClkSelected     : std_logic := '0';
    signal TriggerClkSelected  : std_logic := '0';

    signal CLK_50kHz   : std_logic := '0';
    signal CLK_25kHz   : std_logic := '0';
    signal CLK_10kHz   : std_logic := '0';
    signal CLK_5kHz      : std_logic := '0';
    signal CLK_1kHz      : std_logic := '0';

    signal LED0    : std_logic := '0';
    signal LED1    : std_logic := '0';
    signal LED2    : std_logic := '0';
    signal LED3    : std_logic := '0';
    signal Trigger : std_logic := '0';

begin

   clockModule          : LEDClksGen      port map (CLK_100kHz, RESET,
CLK_50kHz, CLK_25kHz, CLK_10kHz, CLK_5kHz, CLK_1kHz);
      widthModuleLED0    : WidthLEDControl port map (CLK_200MHz,
RESET, GlobalReg(LED0InjWidthSel)+GlobalReg(MinimumWidthSel),
LED0ClkSelected, LED0);
      widthModuleLED1    : WidthLEDControl port map (CLK_200MHz,
RESET, GlobalReg(LED1InjWidthSel)+GlobalReg(MinimumWidthSel),
LED1ClkSelected, LED1);
      widthModuleLED2    : WidthLEDControl port map (CLK_200MHz,
RESET, GlobalReg(LED2InjWidthSel)+GlobalReg(MinimumWidthSel),
LED2ClkSelected, LED2);
      widthModuleLED3    : WidthLEDControl port map (CLK_200MHz,
RESET, GlobalReg(LED3InjWidthSel)+GlobalReg(MinimumWidthSel),
LED3ClkSelected, LED3);
      widthModuleTrigger : WidthLEDControl port map (CLK_200MHz,
RESET, GlobalReg(TriggerWidthSel)+GlobalReg(MinimumWidthSel),
TriggerClkSelected, Trigger);

      P_OUT(0) <= LED0 and GlobalReg(0)(0);
      P_OUT(1) <= LED1 and GlobalReg(0)(1);
      P_OUT(2) <= LED2 and GlobalReg(0)(2);
      P_OUT(3) <= LED3 and GlobalReg(0)(3);
      P_OUT(4) <= Trigger and GlobalReg(0)(4);

      with (GlobalReg(LED0InjFreqSel)) select
        LED0ClkSelected <= CLK_1kHz WHEN "00000000",
      CLK_5kHz WHEN "00000001" ,
      CLK_10kHz WHEN "00000010" ,
      CLK_25kHz WHEN "00000011" ,
      CLK_50kHz WHEN "00000100" ,
      CLK_100kHz WHEN "00000101" ,
      '0' WHEN OTHERS;

      with (GlobalReg(LED1InjFreqSel)) select
        LED1ClkSelected <= CLK_1kHz WHEN "00000000",
        CLK_5kHz WHEN "00000001",
        CLK_10kHz WHEN "00000010",
        CLK_25kHz WHEN "00000011",
```

PulseGenerator.vhd

```
        CLK_50kHz WHEN "00000100",
        CLK_100kHz WHEN "00000101",
        '0' WHEN OTHERS;

    with (GlobalReg(LED2InjFreqSel)) select
        LED2ClkSelected <= CLK_1kHz WHEN "00000000",
        CLK_5kHz WHEN "00000001",
        CLK_10kHz WHEN "00000010",
        CLK_25kHz WHEN "00000011",
        CLK_50kHz WHEN "00000100",
        CLK_100kHz WHEN "00000101",
        '0' WHEN OTHERS;

    with (GlobalReg(LED3InjFreqSel)) select
        LED3ClkSelected <= CLK_1kHz WHEN "00000000",
        CLK_5kHz WHEN "00000001",
        CLK_10kHz WHEN "00000010",
        CLK_25kHz WHEN "00000011",
        CLK_50kHz WHEN "00000100",
        CLK_100kHz WHEN "00000101",
        '0' WHEN OTHERS;

    with (GlobalReg(TriggerFreqSel)) select
        TriggerClkSelected <= CLK_1kHz WHEN "00000000",
        CLK_5kHz WHEN "00000001",
        CLK_10kHz WHEN "00000010",
        CLK_25kHz WHEN "00000011",
        CLK_50kHz WHEN "00000100",
        CLK_100kHz WHEN "00000101",
        '0' WHEN OTHERS;

end circuit;
```

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                14-05-2022
-- ------------------------------------------------------------------
-- Module:              RegAdapterA
-- Description:         Counters Values (3Bytes) Adapter to 1Byte Regs
-- ******************************************************************

-- ##################################################################
-- Revision History:
--    Date     Author              Rev.    Comments
-- 14 May 22   Diogo Ayres Rocha   0001    File Created
-- ##################################################################


LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

LIBRARY work;
USE work.my_types_pkg.all;
USE work.pulse_count_pkg.all;

entity RegAdapterA is
port(
      COUNT  : in  vec24b(71 downto 0);
      MEMORY : out ByteArray(225 downto 0)
);
end RegAdapterA;

architecture fRegAdapterA of RegAdapterA is
begin

process(COUNT)
begin
      for i in 0 to 71 loop
            MEMORY(3*i + 0)(7 downto 0) <= COUNT(i)(23 downto 16);
            MEMORY(3*i + 1)(7 downto 0) <= COUNT(i)(15 downto  8);
            MEMORY(3*i + 2)(7 downto 0) <= COUNT(i)( 7 downto  0);
      end loop;
end process;

end fRegAdapterA;
```

RegAdapterA.vhd

```
-- WARNING: Do NOT edit the input and output ports in this file in a
text
-- editor if you plan to continue editing the block that represents it
in
-- the Block Editor! File corruption is VERY likely to occur.

-- Copyright (C) 2016  Intel Corporation. All rights reserved.
-- Your use of Intel Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- associated documentation or information are expressly subject
-- to the terms and conditions of the Intel Program License
-- Subscription Agreement, the Intel Quartus Prime License Agreement,
-- the Intel MegaCore Function License Agreement, or other
-- applicable license agreement, including, without limitation,
-- that your use is for the sole purpose of programming logic
-- devices manufactured by Intel and sold by Intel or its
-- authorized distributors.  Please refer to the applicable
-- agreement for further details.


-- Generated by Quartus Prime Version 16.1 (Build Build 196
10/24/2016)
-- Created on Thu May 04 15:58:21 2017

LIBRARY ieee;
USE ieee.std_logic_1164.all;


--  Entity Declaration

ENTITY SlaveAddressA IS
-- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
GENERIC(Address : STD_LOGIC_VECTOR(7 DOWNTO 0) := X"01");
PORT
(
I2C_SlaveAdd : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
);
-- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

END SlaveAddressA;


--  Architecture Body

ARCHITECTURE SlaveAddress_architecture OF SlaveAddressA IS


BEGIN

I2C_SlaveAdd <= Address(6 DOWNTO 0);

END SlaveAddress_architecture;
```

SlaveAddressA.vhd

```
-- ******************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
-- Project:             CRE4AT
-- Model:               CRE4AT DAQ V3
-- FPGA Proj. Name:     FW_CREAT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Diogo Ayres Rocha (dayres@cbpf.br)
-- Date:                09-08-2023
-- ------------------------------------------------------------------
-- Module:              WidthLEDControl
-- Description:         LED Pulse Width Control (Based on Latch Block)
-- ******************************************************************

-- ##################################################################
-- Revision History:
--   Date     Author              Rev.   Comments
-- 09 Ago 23  Diogo Ayres Rocha     0001   File Created
-- ##################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

LIBRARY work;
USE work.pulse_count_pkg.all;

entity WidthLEDControl is
port(
      CK_20ns : in  std_logic;
      RST         : in  std_logic;
      nT_GATE : in  unsigned (7 downto 0);
      CH_IN   : in  std_logic;
      CH_OUT  : out std_logic
);
end WidthLEDControl;

architecture fWidthLEDControl of WidthLEDControl is
      -- !!! CH_SWT and LT_SWT must necessarily be initialyzed with
inverted corresponding bits !!!
      signal CH_SWT : std_logic := '1';
      signal LT_SWT : std_logic := '0';
      signal CH_LAT : std_logic := '0';
      signal SET_LT : std_logic := '0';

begin

Ch0_EdgeDetect : process(CH_IN, RST, SET_LT, CH_LAT)
begin
      if RST = '1' then
            CH_SWT <= '1';
      else
            if rising_edge(CH_IN) and SET_LT = '0' and CH_LAT = '0'
then
                  CH_SWT <= not CH_SWT;
            end if;
      end if;
end process;

Latch : process(CK_20ns, RST, SET_LT, CH_LAT)
      variable num_ticks : integer range 0 to 255;
```

WidthLEDControl.vhd

```
begin
      if RST = '1' then
            LT_SWT    <= '0';
            CH_LAT    <= '0';
            num_ticks :=  0;

      elsif falling_edge(CK_20ns) then

        if SET_LT = '1' then
            LT_SWT    <= not LT_SWT;
            CH_LAT    <= '1';
            num_ticks :=  0 ;

        elsif CH_LAT = '1' then
            num_ticks := num_ticks + 1;

            if num_ticks >= to_integer(nT_GATE) then
                CH_LAT    <= '0';
                num_ticks :=  0 ;
            end if;
        end if;
      end if;
end process;

      SET_LT <= CH_SWT xnor LT_SWT;
      CH_OUT <= CH_LAT;

end fWidthLEDControl;
```

WidthLEDControl.vhd

```
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// i2cSlave.v                                                   ////
////                                                              ////
//// This file is part of the i2cSlave opencores effort.
//// <http://www.opencores.org/cores//>                           ////
////                                                              ////
//// Module Description:                                          ////
//// You will need to modify this file to implement your
//// interface.
////                                                              ////
//// To Do:                                                       ////
////
////                                                              ////
//// Author(s):                                                   ////
//// - Steve Fielding, sfielding@base2designs.com                 ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// Copyright (C) 2008 Steve Fielding and OPENCORES.ORG          ////
////                                                              ////
//// This source file may be used and distributed without         ////
//// restriction provided that this copyright statement is not    ////
//// removed from the file and that any derivative work contains  ////
//// the original copyright notice and the associated disclaimer. ////
////                                                              ////
//// This source file is free software; you can redistribute it   ////
//// and/or modify it under the terms of the GNU Lesser General   ////
//// Public License as published by the Free Software Foundation; ////
//// either version 2.1 of the License, or (at your option) any   ////
//// later version.                                               ////
////                                                              ////
//// This source is distributed in the hope that it will be       ////
//// useful, but WITHOUT ANY WARRANTY; without even the implied   ////
//// warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR      ////
//// PURPOSE. See the GNU Lesser General Public License for more  ////
//// details.                                                     ////
////                                                              ////
//// You should have received a copy of the GNU Lesser General    ////
//// Public License along with this source; if not, download it   ////
//// from <http://www.opencores.org/lgpl.shtml>                   ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
//
`include "i2cSlave_define.v"


module i2cSlave (
  clk,
  rst,
  sda,
  scl,
  addr,
  SlaveAddr,
  dataIn,
  writeEn,
  dataOut
);

input clk;
input rst;
```

```
inout sda;
input scl;

// new pins
input  [6:0] SlaveAddr;

output [7:0] addr;
output writeEn;

input  [7:0] dataOut;
output [7:0] dataIn;


// local wires and regs
reg sdaDeb;
reg sclDeb;
reg [`DEB_I2C_LEN-1:0] sdaPipe;
reg [`DEB_I2C_LEN-1:0] sclPipe;

reg [`SCL_DEL_LEN-1:0] sclDelayed;
reg [`SDA_DEL_LEN-1:0] sdaDelayed;
reg [1:0] startStopDetState;
wire clearStartStopDet;
wire sdaOut;
wire sdaIn;

wire [6:0] SlaveAddr; // new line
wire [7:0] regAddr;
wire [7:0] dataToRegIF;
wire writeEn;
wire [7:0] dataFromRegIF;
reg [1:0] rstPipe;
wire rstSyncToClk;
reg startEdgeDet;

assign sda = (sdaOut == 1'b0) ? 1'b0 : 1'bz;
assign sdaIn = sda;

assign addr=regAddr;
assign dataIn=dataToRegIF;
assign dataFromRegIF=dataOut;

// sync rst rsing edge to clk
always @(posedge clk) begin
  if (rst == 1'b1)
    rstPipe <= 2'b11;
  else
    rstPipe <= {rstPipe[0], 1'b0};
end

assign rstSyncToClk = rstPipe[1];

// debounce sda and scl
always @(posedge clk) begin
  if (rstSyncToClk == 1'b1) begin
    sdaPipe <= {`DEB_I2C_LEN{1'b1}};
    sdaDeb <= 1'b1;
    sclPipe <= {`DEB_I2C_LEN{1'b1}};
    sclDeb <= 1'b1;
  end
```

i2cSlave/i2cSlave.v

```
  else begin
    sdaPipe <= {sdaPipe[`DEB_I2C_LEN-2:0], sdaIn};
    sclPipe <= {sclPipe[`DEB_I2C_LEN-2:0], scl};
    if (&sclPipe[`DEB_I2C_LEN-1:1] == 1'b1)
      sclDeb <= 1'b1;
    else if (|sclPipe[`DEB_I2C_LEN-1:1] == 1'b0)
      sclDeb <= 1'b0;
    if (&sdaPipe[`DEB_I2C_LEN-1:1] == 1'b1)
      sdaDeb <= 1'b1;
    else if (|sdaPipe[`DEB_I2C_LEN-1:1] == 1'b0)
      sdaDeb <= 1'b0;
  end
end


// delay scl and sda
// sclDelayed is used as a delayed sampling clock
// sdaDelayed is only used for start stop detection
// Because sda hold time from scl falling is 0nS
// sda must be delayed with respect to scl to avoid incorrect
// detection of start/stop at scl falling edge.
always @(posedge clk) begin
  if (rstSyncToClk == 1'b1) begin
    sclDelayed <= {`SCL_DEL_LEN{1'b1}};
    sdaDelayed <= {`SDA_DEL_LEN{1'b1}};
  end
  else begin
    sclDelayed <= {sclDelayed[`SCL_DEL_LEN-2:0], sclDeb};
    sdaDelayed <= {sdaDelayed[`SDA_DEL_LEN-2:0], sdaDeb};
  end
end

// start stop detection
always @(posedge clk) begin
  if (rstSyncToClk == 1'b1) begin
    startStopDetState <= `NULL_DET;
    startEdgeDet <= 1'b0;
  end
  else begin
    if (sclDeb == 1'b1 && sdaDelayed[`SDA_DEL_LEN-2] == 1'b0 &&
sdaDelayed[`SDA_DEL_LEN-1] == 1'b1)
      startEdgeDet <= 1'b1;
    else
      startEdgeDet <= 1'b0;
    if (clearStartStopDet == 1'b1)
      startStopDetState <= `NULL_DET;
    else if (sclDeb == 1'b1) begin
      if (sdaDelayed[`SDA_DEL_LEN-2] == 1'b1 &&
sdaDelayed[`SDA_DEL_LEN-1] == 1'b0)
        startStopDetState <= `STOP_DET;
      else if (sdaDelayed[`SDA_DEL_LEN-2] == 1'b0 &&
sdaDelayed[`SDA_DEL_LEN-1] == 1'b1)
        startStopDetState <= `START_DET;
    end
  end
end

serialInterface u_serialInterface (
  .clk(clk),
  .rst(rstSyncToClk | startEdgeDet),
  .SlaveAddr(SlaveAddr), //new line
```

```
  .dataIn(dataFromRegIF),
  .dataOut(dataToRegIF),
  .writeEn(writeEn),
  .regAddr(regAddr),
  .scl(sclDelayed[`SCL_DEL_LEN-1]),
  .sdaIn(sdaDeb),
  .sdaOut(sdaOut),
  .startStopDetState(startStopDetState),
  .clearStartStopDet(clearStartStopDet)
);


endmodule
```

```
// ----------------------- i2cSlave_define.v --------------------

// stream states
`define STREAM_IDLE 2'b00
`define STREAM_READ 2'b01
`define STREAM_WRITE_ADDR 2'b10
`define STREAM_WRITE_DATA 2'b11

// start stop detection states
`define NULL_DET 2'b00
`define START_DET 2'b01
`define STOP_DET 2'b10

// i2c ack and nak
`define I2C_NAK 1'b1
`define I2C_ACK 1'b0

// ------------------------------------------------------------------
// ------------- modify constants below this line -------------------
// ------------------------------------------------------------------

// i2c device address
`define I2C_ADDRESS 7'h3c

// System clock frequency in MHz
// If you are using a clock frequency below 24MHz, then the macro
// for SDA_DEL_LEN will result in compile errors for i2cSlave.v
// you will need to hand tweak the SDA_DEL_LEN constant definition
`define CLK_FREQ 48

// Debounce SCL and SDA over this many clock ticks
// The rise time of SCL and SDA can be up to 1000nS (in standard mode)
// so it is essential to debounce the inputs.
// The spec requires 0.05V of hysteresis, but in practise
// simply debouncing the inputs is sufficient
// I2C spec requires suppresion of spikes of
// maximum duration 50nS, so this debounce time should be greater than
50nS
// Also increases data hold time and decreases data setup time
// during an I2C read operation
// 10 ticks = 208nS @ 48MHz
`define DEB_I2C_LEN (10*`CLK_FREQ)/48

// Delay SCL for use as internal sampling clock
// Using delayed version of SCL to ensure that
// SDA is stable when it is sampled.
// Not entirely citical, as according to I2C spec
// SDA should have a minimum of 100nS of set up time
// with respect to SCL rising edge. But with the very slow edge
// speeds used in I2C it is better to err on the side of caution.
// This delay also has the effect of adding extra hold time to the
data
// with respect to SCL falling edge. I2C spec requires 0nS of data
hold time.
// 10 ticks = 208nS @ 48MHz
`define SCL_DEL_LEN (10*`CLK_FREQ)/48

// Delay SDA for use in start/stop detection
// Use delayed SDA during start/stop detection to avoid
// incorrect detection at SCL falling edge.
```

```
// From I2C spec start/stop setup is 600nS with respect to SCL rising
edge
// and start/stop hold is 600nS wrt SCL falling edge.
// So it is relatively easy to discriminate start/stop,
// but data setup time is a minimum of 100nS with respect to SCL
rising edge
// and 0nS hold wrt to SCL falling edge.
// So the tricky part is providing robust start/stop detection
// in the presence of regular data transitions.
// This delay time should be less than 100nS
// 4 ticks = 83nS @ 48MHz
`define SDA_DEL_LEN (4*`CLK_FREQ)/48
```

i2cSlave/i2cSlave_define.v

```
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// i2cSlaveTop.v
////
////                                                              ////
//// This file is part of the i2cSlave opencores effort.
//// <http://www.opencores.org/cores//>                           ////
////                                                              ////
//// Module Description:                                          ////
//// You will need to modify this file to implement your
//// interface.
////                                                              ////
//// To Do:                                                       ////
////
////                                                              ////
//// Author(s):                                                   ////
//// - Steve Fielding, sfielding@base2designs.com                 ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// Copyright (C) 2008 Steve Fielding and OPENCORES.ORG          ////
////                                                              ////
//// This source file may be used and distributed without         ////
//// restriction provided that this copyright statement is not    ////
//// removed from the file and that any derivative work contains  ////
//// the original copyright notice and the associated disclaimer. ////
////                                                              ////
//// This source file is free software; you can redistribute it   ////
//// and/or modify it under the terms of the GNU Lesser General   ////
//// Public License as published by the Free Software Foundation; ////
//// either version 2.1 of the License, or (at your option) any   ////
//// later version.                                               ////
////                                                              ////
//// This source is distributed in the hope that it will be       ////
//// useful, but WITHOUT ANY WARRANTY; without even the implied   ////
//// warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR      ////
//// PURPOSE. See the GNU Lesser General Public License for more  ////
//// details.                                                     ////
////                                                              ////
//// You should have received a copy of the GNU Lesser General    ////
//// Public License along with this source; if not, download it   ////
//// from <http://www.opencores.org/lgpl.shtml>                   ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
//
`include "i2cSlave_define.v"


module i2cSlaveTop (
  clk,
  rst,
  sda,
  scl,
  myReg0
);
input clk;
input rst;
inout sda;
input scl;
output [7:0] myReg0;
```

```
i2cSlave u_i2cSlave(
  .clk(clk),
  .rst(rst),
  .sda(sda),
  .scl(scl),
  .myReg0(myReg0),
  .myReg1(),
  .myReg2(),
  .myReg3(),
  .myReg4(8'h12),
  .myReg5(8'h34),
  .myReg6(8'h56),
  .myReg7(8'h78)

);


endmodule
```

i2cSlave/i2cSlaveTop.v

```
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// registerInterface.v                                          ////
////                                                              ////
//// This file is part of the i2cSlave opencores effort.
//// <http://www.opencores.org/cores//>                           ////
////                                                              ////
//// Module Description:                                          ////
//// You will need to modify this file to implement your
//// interface.
//// Add your control and status bytes/bits to module inputs and
outputs,
//// and also to the I2C read and write process blocks
////                                                              ////
//// To Do:                                                       ////
////
////                                                              ////
//// Author(s):                                                   ////
//// - Steve Fielding, sfielding@base2designs.com                 ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// Copyright (C) 2008 Steve Fielding and OPENCORES.ORG           ////
////                                                              ////
//// This source file may be used and distributed without         ////
//// restriction provided that this copyright statement is not    ////
//// removed from the file and that any derivative work contains  ////
//// the original copyright notice and the associated disclaimer. ////
////                                                              ////
//// This source file is free software; you can redistribute it   ////
//// and/or modify it under the terms of the GNU Lesser General   ////
//// Public License as published by the Free Software Foundation; ////
//// either version 2.1 of the License, or (at your option) any   ////
//// later version.                                               ////
////                                                              ////
//// This source is distributed in the hope that it will be       ////
//// useful, but WITHOUT ANY WARRANTY; without even the implied   ////
//// warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR      ////
//// PURPOSE. See the GNU Lesser General Public License for more  ////
//// details.                                                     ////
////                                                              ////
//// You should have received a copy of the GNU Lesser General    ////
//// Public License along with this source; if not, download it   ////
//// from <http://www.opencores.org/lgpl.shtml>                   ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
//
`include "i2cSlave_define.v"


module registerInterface (
  clk,
  addr,
  dataIn,
  writeEn,
  dataOut,
  myReg0,
  myReg1,
  myReg2,
  myReg3,
  myReg4,
```

```
  myReg5,
  myReg6,
  myReg7

);
input clk;
input [7:0] addr;
input [7:0] dataIn;
input writeEn;
output [7:0] dataOut;
output [7:0] myReg0;
output [7:0] myReg1;
output [7:0] myReg2;
output [7:0] myReg3;
input [7:0] myReg4;
input [7:0] myReg5;
input [7:0] myReg6;
input [7:0] myReg7;

reg [7:0] dataOut;
reg [7:0] myReg0;
reg [7:0] myReg1;
reg [7:0] myReg2;
reg [7:0] myReg3;

// --- I2C Read
always @(posedge clk) begin
  case (addr)
    8'h00: dataOut <= myReg0;
    8'h01: dataOut <= myReg1;
    8'h02: dataOut <= myReg2;
    8'h03: dataOut <= myReg3;
    8'h04: dataOut <= myReg4;
    8'h05: dataOut <= myReg5;
    8'h06: dataOut <= myReg6;
    8'h07: dataOut <= myReg7;
    default: dataOut <= 8'h00;
  endcase
end

// --- I2C Write
always @(posedge clk) begin
  if (writeEn == 1'b1) begin
    case (addr)
      8'h00: myReg0 <= dataIn;
      8'h01: myReg1 <= dataIn;
      8'h02: myReg2 <= dataIn;
      8'h03: myReg3 <= dataIn;
    endcase
  end
end

endmodule
```

i2cSlave/registerInterface.v

```
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// serialInterface.v                                            ////
////                                                              ////
//// This file is part of the i2cSlave opencores effort.
//// <http://www.opencores.org/cores//>                           ////
////                                                              ////
//// Module Description:                                          ////
//// Perform all serial to parallel, and parallel
//// to serial conversions. Perform device address matching
//// Handle arbitrary length I2C reads terminated by NAK
//// from host, and arbitrary length I2C writes terminated
//// by STOP from host
//// The second byte of a I2C write is always interpreted
//// as a register address, and becomes the base register address
//// for all read and write transactions.
//// I2C WRITE:    devAddr, regAddr, data[regAddr], data[regAddr+1],
..... data[regAddr+N]
//// I2C READ:    data[regAddr], data[regAddr+1], .....
data[regAddr+N]
//// Note that when regAddR reaches 255 it will automatically wrap
round to 0
////                                                              ////
//// To Do:                                                       ////
////
////                                                              ////
//// Author(s):                                                   ////
//// - Steve Fielding, sfielding@base2designs.com                 ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// Copyright (C) 2008 Steve Fielding and OPENCORES.ORG          ////
////                                                              ////
//// This source file may be used and distributed without         ////
//// restriction provided that this copyright statement is not    ////
//// removed from the file and that any derivative work contains  ////
//// the original copyright notice and the associated disclaimer. ////
////                                                              ////
//// This source file is free software; you can redistribute it   ////
//// and/or modify it under the terms of the GNU Lesser General   ////
//// Public License as published by the Free Software Foundation; ////
//// either version 2.1 of the License, or (at your option) any   ////
//// later version.                                               ////
////                                                              ////
//// This source is distributed in the hope that it will be       ////
//// useful, but WITHOUT ANY WARRANTY; without even the implied   ////
//// warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR      ////
//// PURPOSE. See the GNU Lesser General Public License for more  ////
//// details.                                                     ////
////                                                              ////
//// You should have received a copy of the GNU Lesser General    ////
//// Public License along with this source; if not, download it   ////
//// from <http://www.opencores.org/lgpl.shtml>                   ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
//
`include "timescale.v"
`include "i2cSlave_define.v"

module serialInterface (
  clearStartStopDet,
```

```
  clk,
  SlaveAddr,
  dataIn,
  dataOut,
  regAddr,
  rst,
  scl,
  sdaIn,
  sdaOut,
  startStopDetState,
  writeEn
);

input   clk;
input   [6:0]SlaveAddr;
input   [7:0]dataIn;
input   rst;
input   scl;
input   sdaIn;
input   [1:0]startStopDetState;
output  clearStartStopDet;
output  [7:0]dataOut;
output  [7:0]regAddr;
output  sdaOut;
output  writeEn;

reg     clearStartStopDet, next_clearStartStopDet;
wire    clk;
wire    [6:0]SlaveAddr;
wire    [7:0]dataIn;
reg     [7:0]dataOut, next_dataOut;
reg     [7:0]regAddr, next_regAddr;
wire    rst;
wire    scl;
wire    sdaIn;
reg     sdaOut, next_sdaOut;
wire    [1:0]startStopDetState;
reg     writeEn, next_writeEn;

// diagram signals declarations
reg  [2:0]bitCnt, next_bitCnt;
reg  [7:0]rxData, next_rxData;
reg  [1:0]streamSt, next_streamSt; //[2:0] change in original is for
16-bit?
reg  [7:0]txData, next_txData;

// BINARY ENCODED state machine: SISt
// State codes definitions:
`define START 4'b0000
`define CHK_RD_WR 4'b0001
`define READ_RD_LOOP 4'b0010
`define READ_WT_HI 4'b0011
`define READ_CHK_LOOP_FIN 4'b0100
`define READ_WT_LO 4'b0101
`define READ_WT_ACK 4'b0110
`define WRITE_WT_LO 4'b0111
`define WRITE_WT_HI 4'b1000
`define WRITE_CHK_LOOP_FIN 4'b1001
`define WRITE_LOOP_WT_LO 4'b1010
`define WRITE_ST_LOOP 4'b1011
`define WRITE_WT_LO2 4'b1100
```

```
`define WRITE_WT_HI2 4'b1101
`define WRITE_CLR_WR 4'b1110
`define WRITE_CLR_ST_STOP 4'b1111

reg [3:0]CurrState_SISt, NextState_SISt;

// Diagram actions (continuous assignments allowed only: assign ...)
// diagram ACTION


// Machine: SISt

// NextState logic (combinatorial)
always @ (startStopDetState or streamSt or scl or txData or bitCnt or
rxData or sdaIn or regAddr or dataIn or sdaOut or writeEn or dataOut
or clearStartStopDet or CurrState_SISt)
begin
  NextState_SISt <= CurrState_SISt;
  // Set default values for outputs and signals
  next_streamSt <= streamSt;
  next_txData <= txData;
  next_rxData <= rxData;
  next_sdaOut <= sdaOut;
  next_writeEn <= writeEn;
  next_dataOut <= dataOut;
  next_bitCnt <= bitCnt;
  next_clearStartStopDet <= clearStartStopDet;
  next_regAddr <= regAddr;
  case (CurrState_SISt)  // synopsys parallel_case full_case
    `START:
    begin
      next_streamSt <= `STREAM_IDLE;
      next_txData <= 8'h00;
      next_rxData <= 8'h00;
      next_sdaOut <= 1'b1;
      next_writeEn <= 1'b0;
      next_dataOut <= 8'h00;
      next_bitCnt <= 3'b000;
      next_clearStartStopDet <= 1'b0;
      NextState_SISt <= `CHK_RD_WR;
    end
    `CHK_RD_WR:
    begin
      if (streamSt == `STREAM_READ)
      begin
        NextState_SISt <= `READ_RD_LOOP;
        next_txData <= dataIn;
        next_regAddr <= regAddr + 1'b1;
        next_bitCnt <= 3'b001;
      end
      else
      begin
        NextState_SISt <= `WRITE_WT_HI;
        next_rxData <= 8'h00;
      end
    end
    `READ_RD_LOOP:
    begin
      if (scl == 1'b0)
      begin
        NextState_SISt <= `READ_WT_HI;
```

```
      next_sdaOut <= txData [7];
      next_txData <= {txData [6:0], 1'b0};
    end
  end
  `READ_WT_HI:
  begin
    if (scl == 1'b1)
    begin
      NextState_SISt <= `READ_CHK_LOOP_FIN;
    end
  end
  `READ_CHK_LOOP_FIN:
  begin
    if (bitCnt == 3'b000)
    begin
      NextState_SISt <= `READ_WT_LO;
    end
    else
    begin
      NextState_SISt <= `READ_RD_LOOP;
      next_bitCnt <= bitCnt + 1'b1;
    end
  end
  `READ_WT_LO:
  begin
    if (scl == 1'b0)
    begin
      NextState_SISt <= `READ_WT_ACK;
      next_sdaOut <= 1'b1;
    end
  end
  `READ_WT_ACK:
  begin
    if (scl == 1'b1)
    begin
      NextState_SISt <= `CHK_RD_WR;
      if (sdaIn == `I2C_NAK)
      next_streamSt <= `STREAM_IDLE;
    end
  end
  `WRITE_WT_LO:
  begin
    if ((scl == 1'b0) && (startStopDetState == `STOP_DET ||
      (streamSt == `STREAM_IDLE && startStopDetState == `NULL_DET)))
    begin
      NextState_SISt <= `WRITE_CLR_ST_STOP;
      case (startStopDetState)
      `NULL_DET:
      next_bitCnt <= bitCnt + 1'b1;
      `START_DET: begin
      next_streamSt <= `STREAM_IDLE;
      next_rxData <= 8'h00;
      end
      default: ;
      endcase
      next_streamSt <= `STREAM_IDLE;
      next_clearStartStopDet <= 1'b1;
    end
    else if (scl == 1'b0)
    begin
      NextState_SISt <= `WRITE_ST_LOOP;
```

i2cSlave/serialInterface.v

```
        case (startStopDetState)
        `NULL_DET:
        next_bitCnt <= bitCnt + 1'b1;
        `START_DET: begin
        next_streamSt <= `STREAM_IDLE;
        next_rxData <= 8'h00;
        end
        default: ;
        endcase
      end
    end
    `WRITE_WT_HI:
    begin
      if (scl == 1'b1)
      begin
        NextState_SISt <= `WRITE_WT_LO;
        next_rxData <= {rxData [6:0], sdaIn};
        next_bitCnt <= 3'b000;
      end
    end
    `WRITE_CHK_LOOP_FIN:
    begin
      if (bitCnt == 3'b111)
      begin
        NextState_SISt <= `WRITE_CLR_WR;
        next_sdaOut <= `I2C_ACK;
        case (streamSt)
        `STREAM_IDLE: begin
        if (rxData[7:1] == SlaveAddr &&      //changed from
'I2C_ADDRESS
        startStopDetState == `START_DET) begin
        if (rxData[0] == 1'b1)
        next_streamSt <= `STREAM_READ;
        else
        next_streamSt <= `STREAM_WRITE_ADDR; //16-bit:
`STREAM_WRITE_ADDR1;
        end
        else
        next_sdaOut <= `I2C_NAK;
        end
        `STREAM_WRITE_ADDR: begin
        next_streamSt <= `STREAM_WRITE_DATA;
        next_regAddr <= rxData;
        end
        `STREAM_WRITE_DATA: begin
        next_dataOut <= rxData;
        next_writeEn <= 1'b1;
        end
        default:
        next_streamSt <= streamSt;
        endcase
      end
      else
      begin
        NextState_SISt <= `WRITE_ST_LOOP;
        next_bitCnt <= bitCnt + 1'b1;
      end
    end
    `WRITE_LOOP_WT_LO:
    begin
      if (scl == 1'b0)
```

```
      begin
        NextState_SISt <= `WRITE_CHK_LOOP_FIN;
      end
    end
    `WRITE_ST_LOOP:
    begin
      if (scl == 1'b1)
      begin
        NextState_SISt <= `WRITE_LOOP_WT_LO;
        next_rxData <= {rxData [6:0], sdaIn};
      end
    end
    `WRITE_WT_LO2:
    begin
      if (scl == 1'b0)
      begin
        NextState_SISt <= `CHK_RD_WR;
        next_sdaOut <= 1'b1;
      end
    end
    `WRITE_WT_HI2:
    begin
      next_clearStartStopDet <= 1'b0;
      if (scl == 1'b1)
      begin
        NextState_SISt <= `WRITE_WT_LO2;
      end
    end
    `WRITE_CLR_WR:
    begin
      if (writeEn == 1'b1)
      next_regAddr <= regAddr + 1'b1;
      next_writeEn <= 1'b0;
      next_clearStartStopDet <= 1'b1;
      NextState_SISt <= `WRITE_WT_HI2;
    end
    `WRITE_CLR_ST_STOP:
    begin
      next_clearStartStopDet <= 1'b0;
      NextState_SISt <= `CHK_RD_WR;
    end
  endcase
end

// Current State Logic (sequential)
always @ (posedge clk)
begin
  if (rst == 1'b1)
    CurrState_SISt <= `START;
  else
    CurrState_SISt <= NextState_SISt;
end

// Registered outputs logic
always @ (posedge clk)
begin
  if (rst == 1'b1)
  begin
    sdaOut <= 1'b1;
    writeEn <= 1'b0;
    dataOut <= 8'h00;
```

i2cSlave/serialInterface.v

```
    clearStartStopDet <= 1'b0;
    // regAddr <=     // Initialization in the reset state or default
value required!!
    streamSt <= `STREAM_IDLE;
    txData <= 8'h00;
    rxData <= 8'h00;
    bitCnt <= 3'b000;
  end
  else
  begin
    sdaOut <= next_sdaOut;
    writeEn <= next_writeEn;
    dataOut <= next_dataOut;
    clearStartStopDet <= next_clearStartStopDet;
    regAddr <= next_regAddr;
    streamSt <= next_streamSt;
    txData <= next_txData;
    rxData <= next_rxData;
    bitCnt <= next_bitCnt;
  end
end

endmodule
```

```
////////////////////////////////////////////////////////////////////////////////
// timescale.v
////////////////////////////////////////////////////////////////////////////////
`timescale 1ns / 1ps
```

Apêndice G

Projeto da firmware do microcontrolador do DAQ

Este apêndice é composto pelo projeto da *firmware* do microcontrolador ESP32, da placa do sistema de aquisição do experimento. O código foi desenvolvido utilizando-se a IDE do Arduino versão 1.8.19 e faz uso das seguintes bibliotecas, que podem ser instaladas por meio da própria IDE:

1. RTClib - Autor: Adafruit - Versão: 2.1.1;
2. Adafruit LSM9DS1 Library - Autor: Adafruit - Versão: 2.1.1;
3. Adafruit BME280 Library - Autor: Adafruit - Versão: 2.2.2;
4. Time - Autor: Paul Stoffregen - Versão: 1.6.1.

As placas ESP32 do autor Espressif Systems, na versão 2.0.11, foram instaladas na IDE por intermédio do URL:
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

```
/*********************************************************************
Company:              CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
Project:              CRE4AT
Model:                CRE4AT ESP32
Arduino Proj. Name:   DAQV3_ESP32Firmware_3Groups
Device:               ESP32S
Author:               Diogo Ayres Rocha (dayres@cbpf.br)
Date:                 18-05-2023
---------------------------------------------------------------------
Module:               MemoryLibrary
Description:          CRE4AT I2C Memory Types and regs definition
---------------------------------------------------------------------
Revision History:
Date       Author             Rev.   Comments
18 May 23  Diogo Ayres Rocha  0000   File Created
04 Ago 23  Diogo Ayres Rocha  0001   Added microSD
18 Ago 23  Diogo Ayres Rocha  0002   Added LED Calibration
18 Sep 23  Diogo Ayres Rocha  0003   Changed LED Calibration Procedure
*********************************************************************/

#include <Wire.h>
#include <HardwareSerial.h>
#include <EEPROM.h>
#include <RTClib.h>
#include <Adafruit_LSM9DS1.h>
#include <Adafruit_BME280.h>
#include <HardwareSerial.h>
#include <Adafruit_PMTK.h>
#include <NMEA_data.h>
#include <Adafruit_GPS.h>
#include <TimeLib.h>
#include <SPI.h>
#include <FS.h>
#include <SD.h>

#define Place "IPANEMA_SiPM"
#define TX_HVPS 35
#define RX_HVPS 16
#define HV_baudrate 38400
#define GPS_baudrate 9600
#define RX_GPS 13
#define TX_GPS 25
#define USBbaudrate 115200
#define MODSsda 21
#define MODSscl 22
#define MODSFreq 400000
#define FPGAsda 27
#define FPGAscl 26
#define FPGAFreq 400000
#define SPTR_SIZE   20
#define BankSelectReg 253
#define AckReg 251
#define FwVerAddr 255
#define LEDreg 15
#define InterruptPin 33
#define DataRegsNum 72
#define i2cMemorySpace 16
#define GPSecho false
#define FileHeader "GPS_timestamp\tRTC_timestamp\tACh0\tACh1\tACh2\
tACh3\tBCh0\tBCh1\tBCh2\tBCh3\tCCh0\tCCh1\tCCh2\tCCh3\tSep0\tA0&1\
tA0&2\tA0&3\tA1&2\tA1&3\tA2&3\tA0&1&2\tA0&1&3\tA0&2&3\tA1&2&3\
```

```
tA0&1&2&3\tB0&1\tB0&2\tB0&3\tB1&2\tB1&3\tB2&3\tB0&1&2\tB0&1&3\tB0&2&3\
tB1&2&3\tB0&1&2&3\tC0&1\tC0&2\tC0&3\tC1&2\tC1&3\tC2&3\tC0&1&2\tC0&1&3\
tC0&2&3\tC1&2&3\tC0&1&2&3\tSep1\tA01&B01\tA01&B12\tA01&B23\tA12&B01\
tA12&B12\tA12&B23\tA23&B01\tA23&B12\tA23&B23\tA01&C01\tA01&C12\
tA01&C23\tA12&C01\tA12&C12\tA12&C23\tA23&C01\tA23&C12\tA23&C23\
tB01&C01\tB01&C12\tB01&C23\tB12&C01\tB12&C12\tB12&C23\tB23&C01\
tB23&C12\tB23&C23\tSep2\tAccX\tAccY\tAccZ\tGyroX\tGyroY\tGyroZ\tMagX\
tMagY\tMagZ\tTemp\tHum\tPress\tHV_Temp\tHV_Voltage\tHV_Current\
tLED_CALIB_Status\n"
const char endMarker = '\n';

bool newData      = false;
bool GetCounters  = false;
bool LSM_enabled  = false;
bool BME_enabled  = false;
bool LED_Calib    = false;
byte LED_FSM      = 0;
byte LED_Freq     = 0;
int LEDFreqVals[6]= {1000, 5000, 10000, 25000, 50000, 100000};
int LED_Status    = 0;
char   *sPtr [SPTR_SIZE];
String UserStr;

RTC_DS3231 rtc;
DateTime RTCtime;
Adafruit_LSM9DS1 lsm = Adafruit_LSM9DS1();
Adafruit_BME280 bme;
Adafruit_Sensor *bme_temp     = bme.getTemperatureSensor();
Adafruit_Sensor *bme_pressure = bme.getPressureSensor();
Adafruit_Sensor *bme_humidity = bme.getHumiditySensor();
sensors_event_t temp_event, pressure_event, humidity_event;
sensors_event_t a, m, g, temp;

unsigned long FileOpenTime;
String FileName;

struct {
    byte FPGAMemory[i2cMemorySpace];
    bool RTCautoAjust=false;
  } CRE4ATSettings;

TwoWire FPGAi2c = TwoWire(1);
HardwareSerial HVPSSerial(2);
HardwareSerial GPSSerial(1);


Adafruit_GPS GPS(&GPSSerial);

//*************** LSM9DS1 Configuration Function *****************

void setupSensorLSM()
{
  // 1.) Set the accelerometer range
  lsm.setupAccel(lsm.LSM9DS1_ACCELRANGE_2G);
  //lsm.setupAccel(lsm.LSM9DS1_ACCELRANGE_4G);
  //lsm.setupAccel(lsm.LSM9DS1_ACCELRANGE_8G);
  //lsm.setupAccel(lsm.LSM9DS1_ACCELRANGE_16G);

  // 2.) Set the magnetometer sensitivity
  lsm.setupMag(lsm.LSM9DS1_MAGGAIN_4GAUSS);
  //lsm.setupMag(lsm.LSM9DS1_MAGGAIN_8GAUSS);
```

```
  //lsm.setupMag(lsm.LSM9DS1_MAGGAIN_12GAUSS);
  //lsm.setupMag(lsm.LSM9DS1_MAGGAIN_16GAUSS);

  // 3.) Setup the gyroscope
  lsm.setupGyro(lsm.LSM9DS1_GYROSCALE_245DPS);
  //lsm.setupGyro(lsm.LSM9DS1_GYROSCALE_500DPS);
  //lsm.setupGyro(lsm.LSM9DS1_GYROSCALE_2000DPS);
}

//******** Hamamatsu HV Power Supply Communication Functions *********

String getChecksum(String str) {
    uint8_t checksum = 0;
    int i = 0;
    while (i < str.length()) {
        char c = str.charAt(i);
        checksum += c;
        if (str.charAt(i) == 0x03)
            break;
        i++;
    }
    String checkStr = String(checksum, HEX);
    String checkStr2 = "";
    int len_stuff = 2 - checkStr.length();
    for (int i = 0; i < len_stuff; i++) {
        checkStr2.concat('0');
    }
    checkStr2.concat(checkStr);
    checkStr2.toUpperCase();
    return checkStr2;
}

String WriteReadDCDC(String command, bool silent=false, float
HV2Set=0)
{
  char str[45];
  char c;
  char STX = 0x02;
  char ETX = 0x03;
  char CR = 0x0D;
  String receivedStr = String("");
  String message = String("");

  message.concat(STX);
  if (command == "HST" && HV2Set != 0)
  {
    Serial.printf("Found %s - Setting HV to %.2f\n", command, HV2Set);
    message.concat(command);
    message.concat("0000000004300430");
    int ConvHV2Set = round(HV2Set/0.001812);
    String tmpVal  = String(ConvHV2Set,HEX);
    tmpVal.toUpperCase();
    message.concat(tmpVal);
    message.concat("B7D7");
  }
  else
  {
    message.concat(command);
  }
  message.concat(ETX);
```

```
  String checksum = getChecksum(message);

  message.concat(checksum);
  message.concat(CR);

  message.toCharArray(str, 45);
  HVPSSerial.write(str);
  receivedStr=String("");
  int n = 0;
  while (!HVPSSerial.available() && n<10)
  {
    n++;
    delay(5);
  }
  if(HVPSSerial.available())
  {
    c = HVPSSerial.read();
    receivedStr.concat(c);
    while (c!=0x0D)
    {
      while (!HVPSSerial.available());
      c = HVPSSerial.read();
      receivedStr.concat(c);
    }
    if (!silent)
    {
      Serial.print("RAW: Message Sent: ");
      for (int n=0; n<message.length(); n++)
        Serial.printf("0x%02X ",message[n]);
      Serial.println("");
      Serial.print("RAW: Message received: ");
      for (int n=0; n<receivedStr.length(); n++)
        Serial.printf("0x%02X ",receivedStr[n]);
      Serial.println("");
    }
    if (command=="HPO" && receivedStr.length()==28)
    {
      String tmp=receivedStr.substring(12,16);
      char c1[tmp.length() + 1];
      tmp.toCharArray(c1, tmp.length() + 1);
      float Voltage=strtol(c1, NULL, 16)*1.812*0.001;
      tmp=receivedStr.substring(16,20);
      char c2[tmp.length() + 1];
      tmp.toCharArray(c2, tmp.length() + 1);
      float Current=strtol(c2, NULL, 16)*5.194*0.001;
      tmp=receivedStr.substring(20,24);
      char c3[tmp.length() + 1];
      tmp.toCharArray(c3, tmp.length() + 1);
      float MMPCTemp=((strtol(c3, NULL, 16)*1.907*0.00001)-1.035)/(-
5.5*0.001);
      tmp=receivedStr.substring(4,8);
      char c4[tmp.length() + 1];
      tmp.toCharArray(c4, tmp.length() + 1);
      int Status=strtol(c4, NULL, 16);
      tmp=receivedStr.substring(8,12);
      char c5[tmp.length() + 1];
      tmp.toCharArray(c5, tmp.length() + 1);
      int Reserve=strtol(c5, NULL, 16);
      Serial.print("INFO: HVPS DATA:\n\tStatus=");
      Serial.print(Status,BIN);
```

```
      Serial.printf("\n\tReserve=%d\n\tVoltage=%.3fV\n\tCurrent=
%.3fmA\n\tTemperature=%.3f°C\n\tRAW Message: ", Reserve, Voltage,
Current, MMPCTemp);
      Serial.print(receivedStr+"\n");
    }
    else if (command=="HGV" && receivedStr.length()==12)
    {
      String tmp=receivedStr.substring(4,8);
      char c1[tmp.length() + 1];
      tmp.toCharArray(c1, tmp.length() + 1);
      float Voltage=strtol(c1, NULL, 16)*1.812*0.001;
      if (!silent)
        Serial.printf("INFO: %.3f\n",Voltage);
      else
      {
        return String(Voltage,3);
      }
    }
    else if (command=="HGC" && receivedStr.length()==12)
    {
      String tmp=receivedStr.substring(4,8);
      char c[tmp.length() + 1];
      tmp.toCharArray(c, tmp.length() + 1);
      float Current=strtol(c, NULL, 16)*5.194*0.001;
      if (!silent)
        Serial.printf("INFO: %.3f\n",Current);
      else
      {
        return String(Current,3);
      }
    }
    else if (command=="HGT" && receivedStr.length()==12)
    {
      String tmp=receivedStr.substring(4,8);
      char c[tmp.length() + 1];
      tmp.toCharArray(c, tmp.length() + 1);
      float MMPCTemp=((strtol(c, NULL, 16)*1.907*0.00001)-1.035)/(-
5.5*0.001);
      if (!silent)
        Serial.printf("INFO: %.3f\n",MMPCTemp);
      else
      {
        return String(MMPCTemp,3);
      }

    }
    else if (command=="HGS" && receivedStr.length()==12)
    {
      String tmp=receivedStr.substring(4,8);
      char c[tmp.length() + 1];
      tmp.toCharArray(c, tmp.length() + 1);
      int Status=strtol(c, NULL, 16);
      Serial.print("INFO: ");
      Serial.print(Status,BIN);
      Serial.println("");
    }
    else if (command=="HFI" && receivedStr.length()==51)
    {
      Serial.print("INFO: HVPS Firmware Information:\n\tDevice Name:
");
      Serial.print(receivedStr.substring(4,20));
```

```
      Serial.print("\n\tVersion: ");
      Serial.print(receivedStr.substring(20,36));
      Serial.print("\n\tBuild date: ");
      Serial.print(receivedStr.substring(36,47));
      Serial.println("");
    }
    else if (command=="HGN" && receivedStr.length()==24)
    {
      if (!silent)
      {
        Serial.print("INFO: HVPS Serial Number:\n\t");
        Serial.print(receivedStr.substring(4,20));
        Serial.println("");
      }
      else
      {
        return receivedStr.substring(4,20);
      }
    }
    else if (command=="HRC" && receivedStr.length()==12)
    {
      Serial.print("INFO: ");
      Serial.println(receivedStr.substring(4,8));
    }
    else
      Serial.println("RAW: Returned string: "+receivedStr);
  }
  return String("");
}

//*************** MicroSD File and Folder Functions
***********************

void listDir(fs::FS &fs, const char * dirname, uint8_t levels){
  Serial.printf("Listing directory: %s\n", dirname);

  File root = fs.open(dirname);
  if(!root){
    Serial.println("Failed to open directory");
    return;
  }
  if(!root.isDirectory()){
    Serial.println("Not a directory");
    return;
  }

  File file = root.openNextFile();
  while(file){
    if(file.isDirectory()){
      Serial.print("  DIR : ");
      Serial.println(file.name());
      if(levels){
        listDir(fs, file.name(), levels -1);
      }
    } else {
      Serial.print("  FILE: ");
      Serial.print(file.name());
      Serial.print("  SIZE: ");
      Serial.println(file.size());
    }
    file = root.openNextFile();
```

```
  }
}

void createDir(fs::FS &fs, const char * path){
  Serial.printf("Creating Dir: %s\n", path);
  if(fs.mkdir(path)){
    Serial.println("Dir created");
  } else {
    Serial.println("mkdir failed");
  }
}

void removeDir(fs::FS &fs, const char * path){
  Serial.printf("Removing Dir: %s\n", path);
  if(fs.rmdir(path)){
    Serial.println("Dir removed");
  } else {
    Serial.println("rmdir failed");
  }
}

void readFile(fs::FS &fs, const char * path){
  Serial.printf("Reading file: %s\n", path);

  File file = fs.open(path);
  if(!file){
    Serial.println("Failed to open file for reading");
    return;
  }

  Serial.print("Read from file: ");
  while(file.available()){
    Serial.write(file.read());
  }
  file.close();
}

void writeFile(fs::FS &fs, const char * path, const char * message){
  //Serial.printf("Writing file: %s\n", path);
  File file = fs.open(path, FILE_WRITE);
  if(!file){
    Serial.println("Failed to open file for writing");
    return;
  }
  if(file.print(message)){
    //Serial.println("File written");
  } else {
    Serial.println("Write failed");
  }
  file.close();
}

void appendFile(fs::FS &fs, const char * path, const char * message){
  //Serial.printf("Appending to file: %s\n", path);
  File file = fs.open(path, FILE_APPEND);
  if(!file){
    Serial.println("Failed to open file for appending");
    return;
  }
  if(file.print(message)){
      //Serial.println("Message appended");
```

```
  } else {
    Serial.println("Append failed");
  }
  file.close();
}

void renameFile(fs::FS &fs, const char * path1, const char * path2){
  Serial.printf("Renaming file %s to %s\n", path1, path2);
  if (fs.rename(path1, path2)) {
    Serial.println("File renamed");
  } else {
    Serial.println("Rename failed");
  }
}

void deleteFile(fs::FS &fs, const char * path){
  Serial.printf("Deleting file: %s\n", path);
  if(fs.remove(path)){
    Serial.println("File deleted");
  } else {
    Serial.println("Delete failed");
  }
}

//*************** FPGA Communication Functions ******************

void writeFPGAReg(byte reg, byte val, bool printMessages=true)
{
  byte c;
  do
  {
    FPGAi2c.beginTransmission(0x01);
    FPGAi2c.write(BankSelectReg);
    FPGAi2c.write(1);
    FPGAi2c.endTransmission();
    delay(5);
    FPGAi2c.beginTransmission(0x01);
    FPGAi2c.write(BankSelectReg);
    FPGAi2c.endTransmission();
    FPGAi2c.requestFrom(0x01, 1);
    while(FPGAi2c.available()) {
      c = FPGAi2c.read();
    }
  }while(c!=1);
  if (printMessages)
  {
    Serial.print("RAW: Reg2Write: ");
    Serial.printf("0x%02X\n",reg);
    Serial.print("RAW: Val2Write: ");
    Serial.printf("0x%02X\n",val);
  }
  FPGAi2c.beginTransmission(0x01);
  FPGAi2c.write(reg);
  FPGAi2c.write(val);
  FPGAi2c.endTransmission();
  if (printMessages)
  {
    delay(1);
    FPGAi2c.beginTransmission(0x01);
    FPGAi2c.write(reg);
    FPGAi2c.endTransmission();
```

```
    FPGAi2c.requestFrom(0x01, 1);
    while(FPGAi2c.available()) {
      c = FPGAi2c.read();
    }
    if (c==val)  Serial.printf("RAW: Value 0x%02X wrote on reg 0x%02X
successfully\n",val,reg);
    else Serial.printf("RAW: Problem writting register on FPGA... Try
again!\n",val,reg);
  }
}
void readFPGAReg(byte reg)
{
  byte c;
  do
  {
    FPGAi2c.beginTransmission(0x01);
    FPGAi2c.write(BankSelectReg);
    FPGAi2c.write(1);
    FPGAi2c.endTransmission();
    delay(5);
    FPGAi2c.beginTransmission(0x01);
    FPGAi2c.write(BankSelectReg);
    FPGAi2c.endTransmission();
    FPGAi2c.requestFrom(0x01, 1);
    while(FPGAi2c.available()) {
      c = FPGAi2c.read();
    }
  }while(c!=1);
  FPGAi2c.beginTransmission(0x01);
  FPGAi2c.write(reg);
  FPGAi2c.endTransmission();
  FPGAi2c.requestFrom(0x01, 1);
  Serial.print("INFO: FPGA REGDATA: ");
  while(FPGAi2c.available()) {
    c = FPGAi2c.read();    // Receive a byte as character
    Serial.printf("0x%02X\n",c);         // Print the character
  }
}

void GetFPGACounters()
{
  RTCtime = rtc.now();
  uint32_t counter;
  byte c;
  String DataLine=String(now());
  if (GPS.milliseconds < 10) {
    DataLine+="00";
  } else if (GPS.milliseconds > 9 && GPS.milliseconds < 100) {
    DataLine+="0";
  }
  DataLine+=String(GPS.milliseconds);
  DataLine+="\t";
  DataLine+=String(RTCtime.unixtime())+"\t";
  Serial.print("DATA: ");
  do
  {
    FPGAi2c.beginTransmission(0x01);
    FPGAi2c.write(BankSelectReg);
    FPGAi2c.write(0);
    FPGAi2c.endTransmission();
    delay(5);
```

```
  FPGAi2c.beginTransmission(0x01);
  FPGAi2c.write(BankSelectReg);
  FPGAi2c.endTransmission();
  FPGAi2c.requestFrom(0x01, 1);
  while(FPGAi2c.available()) {
    c = FPGAi2c.read();
  }
}while(c!=0);

for (int channels=0; channels<DataRegsNum; channels++)
{
  FPGAi2c.beginTransmission(0x01);
  FPGAi2c.write(3*channels);
  FPGAi2c.endTransmission();
  FPGAi2c.requestFrom(0x01, 3);
  byte c2 = FPGAi2c.read();
  byte c1 = FPGAi2c.read();
  byte c0 = FPGAi2c.read();
  counter = (c2<<16)+(c1<<8)+c0;
  if (channels==DataRegsNum-1)
  {
    DataLine+=String(counter)+"\t";
    DataLine+="*\t";
  }
  else if (channels==11 || channels==44)
  {
    DataLine+=String(counter)+"\t";
    DataLine+="*\t";
  }
  else
  {
    DataLine+=String(counter)+"\t";
  }
}
FPGAi2c.beginTransmission(0x01);
FPGAi2c.write(AckReg);
FPGAi2c.write(1);
FPGAi2c.endTransmission();
if (LSM_enabled)
{
  lsm.read();
  lsm.getEvent(&a, &m, &g, &temp);
  DataLine+=String(a.acceleration.x)+"\t";
  DataLine+=String(a.acceleration.y)+"\t";
  DataLine+=String(a.acceleration.z)+"\t";
  DataLine+=String(g.gyro.x)+"\t";
  DataLine+=String(g.gyro.y)+"\t";
  DataLine+=String(g.gyro.z)+"\t";
  DataLine+=String(m.magnetic.x)+"\t";
  DataLine+=String(m.magnetic.y)+"\t";
  DataLine+=String(m.magnetic.z)+"\t";
}
else
{
  DataLine+="\t\t\t\t\t\t\t\t\t";
}
if (BME_enabled)
{
  bme_temp->getEvent(&temp_event);
  bme_pressure->getEvent(&pressure_event);
  bme_humidity->getEvent(&humidity_event);
```

```
    DataLine+=String(temp_event.temperature)+"\t";
    DataLine+=String(humidity_event.relative_humidity)+"\t";
    DataLine+=String(pressure_event.pressure)+"\t";
  }
  else
  {
    DataLine+="\t\t\t";
  }
  DataLine+=WriteReadDCDC("HGT",true)+"\t";
  DataLine+=WriteReadDCDC("HGV",true)+"\t";
  DataLine+=WriteReadDCDC("HGC",true)+"\t";
  DataLine+=String((int)LED_Status);
  DataLine+="\n";
  Serial.print(DataLine);
  appendFile(SD, FileName.c_str(), DataLine.c_str());
}

// ISR Function - Called when FPGA generate the interruption
void IRAM_ATTR ISR()
{
  GetCounters=true;
}

//************** FSM User Control Commands Functions ***************

int separate (String str, char   **p, int size, const char *sep=";")
{
    int  n;
    char s [255];
    strcpy (s, str.c_str ());
    *p++ = strtok (s, sep);
    for (n = 1; NULL != (*p++ = strtok (NULL, sep)); n++)
        if (size == n)
            break;
    return n;
}

void FSM()
{
  char rc;
  while (Serial.available() > 0 && newData == false) {
      rc = Serial.read();
      if (rc != endMarker) {
          UserStr += rc;
      }
      else {
          newData = true;
          Serial.print("RAW: User Received String: ");
          Serial.println(UserStr);
          int N = separate (UserStr, sPtr, SPTR_SIZE);
          if (String(sPtr[0])=="HV" && N==2) String received =
WriteReadDCDC(String(sPtr[1]));
          else if (String(sPtr[0])=="HV" && String(sPtr[1])=="HST" &&
N==3) String received = WriteReadDCDC(String(sPtr[1]), false,
String(sPtr[2]).toFloat());
          else if (String(sPtr[0])=="ESP-RESTART") ESP.restart();
          else if(String(sPtr[0])=="REG" && N==3)
          {
            byte reg = String(sPtr[1]).toInt();
            byte val = String(sPtr[2]).toInt();
            CRE4ATSettings.FPGAMemory[int(reg)] = val;
```

```
            writeFPGAReg(reg,val);
          }
          else if(String(sPtr[0])=="SAVEDATA" && N==1)
          {
            EEPROM.put(0, CRE4ATSettings);
            EEPROM.commit();
            Serial.println("RAW: FPGA Bank Values Saved to EEPROM!");
          }
          else if(String(sPtr[0])=="RTC" && N==7)
rtc.adjust(DateTime(String(sPtr[1]).toInt(), String(sPtr[2]).toInt(),
String(sPtr[3]).toInt(), String(sPtr[4]).toInt(),
String(sPtr[5]).toInt(), String(sPtr[6]).toInt()));
          else if(String(sPtr[0])=="REG" && N==2)
readFPGAReg(String(sPtr[1]).toInt());
          else if(String(sPtr[0])=="GETDATA" && N==1)
GetFPGACounters();
          else if (String(sPtr[0])=="StartLEDCalib" && N==1)
          {
            LED_FSM = 0;
            LED_Calib = true;
            Serial.println("LED_INFO: LED Calibration Started!");
          }
          else Serial.println("\n\nWrong Data Format!\nType:\t'REG;
[CONFIG]';\t[CONFIG]: 'REGNum' to read or 'REGNum;Value' to write\n\
tReg List:\n\t\t\t\t[0] -> Bit 0-3(LED0-3 On[1]/Off[0]) - Bit 4
(Trigger Out On[1]/Off[0])\n\t\t\t\t[1-4] -> LED 0-3 Freq Selection\n\
t\t\t\t\t\t0 -> 1kHz\n\t\t\t\t\t\t1 -> 5kHz\n\t\t\t\t\t\t2 -> 10kHz\n\
t\t\t\t\t\t3 -> 25kHz\n\t\t\t\t\t\t4 -> 50kHz\n\t\t\t\t\t\t5 ->
100kHz\n\t\t\t\t[5] -> Trigger Freq Selection\n\t\t\t\t[6-9] -> LED 0-
3 Pulse Width (Steps 20ns)\n\t\t\t\t[10] -> Trigger Pulse Width\n\t\t\
t\t[11] -> LED Minimum Pulse Width\n\t\t\t\t[12] -> Latch Input Pulses
Width Selection (Step 20ns)\n\t\t\t\t[13-14] -> Counter time in
seconds (Reg 13 - Most significant Byte)\n\
tRTC;year;month;day;hour;minute;second to set RTC time\n\t'GETDATA' to
force FPGA count\n\t'SAVEDATA' to save FPGA CONFIG REG Bank in the
memory\n\t'HV;[COMMAND]'\t\t-> Userful Commands:\n\t\t\t\tHPO -> Check
all info from HV PS\n\t\t\t\tHGV -> Get Voltage read\n\t\t\t\tHGC ->
Get Current read\n\t\t\t\tHGT -> Get Temperature read\n\t\t\t\tHON ->
Turn On output\n\t\t\t\tHOF -> Turn OFF output\n\t\t\t\tHST;Value ->
Set HV Output Value\n\t'StartLEDCalib' to force LED Calibration
Procedure\n");
          newData = false;
          UserStr = String("");
      }
  }
}

void setup()
{
  delay(5000);

  // Initialization of USB PC connection
  Serial.begin(USBbaudrate);
  Serial.println("INFO: Starting DAQ V3 - CBPF...");

  // Setting interrupt pin to input
  pinMode(InterruptPin,INPUT);

  // FPGA I2C connection
  FPGAi2c.begin(FPGAsda, FPGAscl, uint32_t(FPGAFreq));
```

```
  // GPS Configuration
  //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
  GPS.sendCommand(PGCMD_ANTENNA);

  // EEPROM initialization
  EEPROM.begin(256);
  EEPROM.get(0,CRE4ATSettings);

  // RTC initialization
  if (!rtc.begin())
  {
    Serial.println("ERROR: Couldn't find RTC...");
  }

  // LSM9DS1 initialization
  if (!lsm.begin())
  {
    Serial.println("ERROR: Problem Initializing LSM9DS1 (Acc,Gyro,Mag
Sensors)...");
  }
  else
  {
    LSM_enabled=true;
    setupSensorLSM();
  }
  if (!bme.begin())
  {
    Serial.println("ERROR: Problem Initializing BME280 (Temp,Press,Hum
Sensors)...");
  }
  else
  {
    BME_enabled = true;
  }

  // FPGA check Firmware
  FPGAi2c.beginTransmission(0x01);
  FPGAi2c.write(FwVerAddr);
  FPGAi2c.endTransmission();
  FPGAi2c.requestFrom(0x01, 1);
  Serial.print("INFO: Firmware Ver: ");
  while(FPGAi2c.available()) {
    int c = FPGAi2c.read();    // Receive a byte as character
    Serial.printf("0x%02X\n",c);         // Print the character
  }

  // FPGA Memory Bank Initialization
  for (int i=0;i<i2cMemorySpace;i++)
    writeFPGAReg(i,CRE4ATSettings.FPGAMemory[i],false);

  // Start Counting process
  FPGAi2c.beginTransmission(0x01);
  FPGAi2c.write(AckReg);
  FPGAi2c.write(1);
  FPGAi2c.endTransmission();

  //Initializing microSD Card
   if(!SD.begin())
   {
```

```
      Serial.println("ERROR: Card Mount Failed");
    }

  // Initial File Creation
  RTCtime = rtc.now();
  unsigned long t=RTCtime.unixtime();
  char buff[64];
  sprintf(buff, "/CRE4AT_%s_%04u%02u%02u_%02uh%02um%02us.txt",Place,
(int)year(t),(int)month(t),int(day(t)),(int)hour(t),(int)minute(t),
(int)second(t));
  FileOpenTime=t;
  FileName=String(buff);
  writeFile(SD, FileName.c_str(), FileHeader);
  Serial.print("FILE_INFO: File oppened to save data: ");
  Serial.println(FileName);

  delay(1000);

  // Hamamatsu Serial connection
  HVPSSerial.begin(HV_baudrate, SERIAL_8E1, TX_HVPS, RX_HVPS);

  // GPS Serial Connection
  GPSSerial.begin(GPS_baudrate, SERIAL_8N1, TX_GPS, RX_GPS);

  delay(4000);

  // Interrupt Initialization
  attachInterrupt(InterruptPin, ISR, FALLING);
}

void loop()
{
  // GPS Parse
  char c = GPS.read();
  if (GPSecho)
    Serial.print(c);
  if (GPS.newNMEAreceived())
  {
    if (!GPS.parse(GPS.lastNMEA()))
      return;
    setTime(GPS.hour, GPS.minute, GPS.seconds, GPS.day, GPS.month,
GPS.year);
  }

  // Getting FPGA Counters and LED Calibration
  if (GetCounters)
  {
    GetFPGACounters();
    GetCounters=false;
    if (LED_Calib)
    {
      switch(LED_FSM)
      {
        case 0:
          LED_Freq = 0;
          writeFPGAReg(1,LED_Freq,false);
          writeFPGAReg(2,LED_Freq,false);
          writeFPGAReg(3,LED_Freq,false);
          writeFPGAReg(4,LED_Freq,false);
          writeFPGAReg(0,0b0001,false);
          LED_Status = 1;
```

```
  break;
case 1:
  LED_Status = LEDFreqVals[LED_Freq]+2;
  break;
case 2:
  writeFPGAReg(0,0b0010,false);
  LED_Status = 1;
  break;
case 3:
  LED_Status = LEDFreqVals[LED_Freq]+3;
  break;
case 4:
  writeFPGAReg(0,0b0100,false);
  LED_Status = 1;
  break;
case 5:
  LED_Status = LEDFreqVals[LED_Freq]+4;
  break;
case 6:
  writeFPGAReg(0,0b1000,false);
  LED_Status = 1;
  break;
case 7:
  LED_Status = LEDFreqVals[LED_Freq]+5;
  break;
case 8:
  writeFPGAReg(0,0b0011,false);
  LED_Status = 1;
  break;
case 9:
  LED_Status = LEDFreqVals[LED_Freq]+6;
  break;
case 10:
  writeFPGAReg(0,0b0101,false);
  LED_Status = 1;
  break;
case 11:
  LED_Status = LEDFreqVals[LED_Freq]+7;
  break;
case 12:
  writeFPGAReg(0,0b1001,false);
  LED_Status = 1;
  break;
case 13:
  LED_Status = LEDFreqVals[LED_Freq]+8;
  break;
case 14:
  writeFPGAReg(0,0b0110,false);
  LED_Status = 1;
  break;
case 15:
  LED_Status = LEDFreqVals[LED_Freq]+9;
  break;
case 16:
  writeFPGAReg(0,0b1010,false);
  LED_Status = 1;
  break;
case 17:
  LED_Status = LEDFreqVals[LED_Freq]+10;
  break;
case 18:
```

```
    writeFPGAReg(0,0b1100,false);
    LED_Status = 1;
    break;
  case 19:
    LED_Status = LEDFreqVals[LED_Freq]+11;
    break;
  case 20:
    writeFPGAReg(0,0b0111,false);
    LED_Status = 1;
    break;
  case 21:
    LED_Status = LEDFreqVals[LED_Freq]+12;
    break;
  case 22:
    writeFPGAReg(0,0b1011,false);
    LED_Status = 1;
    break;
  case 23:
    LED_Status = LEDFreqVals[LED_Freq]+13;
    break;
  case 24:
    writeFPGAReg(0,0b1101,false);
    LED_Status = 1;
    break;
  case 25:
    LED_Status = LEDFreqVals[LED_Freq]+14;
    break;
  case 26:
    writeFPGAReg(0,0b1110,false);
    LED_Status = 1;
    break;
  case 27:
    LED_Status = LEDFreqVals[LED_Freq]+15;
    break;
  case 28:
    writeFPGAReg(0,0b1111,false);
    LED_Status = 1;
    break;
  case 29:
    LED_Status = LEDFreqVals[LED_Freq]+16;
    break;
  case 30:
    LED_Freq = 1;
    writeFPGAReg(1,LED_Freq,false);
    writeFPGAReg(2,LED_Freq,false);
    writeFPGAReg(3,LED_Freq,false);
    writeFPGAReg(4,LED_Freq,false);
    writeFPGAReg(0,0b0001,false);
    LED_Status = 1;
    break;
  case 31:
    LED_Status = LEDFreqVals[LED_Freq]+2;
    break;
  case 32:
    writeFPGAReg(0,0b0010,false);
    LED_Status = 1;
    break;
  case 33:
    LED_Status = LEDFreqVals[LED_Freq]+3;
    break;
  case 34:
```

```
    writeFPGAReg(0,0b0100,false);
    LED_Status = 1;
    break;
  case 35:
    LED_Status = LEDFreqVals[LED_Freq]+4;
    break;
  case 36:
    writeFPGAReg(0,0b1000,false);
    LED_Status = 1;
    break;
  case 37:
    LED_Status = LEDFreqVals[LED_Freq]+5;
    break;
  case 38:
    writeFPGAReg(0,0b0011,false);
    LED_Status = 1;
    break;
  case 39:
    LED_Status = LEDFreqVals[LED_Freq]+6;
    break;
  case 40:
    writeFPGAReg(0,0b0101,false);
    LED_Status = 1;
    break;
  case 41:
    LED_Status = LEDFreqVals[LED_Freq]+7;
    break;
  case 42:
    writeFPGAReg(0,0b1001,false);
    LED_Status = 1;
    break;
  case 43:
    LED_Status = LEDFreqVals[LED_Freq]+8;
    break;
  case 44:
    writeFPGAReg(0,0b0110,false);
    LED_Status = 1;
    break;
  case 45:
    LED_Status = LEDFreqVals[LED_Freq]+9;
    break;
  case 46:
    writeFPGAReg(0,0b1010,false);
    LED_Status = 1;
    break;
  case 47:
    LED_Status = LEDFreqVals[LED_Freq]+10;
    break;
  case 48:
    writeFPGAReg(0,0b1100,false);
    LED_Status = 1;
    break;
  case 49:
    LED_Status = LEDFreqVals[LED_Freq]+11;
    break;
  case 50:
    writeFPGAReg(0,0b0111,false);
    LED_Status = 1;
    break;
  case 51:
    LED_Status = LEDFreqVals[LED_Freq]+12;
```

```
  break;
case 52:
  writeFPGAReg(0,0b1011,false);
  LED_Status = 1;
  break;
case 53:
  LED_Status = LEDFreqVals[LED_Freq]+13;
  break;
case 54:
  writeFPGAReg(0,0b1101,false);
  LED_Status = 1;
  break;
case 55:
  LED_Status = LEDFreqVals[LED_Freq]+14;
  break;
case 56:
  writeFPGAReg(0,0b1110,false);
  LED_Status = 1;
  break;
case 57:
  LED_Status = LEDFreqVals[LED_Freq]+15;
  break;
case 58:
  writeFPGAReg(0,0b1111,false);
  LED_Status = 1;
  break;
case 59:
  LED_Status = LEDFreqVals[LED_Freq]+16;
  break;
case 60:
  LED_Freq = 2;
  writeFPGAReg(1,LED_Freq,false);
  writeFPGAReg(2,LED_Freq,false);
  writeFPGAReg(3,LED_Freq,false);
  writeFPGAReg(4,LED_Freq,false);
  writeFPGAReg(0,0b0001,false);
  LED_Status = 1;
  break;
case 61:
  LED_Status = LEDFreqVals[LED_Freq]+2;
  break;
case 62:
  writeFPGAReg(0,0b0010,false);
  LED_Status = 1;
  break;
case 63:
  LED_Status = LEDFreqVals[LED_Freq]+3;
  break;
case 64:
  writeFPGAReg(0,0b0100,false);
  LED_Status = 1;
  break;
case 65:
  LED_Status = LEDFreqVals[LED_Freq]+4;
  break;
case 66:
  writeFPGAReg(0,0b1000,false);
  LED_Status = 1;
  break;
case 67:
  LED_Status = LEDFreqVals[LED_Freq]+5;
```

```
  break;
case 68:
  writeFPGAReg(0,0b0011,false);
  LED_Status = 1;
  break;
case 69:
  LED_Status = LEDFreqVals[LED_Freq]+6;
  break;
case 70:
  writeFPGAReg(0,0b0101,false);
  LED_Status = 1;
  break;
case 71:
  LED_Status = LEDFreqVals[LED_Freq]+7;
  break;
case 72:
  writeFPGAReg(0,0b1001,false);
  LED_Status = 1;
  break;
case 73:
  LED_Status = LEDFreqVals[LED_Freq]+8;
  break;
case 74:
  writeFPGAReg(0,0b0110,false);
  LED_Status = 1;
  break;
case 75:
  LED_Status = LEDFreqVals[LED_Freq]+9;
  break;
case 76:
  writeFPGAReg(0,0b1010,false);
  LED_Status = 1;
  break;
case 77:
  LED_Status = LEDFreqVals[LED_Freq]+10;
  break;
case 78:
  writeFPGAReg(0,0b1100,false);
  LED_Status = 1;
  break;
case 79:
  LED_Status = LEDFreqVals[LED_Freq]+11;
  break;
case 80:
  writeFPGAReg(0,0b0111,false);
  LED_Status = 1;
  break;
case 81:
  LED_Status = LEDFreqVals[LED_Freq]+12;
  break;
case 82:
  writeFPGAReg(0,0b1011,false);
  LED_Status = 1;
  break;
case 83:
  LED_Status = LEDFreqVals[LED_Freq]+13;
  break;
case 84:
  writeFPGAReg(0,0b1101,false);
  LED_Status = 1;
  break;
```

```
case 85:
  LED_Status = LEDFreqVals[LED_Freq]+14;
  break;
case 86:
  writeFPGAReg(0,0b1110,false);
  LED_Status = 1;
  break;
case 87:
  LED_Status = LEDFreqVals[LED_Freq]+15;
  break;
case 88:
  writeFPGAReg(0,0b1111,false);
  LED_Status = 1;
  break;
case 89:
  LED_Status = LEDFreqVals[LED_Freq]+16;
  break;
case 90:
  LED_Freq = 3;
  writeFPGAReg(1,LED_Freq,false);
  writeFPGAReg(2,LED_Freq,false);
  writeFPGAReg(3,LED_Freq,false);
  writeFPGAReg(4,LED_Freq,false);
  writeFPGAReg(0,0b0001,false);
  LED_Status = 1;
  break;
case 91:
  LED_Status = LEDFreqVals[LED_Freq]+2;
  break;
case 92:
  writeFPGAReg(0,0b0010,false);
  LED_Status = 1;
  break;
case 93:
  LED_Status = LEDFreqVals[LED_Freq]+3;
  break;
case 94:
  writeFPGAReg(0,0b0100,false);
  LED_Status = 1;
  break;
case 95:
  LED_Status = LEDFreqVals[LED_Freq]+4;
  break;
case 96:
  writeFPGAReg(0,0b1000,false);
  LED_Status = 1;
  break;
case 97:
  LED_Status = LEDFreqVals[LED_Freq]+5;
  break;
case 98:
  writeFPGAReg(0,0b0011,false);
  LED_Status = 1;
  break;
case 99:
  LED_Status = LEDFreqVals[LED_Freq]+6;
  break;
case 100:
  writeFPGAReg(0,0b0101,false);
  LED_Status = 1;
  break;
```

```
case 101:
  LED_Status = LEDFreqVals[LED_Freq]+7;
  break;
case 102:
  writeFPGAReg(0,0b1001,false);
  LED_Status = 1;
  break;
case 103:
  LED_Status = LEDFreqVals[LED_Freq]+8;
  break;
case 104:
  writeFPGAReg(0,0b0110,false);
  LED_Status = 1;
  break;
case 105:
  LED_Status = LEDFreqVals[LED_Freq]+9;
  break;
case 106:
  writeFPGAReg(0,0b1010,false);
  LED_Status = 1;
  break;
case 107:
  LED_Status = LEDFreqVals[LED_Freq]+10;
  break;
case 108:
  writeFPGAReg(0,0b1100,false);
  LED_Status = 1;
  break;
case 109:
  LED_Status = LEDFreqVals[LED_Freq]+11;
  break;
case 110:
  writeFPGAReg(0,0b0111,false);
  LED_Status = 1;
  break;
case 111:
  LED_Status = LEDFreqVals[LED_Freq]+12;
  break;
case 112:
  writeFPGAReg(0,0b1011,false);
  LED_Status = 1;
  break;
case 113:
  LED_Status = LEDFreqVals[LED_Freq]+13;
  break;
case 114:
  writeFPGAReg(0,0b1101,false);
  LED_Status = 1;
  break;
case 115:
  LED_Status = LEDFreqVals[LED_Freq]+14;
  break;
case 116:
  writeFPGAReg(0,0b1110,false);
  LED_Status = 1;
  break;
case 117:
  LED_Status = LEDFreqVals[LED_Freq]+15;
  break;
case 118:
  writeFPGAReg(0,0b1111,false);
```

```
    LED_Status = 1;
    break;
  case 119:
    LED_Status = LEDFreqVals[LED_Freq]+16;
    break;
  case 120:
    LED_Freq = 4;
    writeFPGAReg(1,LED_Freq,false);
    writeFPGAReg(2,LED_Freq,false);
    writeFPGAReg(3,LED_Freq,false);
    writeFPGAReg(4,LED_Freq,false);
    writeFPGAReg(0,0b0001,false);
    LED_Status = 1;
    break;
  case 121:
    LED_Status = LEDFreqVals[LED_Freq]+2;
    break;
  case 122:
    writeFPGAReg(0,0b0010,false);
    LED_Status = 1;
    break;
  case 123:
    LED_Status = LEDFreqVals[LED_Freq]+3;
    break;
  case 124:
    writeFPGAReg(0,0b0100,false);
    LED_Status = 1;
    break;
  case 125:
    LED_Status = LEDFreqVals[LED_Freq]+4;
    break;
  case 126:
    writeFPGAReg(0,0b1000,false);
    LED_Status = 1;
    break;
  case 127:
    LED_Status = LEDFreqVals[LED_Freq]+5;
    break;
  case 128:
    writeFPGAReg(0,0b0011,false);
    LED_Status = 1;
    break;
  case 129:
    LED_Status = LEDFreqVals[LED_Freq]+6;
    break;
  case 130:
    writeFPGAReg(0,0b0101,false);
    LED_Status = 1;
    break;
  case 131:
    LED_Status = LEDFreqVals[LED_Freq]+7;
    break;
  case 132:
    writeFPGAReg(0,0b1001,false);
    LED_Status = 1;
    break;
  case 133:
    LED_Status = LEDFreqVals[LED_Freq]+8;
    break;
  case 134:
    writeFPGAReg(0,0b0110,false);
```

```
    LED_Status = 1;
    break;
  case 135:
    LED_Status = LEDFreqVals[LED_Freq]+9;
    break;
  case 136:
    writeFPGAReg(0,0b1010,false);
    LED_Status = 1;
    break;
  case 137:
    LED_Status = LEDFreqVals[LED_Freq]+10;
    break;
  case 138:
    writeFPGAReg(0,0b1100,false);
    LED_Status = 1;
    break;
  case 139:
    LED_Status = LEDFreqVals[LED_Freq]+11;
    break;
  case 140:
    writeFPGAReg(0,0b0111,false);
    LED_Status = 1;
    break;
  case 141:
    LED_Status = LEDFreqVals[LED_Freq]+12;
    break;
  case 142:
    writeFPGAReg(0,0b1011,false);
    LED_Status = 1;
    break;
  case 143:
    LED_Status = LEDFreqVals[LED_Freq]+13;
    break;
  case 144:
    writeFPGAReg(0,0b1101,false);
    LED_Status = 1;
    break;
  case 145:
    LED_Status = LEDFreqVals[LED_Freq]+14;
    break;
  case 146:
    writeFPGAReg(0,0b1110,false);
    LED_Status = 1;
    break;
  case 147:
    LED_Status = LEDFreqVals[LED_Freq]+15;
    break;
  case 148:
    writeFPGAReg(0,0b1111,false);
    LED_Status = 1;
    break;
  case 149:
    LED_Status = LEDFreqVals[LED_Freq]+16;
    break;
  case 150:
    LED_Freq = 5;
    writeFPGAReg(1,LED_Freq,false);
    writeFPGAReg(2,LED_Freq,false);
    writeFPGAReg(3,LED_Freq,false);
    writeFPGAReg(4,LED_Freq,false);
    writeFPGAReg(0,0b0001,false);
```

```
    LED_Status = 1;
    break;
  case 151:
    LED_Status = LEDFreqVals[LED_Freq]+2;
    break;
  case 152:
    writeFPGAReg(0,0b0010,false);
    LED_Status = 1;
    break;
  case 153:
    LED_Status = LEDFreqVals[LED_Freq]+3;
    break;
  case 154:
    writeFPGAReg(0,0b0100,false);
    LED_Status = 1;
    break;
  case 155:
    LED_Status = LEDFreqVals[LED_Freq]+4;
    break;
  case 156:
    writeFPGAReg(0,0b1000,false);
    LED_Status = 1;
    break;
  case 157:
    LED_Status = LEDFreqVals[LED_Freq]+5;
    break;
  case 158:
    writeFPGAReg(0,0b0011,false);
    LED_Status = 1;
    break;
  case 159:
    LED_Status = LEDFreqVals[LED_Freq]+6;
    break;
  case 160:
    writeFPGAReg(0,0b0101,false);
    LED_Status = 1;
    break;
  case 161:
    LED_Status = LEDFreqVals[LED_Freq]+7;
    break;
  case 162:
    writeFPGAReg(0,0b1001,false);
    LED_Status = 1;
    break;
  case 163:
    LED_Status = LEDFreqVals[LED_Freq]+8;
    break;
  case 164:
    writeFPGAReg(0,0b0110,false);
    LED_Status = 1;
    break;
  case 165:
    LED_Status = LEDFreqVals[LED_Freq]+9;
    break;
  case 166:
    writeFPGAReg(0,0b1010,false);
    LED_Status = 1;
    break;
  case 167:
    LED_Status = LEDFreqVals[LED_Freq]+10;
    break;
```

```
      case 168:
        writeFPGAReg(0,0b1100,false);
        LED_Status = 1;
        break;
      case 169:
        LED_Status = LEDFreqVals[LED_Freq]+11;
        break;
      case 170:
        writeFPGAReg(0,0b0111,false);
        LED_Status = 1;
        break;
      case 171:
        LED_Status = LEDFreqVals[LED_Freq]+12;
        break;
      case 172:
        writeFPGAReg(0,0b1011,false);
        LED_Status = 1;
        break;
      case 173:
        LED_Status = LEDFreqVals[LED_Freq]+13;
        break;
      case 174:
        writeFPGAReg(0,0b1101,false);
        LED_Status = 1;
        break;
      case 175:
        LED_Status = LEDFreqVals[LED_Freq]+14;
        break;
      case 176:
        writeFPGAReg(0,0b1110,false);
        LED_Status = 1;
        break;
      case 177:
        LED_Status = LEDFreqVals[LED_Freq]+15;
        break;
      case 178:
        writeFPGAReg(0,0b1111,false);
        LED_Status = 1;
        break;
      case 179:
        LED_Status = LEDFreqVals[LED_Freq]+16;
        break;
      case 180:
        writeFPGAReg(0,0,false);
        LED_Status = 1;
        break;
      case 181:
        LED_Status = 0;
        LED_Calib = false;
        Serial.println("LED_INFO: LED Calibration Finished!");
        break;
      default:
        writeFPGAReg(0,0,false);
        LED_Status = 0;
        LED_Calib = false;
    }
    LED_FSM++;
  }
}

// Getting User Commands
```

```
  if (Serial.available()) FSM();

  // Checking LED Calibration time to start
  RTCtime = rtc.now();
  if(RTCtime.day() == 1 && RTCtime.hour() == 0 && RTCtime.minute() ==
0 && LED_Calib == false)
  {
    LED_FSM = 0;
    LED_Freq = 0;
    LED_Calib = true;
    Serial.println("LED_INFO: LED Calibration Started!");
  }

  // Changing Data File on uSD
  int diffTime=RTCtime.unixtime()-FileOpenTime;
  if ((diffTime>=3600) || (diffTime<0))
  {
    unsigned long t = RTCtime.unixtime();
    char buff[64];
    sprintf(buff, "/CRE4AT_%s_%04u%02u%02u_%02uh%02um%02us.txt",Place,
(int)year(t),(int)month(t),int(day(t)),(int)hour(t),(int)minute(t),
(int)second(t));
    FileOpenTime=t;
    FileName=String(buff);
    writeFile(SD, FileName.c_str(), FileHeader);
    Serial.print("FILE_INFO: File oppened to save data: ");
    Serial.println(FileName);
  }
  delay(1);
}
```

Apêndice H

Programa de aquisição de dados do experimento CRE4AT

Este apêndice é composto pelo programa criado em *Python3* para realizar a aquisição dos dados do experimento CRE4AT.

O código faz uso de algumas bibliotecas *python* nas seguintes versões:

1. Datetime, versão 4.9;
2. Pyserial, versão 3.5.

```
#######################################################################
# Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:           CRE4AT_IPANEMA_SiPM_SaveData.py
# Author:             Diogo Ayres Rocha (dayres@cbpf.br)
# Date:               10-07-2023
#######################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 10 Jul 23  Diogo Ayres Rocha  0001   File Created
# 15 Mar 24  Diogo Ayres Rocha  0002   PID data import implemented
# 19 Mar 24  Diogo Ayres Rocha  0003   PID data import exception added
#######################################################################

import serial
import serial.tools.list_ports
from datetime import datetime, timezone
import time
import sys
PIDImportDataFile   = "/home/cre4at/.PID_Export_DATA.dat"
now                 = datetime.now(timezone.utc)
opened_File       = datetime.timestamp(now)
timedate          = now.strftime("%Y%m%d_%Hh%Mm%Ss")
DataFile            =
"/home/cre4at/CRE4AT_SiPM/DATA/CRE4AT_IPANEMA_SiPM_%s.dat"%(timedate)
previous_PID_Data = "\t\t\t\n"
ports = list(serial.tools.list_ports.comports())
USBport = ""
for p in ports:
    if (p.vid == 4292 and p.pid == 60000):
        USBport = p.device
if (USBport == ""):
    sys.exit(-1)
print ("Found ESP32 CRE4AT USB Port: %s"%(USBport))
Daq = serial.Serial(USBport, 115200, timeout=None)
outFile           = open(DataFile,"w")
outFile.write("Timestamp\tGPS_timestamp\tRTC_timestamp\tACh0\tACh1\tAC
h2\tACh3\tBCh0\tBCh1\tBCh2\tBCh3\tCCh0\tCCh1\tCCh2\tCCh3\tSep0\tA0&1\t
A0&2\tA0&3\tA1&2\tA1&3\tA2&3\tA0&1&2\tA0&1&3\tA0&2&3\tA1&2&3\tA0&1&2&3
\tB0&1\tB0&2\tB0&3\tB1&2\tB1&3\tB2&3\tB0&1&2\tB0&1&3\tB0&2&3\tB1&2&3\t
B0&1&2&3\tC0&1\tC0&2\tC0&3\tC1&2\tC1&3\tC2&3\tC0&1&2\tC0&1&3\tC0&2&3\t
C1&2&3\tC0&1&2&3\tSep1\tA01&B01\tA01&B12\tA01&B23\tA12&B01\tA12&B12\tA
12&B23\tA23&B01\tA23&B12\tA23&B23\tA01&C01\tA01&C12\tA01&C23\tA12&C01\
tA12&C12\tA12&C23\tA23&C01\tA23&C12\tA23&C23\tB01&C01\tB01&C12\tB01&C2
3\tB12&C01\tB12&C12\tB12&C23\tB23&C01\tB23&C12\tB23&C23\tSep2\tAccX\tA
ccY\tAccZ\tGyroX\tGyroY\tGyroZ\tMagX\tMagY\tMagZ\tTemp\tHum\tPress\tHV
_Temp\tHV_Voltage\tHV_Current\tLED_CALIB_Status\tPID_Sensor1\tPID_Sens
or2\tPID_Avg\tPID_Flag\n")
outFile.close()
while(True):
      now         = datetime.now(timezone.utc)
      timestamp   = datetime.timestamp(now)
      if(timestamp - opened_File >= 3600):
            timedate    = now.strftime("%Y%m%d_%Hh%Mm%Ss")
            opened_File = timestamp
            DataFile    =
"/home/cre4at/CRE4AT_SiPM/DATA/CRE4AT_IPANEMA_SiPM_%s.dat"%(timedate)
            outFile           = open(DataFile,"w")

      outFile.write("Timestamp\tGPS_timestamp\tRTC_timestamp\tACh0\tAC
h1\tACh2\tACh3\tBCh0\tBCh1\tBCh2\tBCh3\tCCh0\tCCh1\tCCh2\tCCh3\tSep0\t
A0&1\tA0&2\tA0&3\tA1&2\tA1&3\tA2&3\tA0&1&2\tA0&1&3\tA0&2&3\tA1&2&3\tA0
```

```
&1&2&3\tB0&1\tB0&2\tB0&3\tB1&2\tB1&3\tB2&3\tB0&1&2\tB0&1&3\tB0&2&3\tB1
&2&3\tB0&1&2&3\tC0&1\tC0&2\tC0&3\tC1&2\tC1&3\tC2&3\tC0&1&2\tC0&1&3\tC0
&2&3\tC1&2&3\tC0&1&2&3\tSep1\tA01&B01\tA01&B12\tA01&B23\tA12&B01\tA12&
B12\tA12&B23\tA23&B01\tA23&B12\tA23&B23\tA01&C01\tA01&C12\tA01&C23\tA1
2&C01\tA12&C12\tA12&C23\tA23&C01\tA23&C12\tA23&C23\tB01&C01\tB01&C12\t
B01&C23\tB12&C01\tB12&C12\tB12&C23\tB23&C01\tB23&C12\tB23&C23\tSep2\tA
ccX\tAccY\tAccZ\tGyroX\tGyroY\tGyroZ\tMagX\tMagY\tMagZ\tTemp\tHum\tPre
ss\tHV_Temp\tHV_Voltage\tHV_Current\tLED_CALIB_Status\tPID_Sensor1\tPI
D_Sensor2\tPID_Avg\tPID_Flag\n")
            outFile.close()
      lineReadFlag = False
      while(not(lineReadFlag)):
            lineRead = Daq.readline().decode('utf-8')
            if ('DATA:' in lineRead):
                  now             = datetime.now(timezone.utc)
                  timestamp       = datetime.timestamp(now)
                  lineRead        = lineRead.replace("DATA: ",
"").replace("\n","")
                  lineReadFlag      = True
                  try:
                        PID_File          = open(PIDImportDataFile,"r")
                        PID_Data        = PID_File.readline()
                        PID_File.close()
                  except:
                        PID_Data = "\t\t\t\n"
                  if(PID_Data == ""):
                        PID_Data = previous_PID_Data
                  else:
                        previous_PID_Data = PID_Data
                  outFile           = open(DataFile,"a")
                  outFile.write("%d\t%s\t%s"%(timestamp, lineRead,
PID_Data))
                  outFile.close()
                  print("Saved Line: %d\t%s\t%s"%(timestamp, lineRead,
PID_Data)), ZAfRef))
```

CRE4AT_IPANEMA_SiPM_SaveData.py

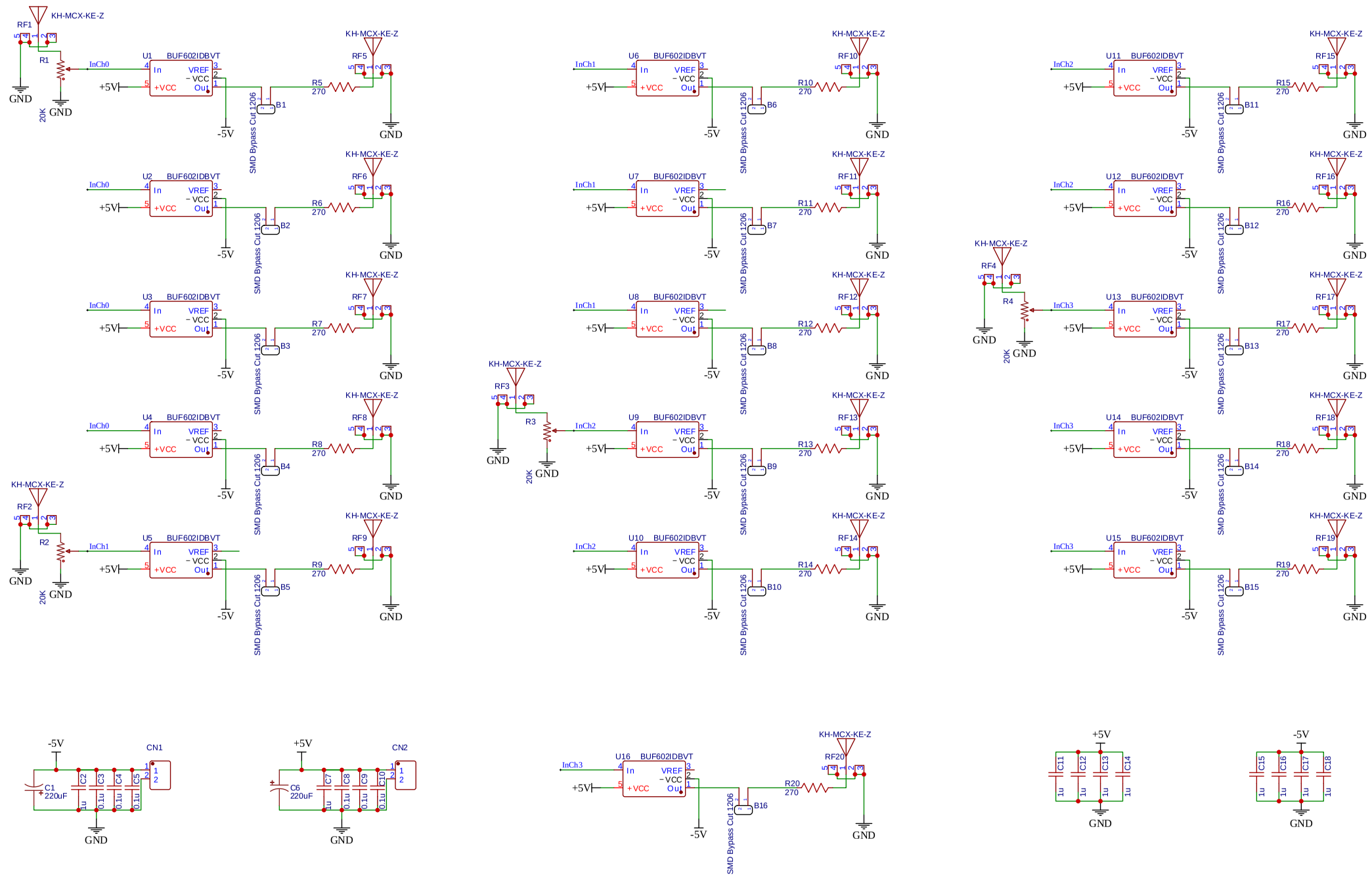
Apêndice I

Projeto da placa repetidora para injeção de luz

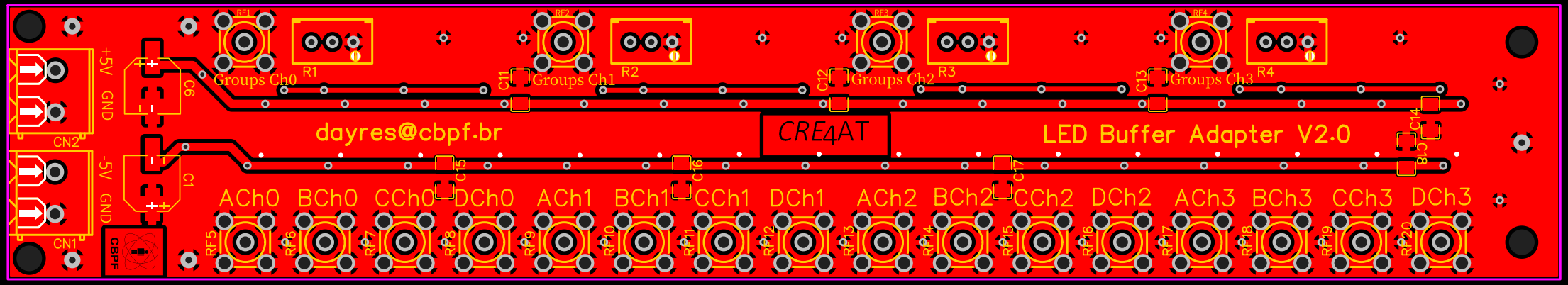
Este apêndice é composto pelo projeto da placa repetidora de sinal do sistema de injeção de luz, desenvolvido no *software EasyEDA*. O *layout* da placa foi feito em 2 camadas. O projeto, neste apêndice, será mostrado na seguinte ordem:

1. Esquemático da placa;
2. Layout camada de sinais superior;
3. Layout máscara de solda superior;
4. Layout camada de sinais inferior;
5. Layout máscara de solda inferior.

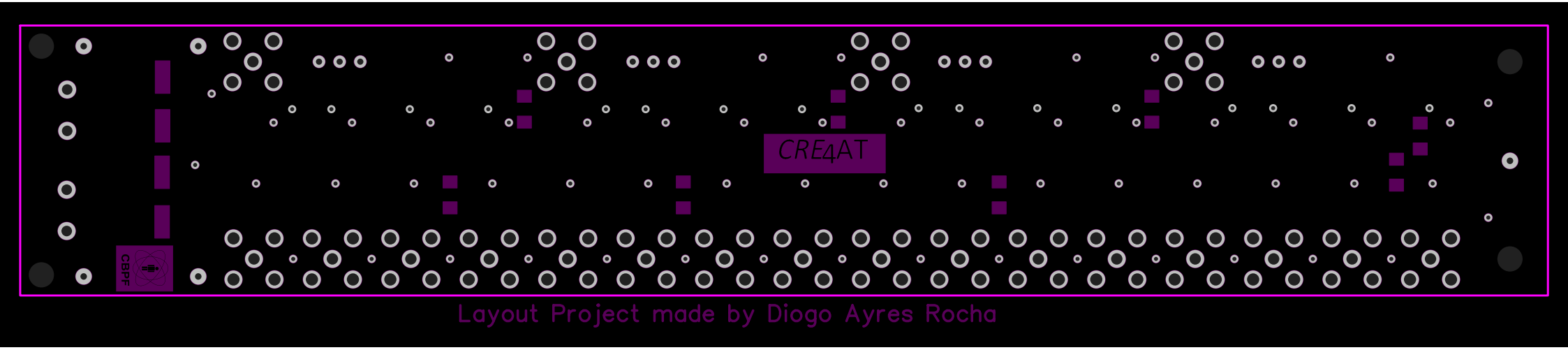
A camada inferior e a máscara de solda inferior foram espelhadas para melhor entendimento do projeto.

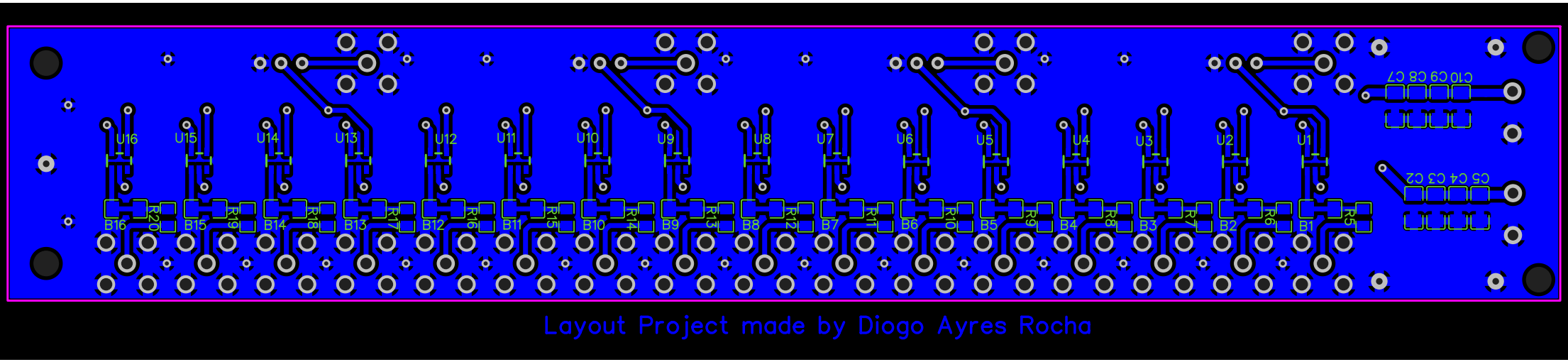


TITLE: Buffer Adapter DAQ v3	REV: 2.0
CBPF	Company: CBPF
	Sheet: 1/1
	Date: 2023-07-14 Drawn By: Diogo Ayres Rocha

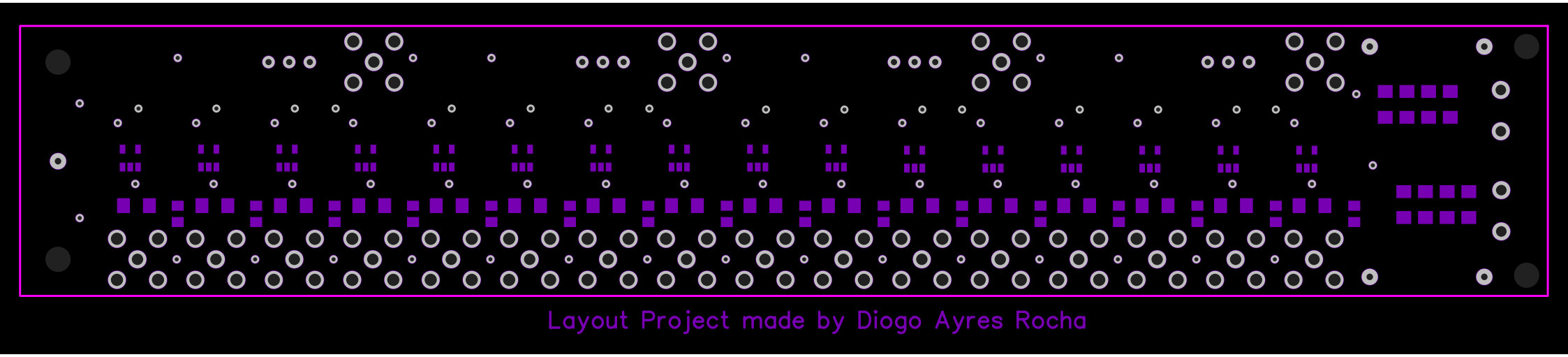


Layout Project made by Diogo Ayres Rocha


CRE4AT
CBPF
Layout Project made by Diogo Ayres Rocha



Layout Project made by Diogo Ayres Rocha

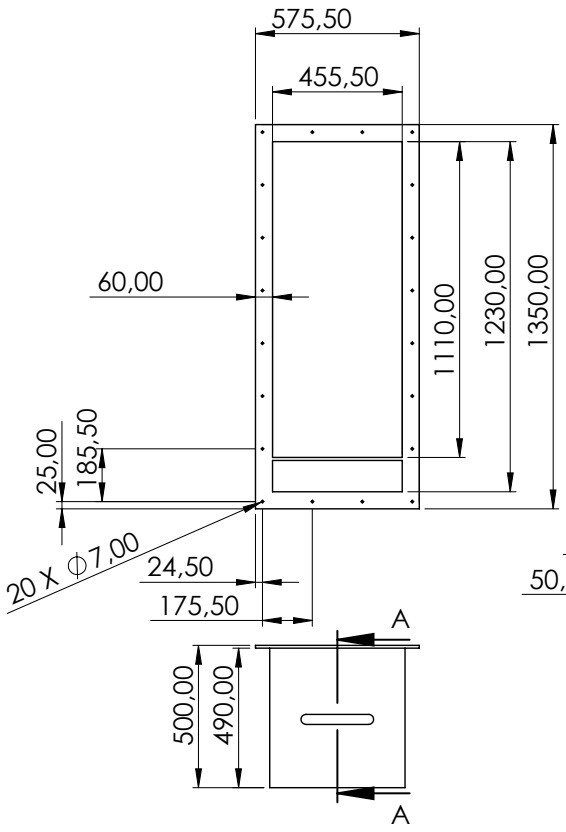

Layout Project made by Diogo Ayres Rocha

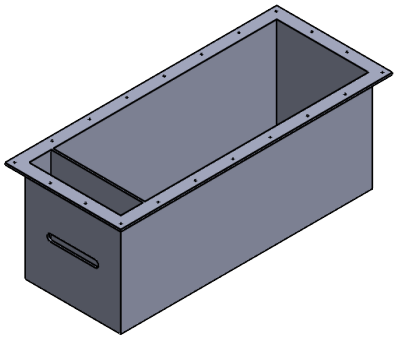
Apêndice J

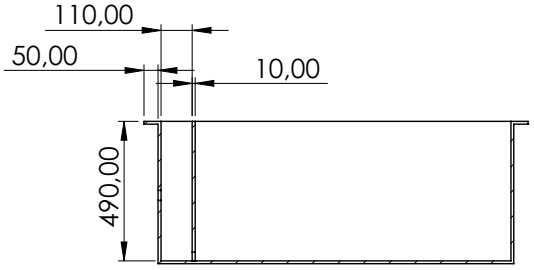
Projeto da caixa de proteção de entrada de luz

Este apêndice é composto pelo projeto mecânico da caixa de proteção de entrada de luz, também utilizado para o transporte do experimento.

montagem caixa 1411





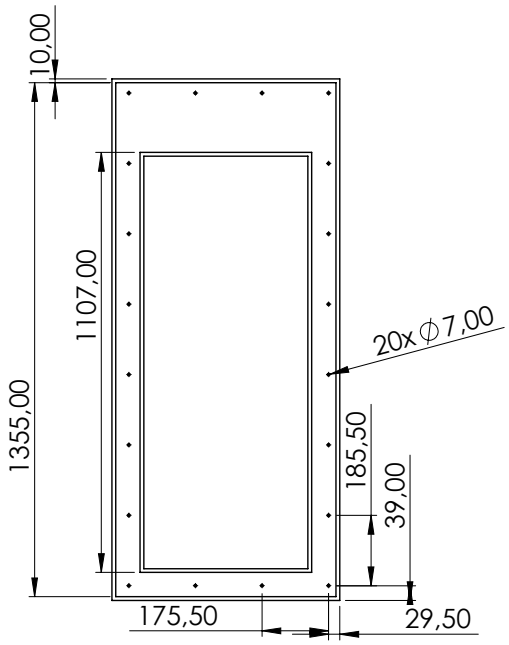


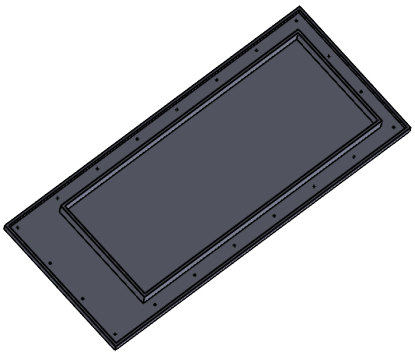
SEÇÃO A-A

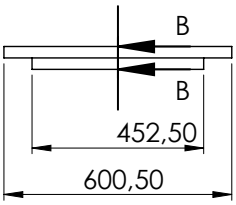
Dimensões em milímetros	3° diedro	**CAT - LITMEC** Laboratório de Instrumentação e Tecnologia Mecânica	CBPF
Material:		**Projeto:**	
Quantidade:	**Escala:** 1:20		
Peso: g	**Tolerância:** ± 0,1	Solicitante:	Dep:
Rebarbar e Quebrar arestas agudas			
Observação:		N° O.S:	**Data:** 15/12/2023
		Versão: **Folha:** 1/2	**Desenhista:**

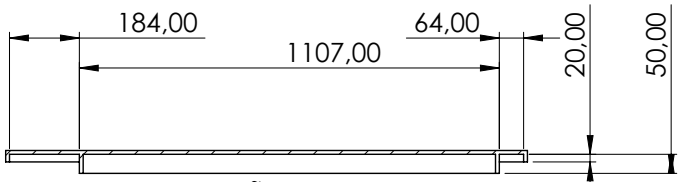
Z:\2023\NUCLEO DE PROJETOS MECANICOS -NPM\COHEP\PROF ANDRE MASSA\SUPORTE PARA CAIXA CREAT - 6 CONJUNTOS

montagem caixa 1411









SEÇÃO B-B
ESCALA 1 : 15

Dimensões em milímetros	3° diedro	**CAT - LITMEC** Laboratório de Instrumentação e Tecnologia Mecânica	CBPF
Material:		**Projeto:**	
Quantidade:	**Escala:** 1:20		
Peso: g	**Tolerância:** ± 0,1	Solicitante:	Dep:
Rebarbar e Quebrar arestas agudas			
Observação:		N° O.S:	**Data:** 15/12/2023
		Versão: **Folha:** 2/2	**Desenhista:**

Z:\2023\NUCLEO DE PROJETOS MECANICOS -NPM\COHEP\PROF ANDRE MASSA\SUPORTE PARA CAIXA CREAT - 6 CONJUNTOS

Apêndice K

Programas de leitura QDC (C++) e de análise do resultado (pyROOT)

Este apêndice é composto pelos códigos desenvolvidos para controle do módulo QDC V965 da *CAEN*, bem como o arquivo *Makefile*, para compilar o código em um sistema operacional *Linux* e um código *python3*, realizar a análise dos dados adquiridos do QDC e colocar em um gráfico o resultado, nessa ordem.

Para compilar o código C de controle, algumas bibliotecas da CAEN são necessárias. As bibliotecas usadas foram as seguintes, nas versões:

1. CAENVMELIB, versão 3.3.7;
2. CAENUSBdrvB, versão 1.5.4;
3. CAENComm, versão 1.5.0.

O código em *python3* faz uso do *framework* ROOT/CERN, na versão 6.26/04.

```
//*************************************************************************
// Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
// Filename:            macro-QDC.C
// Author:              Diogo Ayres Rocha (dayres@cbpf.br)
// Date:                23-01-2023
//*************************************************************************
//Revision History:
//   Date     Author             Rev.   Comments
// 23 Jan 23  Diogo Ayres Rocha  0001   File Created
//*************************************************************************

#include <stdlib.h>
#include <stdio.h>
#include <iomanip>
#include "CAENVMElib.h"
#include "CAENVMEtypes.h"
#include "CAENVMEoslib.h"
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <sstream>
#define Sleep(x) usleep((x)*1000)

using namespace std;

// --------------------------
// Global Variables
// --------------------------
char time2name[50];

/*************************************************/

#define MAX_BLT_SIZE          (256*1024)

#define DATATYPE_MASK         0x06000000
#define DATATYPE_HEADER       0x02000000
#define DATATYPE_CHDATA       0x00000000
#define DATATYPE_EOB          0x04000000
#define DATATYPE_FILLER       0x06000000

int exists(const char *fname)
{
    FILE *file;
    if ((file = fopen(fname, "r")))
    {
        fclose(file);
        return true;
    }
    return false;
}

/*************************************************************************
**********/
/*                                READ_REG
*/
/*************************************************************************
**********/
uint16_t read_reg(uint16_t reg_addr, long handle, uint32_t
BaseAddress, int& VMEerror)
{
      uint16_t data = 0;
```

```
	CVErrorCodes ret;
	ret = CAENVME_ReadCycle(handle, BaseAddress + reg_addr, &data,
cvA32_U_DATA, cvD16);
	if (ret != cvSuccess) {
		printf("Cannot read at address %08X\n",
(uint32_t)(BaseAddress + reg_addr));
		VMEerror = 1;
	}
	return(data);
}

/*****************************************************************
**********/
/*                                write_reg
*/
/*****************************************************************
**********/
void write_reg(uint16_t reg_addr, uint16_t data, long handle, uint32_t
BaseAddress, int& VMEerror)
{
	CVErrorCodes ret;
	ret = CAENVME_WriteCycle(handle, BaseAddress + reg_addr, &data,
cvA32_U_DATA, cvD16);
	if (ret != cvSuccess) {
		printf("Cannot write at address %08X\n",
(uint32_t)(BaseAddress + reg_addr));
		VMEerror = 1;
	}
}


int main()
{
	CVBoardTypes VMEBoard;
	short Link = 0, Device = 0;
	long BHandle;
	VMEBoard = cvV1718;
	// Base Addresses
	uint32_t BaseAddress = 0xDD220000; // IMPORTANTE SETAR!!!!!!!!!
	char ErrorString[100];
	int nchannels = 1;
	int CHs[nchannels] = {0}; // Channels to acquire
	int j, chindex, wcnt, nch, pnt, bcnt, brd_nch = 16;  //
IMPORTANTE SETAR!!!!!!!!!
	int DataError = 0;
	int DataType = DATATYPE_HEADER;
	int EnableSuppression = 1;		// Enable Zero and
Overflow suppression if QTP boards  // IMPORTANTE SETAR!!!!!!!!!
	uint16_t QTP_LLD[16] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0}; // IMPORTANTE SETAR!!!!!!!!!
	uint16_t Iped = 100;			//150 pedestal of the QDC
(or resolution of the TDC)  // IMPORTANTE SETAR!!!!!!!!!
	uint32_t buffer[MAX_BLT_SIZE / 4]; // readout buffer (raw data
from the board)
	uint16_t ADCdata[32];			// ADC data (charge, peak
or TAC)
	FILE *OutFile = NULL;    // list data file
	int VMEerror = 0;
	char dataname[80];
```

macro-QDC.C

```
      printf("\n\n-------------- MAIN PROGRAM INFO ------------------
\nYou need to create a file 'StopAcq.txt' on the program folder to
quit [it doesn't need to have anything inside...]\n");

      if (CAENVME_Init(VMEBoard, Device, Link, &BHandle) != cvSuccess)
      {
      printf(" error init !!! \n");
      return -1;
      } else {
      printf(" OK init ----\n");
      }

      Sleep(600);

      // Creating file with date/time on file name
      std::stringstream time2name_tmp;
      time_t t = time(NULL);
      tm tm = *std::localtime(&t);
      time2name_tmp << std::put_time(&tm, "%Y%m%d_%H%M%S");
      sprintf(time2name, "%s", time2name_tmp.str().c_str());
      sprintf(dataname, "DATA/%s_%s.dat", "qdcdata", time2name);

      //-------------------------------------------------------------

      printf("\n\n-------------- QDC INFO ------------------\nOpening
file: %s",dataname);
      // Open output files
      if ((OutFile = fopen(dataname, "w")) == NULL)
            printf("\nCan't open list file for writing");
    for (int i = 0; i < nchannels; i++)
    {
      if (i == nchannels-1)
            fprintf(OutFile, "Ch%d_HR\tCh%d_LR\n", CHs[i], CHs[i]);
      else
            fprintf(OutFile, "Ch%d_HR\tCh%d_LR\t", CHs[i], CHs[i]);
    }
      //
**************************************************************************
**
      // QTP settings
      //
**************************************************************************
**
      // Reset QTP board
      write_reg(0x1016, 0, BHandle, BaseAddress, VMEerror);
      if (VMEerror) {
            printf("\nError during QDC programming: ");
            printf(ErrorString);
        return -1;
      }

      write_reg(0x1060, Iped, BHandle, BaseAddress, VMEerror);  // Set
pedestal
      write_reg(0x1010, 0x60, BHandle, BaseAddress, VMEerror);  //
enable BERR to close BLT at and of block

      // Set LLD (low level threshold for ADC data)
      write_reg(0x1034, 0x100, BHandle, BaseAddress, VMEerror);  //
set threshold step = 16
      for (int i = 0; i < brd_nch; i++)
      {
```

```
            write_reg(0x1080 + i * 2, QTP_LLD[i] / 16, BHandle,
BaseAddress, VMEerror);
      }

      if (!EnableSuppression) {
            write_reg(0x1032, 0x0010, BHandle, BaseAddress, VMEerror);
// disable zero suppression
            write_reg(0x1032, 0x0008, BHandle, BaseAddress, VMEerror);
// disable overrange suppression
            write_reg(0x1032, 0x1000, BHandle, BaseAddress, VMEerror);
// enable empty events
      }

      //printf("Ctrl Reg = %04X\n", read_reg(0x1032));
      printf("\nQDC board programmed\n");
      pnt = 0;  // word pointer
      wcnt = 0; // num of lword read in the MBLT cycle
      buffer[0] = DATATYPE_FILLER;

      // clear Event Counter
      write_reg(0x1040, 0x0, BHandle, BaseAddress, VMEerror);
      // clear QDC
      write_reg(0x1032, 0x4, BHandle, BaseAddress, VMEerror);
      write_reg(0x1034, 0x4, BHandle, BaseAddress, VMEerror);

      do
      {
            // if needed, read a new block of data from the board
            if ((pnt == wcnt) || ((buffer[pnt] & DATATYPE_MASK) ==
DATATYPE_FILLER))
            {
                  CAENVME_MBLTReadCycle(BHandle, BaseAddress, (unsigned
char *)buffer, MAX_BLT_SIZE, cvA32_U_MBLT, &bcnt);
                  wcnt = bcnt / 4;
                  pnt = 0;
            }
            if (wcnt == 0)  // no data available
                  continue;
            /* header */
            switch (DataType)
            {
            case DATATYPE_HEADER:
                  if ((buffer[pnt] & DATATYPE_MASK) != DATATYPE_HEADER)
{
                        //printf("Header not found: %08X (pnt=%d)\n",
buffer[pnt], pnt);
                        DataError = 1;
                  }
                  else
                  {
                        nch = (buffer[pnt] >> 8) & 0x3F;
                        chindex = 0;
                        memset(ADCdata, 0xFFFF, 32 * sizeof(uint16_t));
                        if (nch > 0)
                              DataType = DATATYPE_CHDATA;
                        else
                              DataType = DATATYPE_EOB;
                  }
                  break;

                  /* Channel data */
```

```
            case DATATYPE_CHDATA:
                  if ((buffer[pnt] & DATATYPE_MASK) != DATATYPE_CHDATA)
{
                        //printf("Wrong Channel Data: %08X (pnt=%d)\n",
buffer[pnt], pnt);
                        DataError = 1;
                  }
                  else
                  {
                        j = (int)((buffer[pnt] >> 16) & 0x3F);
                        ADCdata[j] = buffer[pnt] & 0xFFF;
                        if (chindex == (nch - 1))
                              DataType = DATATYPE_EOB;
                        chindex++;
                  }
                  break;

                  /* EOB */
            case DATATYPE_EOB:
                  if ((buffer[pnt] & DATATYPE_MASK) != DATATYPE_EOB) {
                        //printf("EOB not found: %08X (pnt=%d)\n",
buffer[pnt], pnt);
                        DataError = 1;
                  }
                  else
                  {
                        DataType = DATATYPE_HEADER;
                        for (int i = 0; i < nchannels*2; i++)
                        {
                              //if (ADCdata[CHs[i]] != 0xFFFF)
                              //{
                              int intCh   = i / 2;
                              int restCh  = i % 2;
                              if (i==nchannels*2-1)
                              {
                                    fprintf(OutFile, "%d\n",
ADCdata[CHs[intCh]+restCh]);
                              }
                              else fprintf(OutFile, "%d\t",
ADCdata[CHs[intCh]+restCh]);
                              //}
                        }
                  }
                  break;
            }
            pnt++;

            if (DataError)
            {
                  pnt = wcnt;
                  write_reg(0x1032, 0x4, BHandle, BaseAddress,
VMEerror);
                  write_reg(0x1034, 0x4, BHandle, BaseAddress,
VMEerror);
                  DataType = DATATYPE_HEADER;
                  DataError = 0;
            }
      } while (!exists("./StopAcq.txt"));

      fclose(OutFile);
    printf("QDC closed!");
```

macro-QDC.C

```
    CAENVME_End(BHandle);
}
```

```
#########################################################################
# Company:              CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:             Makefile
# Author:               Diogo Ayres Rocha (dayres@cbpf.br)
# Date:                 23-01-2023
#########################################################################
# Revision History:
#   Date     Author             Rev.    Comments
# 23 Jan 23  Diogo Ayres Rocha  0001    File Created
#########################################################################

### to compile perform:
# make
# optional: make clean

EXE    =      QDC.a

CXX    =      g++

CXXFLAGS     = -std=c++17 -Wall -s

COPTS =      -fPIC -DLINUX

DEPLIBS      =        -lCAENVME -lm   -pthread

LIBS   =

INCLUDEDIR =       -I.

OBJS   =      macro-QDC.o

INCLUDES =

#########################################################################
###

all    :      $(EXE)

clean :
             /bin/rm -f $(OBJS) $(EXE)

$(EXE)       :      $(OBJS)
             /bin/rm -f $(EXE)
             $(CXX) $(CXXFLAGS) -o $(EXE)   $(OBJS) $(DEPLIBS)


$(OBJS)      :      $(INCLUDES) Makefile

%.o    :      %.c
              $(CXX) $(COPTS) $(INCLUDEDIR) -c -o $@   $<
```

Makefile

```
#######################################################################
# Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:           macro-PlotQDC.py
# Author:             Diogo Ayres Rocha (dayres@cbpf.br)
# Date:               23-01-2023
#######################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 23 Jan 23  Diogo Ayres Rocha  0001   File Created
#######################################################################

import ROOT
import pandas as pd
import numpy as np
import sys
from ROOT import TCanvas, TH1F, TF1
from array import array

ChRange = "Ch0_HR"
data  = pd.read_csv(sys.argv[1], sep='\t')
if ("LR" in ChRange):
      Ch0         = (data[ChRange].values) * (0.025)
else:
      Ch0         = (data[ChRange].values) * (0.2)
h1=TH1F("Ch0 Charge", "Ch0 Charge", 500, 0, 600)
for value in Ch0:
      h1.Fill(value)
h1.GetYaxis().SetTitle("Bins")
h1.GetXaxis().SetTitle("Charge [pC]")
c1=TCanvas("Ch0 Charge Histogram","Ch0 Charge Histogram", 1280, 800)
h1.Draw()
c1.Print("Charge_Histogram_CRE4AT.png")
```

Apêndice L

Programas desenvolvidos para executar a varredura da tensão de limiar e latch e de análise do resultado (pyROOT)

Este apêndice é composto pelo código desenvolvido em *python3*, tendo por objetivo o controle do sistema de aquisição do experimento CRE4AT e da fonte de alimentação controlada *Keithley* para execução automatizada das varreduras da tensão de limiar e do *latch*; e um programa em pyROOT que realiza a análise dos dados obtidos, gerando gráficos do resultado.

Os códigos fazem uso de algumas bibliotecas *python* nas seguintes versões:

1. Numpy, versão 1.19.5;
2. Pandas, versão 1.1.5;
3. Datetime, versão 4.9;
4. Python-usbtmc, versão 0.8;
5. Pyserial, versão 3.5.

O código de análise do resultado faz uso do *framework* ROOT/CERN, na versão 6.26/04.

```
#######################################################################
# Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:           CRE4AT_ThScan_With&Without_LEDInj.py
# Author:             Diogo Ayres Rocha (dayres@cbpf.br)
# Date:               18-10-2023
#######################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 18 OCT 23  Diogo Ayres Rocha  0001   File Created
# 10 NOV 23  Diogo Ayres Rocha  0002   Protocol changes
#######################################################################

import serial
from datetime import datetime
import time
import sys
import numpy as np
from ROOT import TCanvas, TGraph
import usbtmc


# User Parameters
nGroups                 = 1
LatchREG          = 12
AcqTimeMSB        = 13
AcqTimeLSB        = 14
LEDReg                  = 0
DetArea                 = 0.1 * 0.5
# ---------------------------------------------------------------------
--------

def SetVolt(voltage):
      ps.write("SYSTEM:REMOTE\n")
      ps.write("VOLT %f\n"%(voltage))
      time.sleep(5)
      for secs in np.arange(0,3,0.1):
            time.sleep(0.1)
            VoltRead=float(ps.ask("MEASure:VOLTage?"))
            print("INFO: Measure Th Voltage = %f"%(VoltRead))
            if (abs(VoltRead-voltage) < 0.01):
                  print("%4fV successfully applied!"%(voltage))
                  return VoltRead
                  break
      return VoltRead

def ClearCountersData(Daq, Time):
    RegLSB = Time & 0xFF
    RegMSB = (Time >> 8)
    str="REG;%d;0;\n"%(AcqTimeMSB)
    Daq.write(str.encode('utf-8'))
    ret = ""
    time.sleep(1)
    while (Daq.inWaiting()):
        ret+= Daq.readline().decode('utf-8')
    #print('Returned string: %s'%(ret))
    TimeOk=False
    if ('successfully' in ret):
        #print('Time successfully applied!\n\n')
        TimeOk=True
    str="REG;%d;1;\n"%(AcqTimeLSB)
    Daq.write(str.encode('utf-8'))
```

```
    ret = ""
    time.sleep(1)
    while (Daq.inWaiting()):
        ret+= Daq.readline().decode('utf-8')
    time.sleep(1)
    str="REG;%d;%d;\n"%(AcqTimeMSB,RegMSB)
    Daq.write(str.encode('utf-8'))
    ret = ""
    time.sleep(1)
    while (Daq.inWaiting()):
        ret+= Daq.readline().decode('utf-8')
    #print('Returned string: %s'%(ret))
    TimeOk=False
    if ('successfully' in ret):
        #print('Time successfully applied!\n\n')
        TimeOk=True
    str="REG;%d;%d;\n"%(AcqTimeLSB,RegLSB)
    Daq.write(str.encode('utf-8'))
    ret = ""
    time.sleep(1)
    while (Daq.inWaiting()):
        ret+= Daq.readline().decode('utf-8')
    if (('successfully' in ret) and TimeOk):
        print("INFO: Counters Cleaned!")

now         = datetime.now()
timedate    = now.strftime("%d%m%Y_%Hh%Mm%Ss")
Filename    = "CRE4AT_ThScan_%s.dat"%(timedate)
scanOutFile = open(Filename,"w")
scanOutFile.write("Th\tThMeas\tLatch\tLEDStatus\tEff0\tEff1\tEff2\tEff
3\tFlux\tFluxSingle\tGPS_timestamp\tRTC_timestamp\tACh0\tACh1\tACh2\tA
Ch3\tBCh0\tBCh1\tBCh2\tBCh3\tCCh0\tCCh1\tCCh2\tCCh3\tSep0\tA0&1\tA0&2\
tA0&3\tA1&2\tA1&3\tA2&3\tA0&1&2\tA0&1&3\tA0&2&3\tA1&2&3\tA0&1&2&3\tB0&
1\tB0&2\tB0&3\tB1&2\tB1&3\tB2&3\tB0&1&2\tB0&1&3\tB0&2&3\tB1&2&3\tB0&1&
2&3\tC0&1\tC0&2\tC0&3\tC1&2\tC1&3\tC2&3\tC0&1&2\tC0&1&3\tC0&2&3\tC1&2&
3\tC0&1&2&3\tSep1\tA01&B01\tA01&B12\tA01&B23\tA12&B01\tA12&B12\tA12&B2
3\tA23&B01\tA23&B12\tA23&B23\tA01&C01\tA01&C12\tA01&C23\tA12&C01\tA12&
C12\tA12&C23\tA23&C01\tA23&C12\tA23&C23\tB01&C01\tB01&C12\tB01&C23\tB1
2&C01\tB12&C12\tB12&C23\tB23&C01\tB23&C12\tB23&C23\tSep2\tAccX\tAccY\t
AccZ\tGyroX\tGyroY\tGyroZ\tMagX\tMagY\tMagZ\tTemp\tHum\tPress\tHV_Temp
\tHV_Voltage\tHV_Current\tLED_CALIB_Status\n")
scanOutFile.close()
ps=usbtmc.Instrument(int('05e6', 16), int('2200', 16))
try:
    ps.clear()
except:
    print("Power Supply not connected!")
    sys.exit(1)
ps.write("OUTPUT 1\n")
Daq = serial.Serial('/dev/ttyUSB1', 115200, timeout=None)
time.sleep(5)
while (Daq.inWaiting()):
    tmp=Daq.readline()

# Getting latches to test on the scan
Minlatch = input("Set the Min Latch fot the test[default=5(100ns) |
!Max=255!]: ")
if (Minlatch==""):
    Minlatch=5
else:
    Minlatch=int(Minlatch)
```

CRE4AT_ThScan_With&Without_LEDInj.py

```
print('INFO: Setting Latch to %d...'%(Minlatch))
str="REG;%d;%d;\n"%(LatchREG,Minlatch)
Daq.write(str.encode('utf-8'))
ret = ""
time.sleep(1)
while (Daq.inWaiting()):
    ret+= Daq.readline().decode('utf-8')
#print('Returned string: %s'%(ret))
if ('successfully' in ret):
    print('INFO: Latch %d successfully applied!\n\n'%(Minlatch))

Maxlatch = input("Set the Max Latch fot the test[default=30(600ns) |
!Max=255!]: ")
if (Maxlatch==""):
    Maxlatch=30
else:
    Maxlatch=int(Maxlatch)

Stepthlatch = input("Set the step of Latches to test[default=5]: ")
if (Stepthlatch==""):
    Stepthlatch=5
else:
    Stepthlatch=int(Stepthlatch)

Latches     = np.arange(Minlatch, Maxlatch+Stepthlatch, Stepthlatch)
print("\n\nINFO: Latches to test: ")
print(Latches)
print("\n\n")
NLatches    = len(Latches)
# Getting Thresholds to test on the scan
Minth = input("Set FEE Min Threshold[default=0.01V | !Min=0V!]: ")
if (Minth==""):
    Minth=0.01
else:
    Minth=float(Minth)
Maxth = input("Set FEE Max Threshold[default=0.15V | !Max=5V!]: ")
if (Maxth==""):
    Maxth=0.15
else:
    Maxth=float(Maxth)
Stepth = input("Set the step of Thresholds to Scan [default=0.01]: ")
if (Stepth==""):
    Stepth=0.01
else:
    Stepth=float(Stepth)
Thresholds = np.arange(Minth, Maxth+Stepth, Stepth)
print("\n\nINFO: Thesholds to test: ")
print(Thresholds)
print("\n\n")
NThs = len(Thresholds)

# Getting the Acquisiton time to execute the scan
Time2Acq = input("Set time to Acquire data in seconds[default=100 |
!Max=65535!]: ")
if (Time2Acq == ""):
    Time2Acq = 100
else:
    Time2Acq=int(Time2Acq)
RegLSB = Time2Acq & 0xFF
RegMSB = (Time2Acq >> 8)
print('INFO: Setting Acquisition Time to %ds...'%(Time2Acq))
```

CRE4AT_ThScan_With&Without_LEDInj.py

```
str="REG;%d;%d;\n"%(AcqTimeMSB,RegMSB)
Daq.write(str.encode('utf-8'))
ret = ""
time.sleep(1)
while (Daq.inWaiting()):
    ret+= Daq.readline().decode('utf-8')
#print('Returned string: %s'%(ret))
TimeOk=False
if ('successfully' in ret):
    #print('Time successfully applied!\n\n')
    TimeOk=True
str="REG;%d;%d;\n"%(AcqTimeLSB,RegLSB)
Daq.write(str.encode('utf-8'))
ret = ""
time.sleep(1)
while (Daq.inWaiting()):
    ret+= Daq.readline().decode('utf-8')
#print('Returned string: %s'%(ret))
if (('successfully' in ret) and TimeOk):
      print("INFO: Time successfully applied!\n\n")
eff0              = np.zeros((nGroups,2,NLatches,NThs))
eff1              = np.zeros((nGroups,2,NLatches,NThs))
eff2              = np.zeros((nGroups,2,NLatches,NThs))
eff3              = np.zeros((nGroups,2,NLatches,NThs))
flux              = np.zeros((nGroups,2,NLatches,NThs))
fluxSingle  = np.zeros((nGroups,2,NLatches,NThs))
for thPos, th in enumerate(Thresholds):
      #Stop2Set = input("Ajust the Threshold to %.4fV and click enter
to continue!"%(th))
      print("Setting threshold to %.4f"%(th))
      ThVoltageRead = SetVolt(th)
      for latchPos in range(0, NLatches):
            print('INFO: Setting Latch to %d...'%(Latches[latchPos]))
            str="REG;%d;%d;\n"%(LatchREG,Minlatch)
            Daq.write(str.encode('utf-8'))
            ret = ""
            time.sleep(1)
            while (Daq.inWaiting()):
                  ret+= Daq.readline().decode('utf-8')
            #print('Returned string: %s'%(ret))
            if ('successfully' in ret):
                  print('INFO: Latch %d successfully
applied!\n\n'%(Latches[latchPos]))
            for LEDStatus in range(0,2):
                  if (LEDStatus == 1):
                        print('INFO: Setting LEDs to ON ...')
                        str="REG;%d;%d;\n"%(LEDReg, 255)
                        Daq.write(str.encode('utf-8'))
                        ret = ""
                        time.sleep(1)
                        while (Daq.inWaiting()):
                              ret+= Daq.readline().decode('utf-8')
                        #print('Returned string: %s'%(ret))
                        if ('successfully' in ret):
                              print('INFO: LED ON!\n\n')
                  ClearCountersData(Daq, Time2Acq)
                  while(True):
                        lineRead = Daq.readline().decode('utf-8')
                        if ('DATA:' in lineRead):
                              lineRead = lineRead.replace("DATA: ", "")
                              print("Data Acquired: %s"%(lineRead))
```

CRE4AT_ThScan_With&Without_LEDInj.py

```
                                    Values=lineRead.split("\t")
                                    #print(Values)
                                    str=""
                                    for group in range(0, nGroups):
                                          PosCh2
      = 4
                                          Pos0and1and2            =
11*group+21
                                          Pos0and1and3            =
11*group+22
                                          Pos0and2and3            =
11*group+23
                                          Pos1and2and3            =
11*group+24
                                          Pos0and1and2and3  = 11*group+25
                                          try:

      eff0[group][LEDStatus][latchPos][thPos]         =
float(Values[Pos0and1and2and3])/float(Values[Pos1and2and3])
                                          except:

      eff0[group][LEDStatus][latchPos][thPos]         = -1
                                          try:

      eff1[group][LEDStatus][latchPos][thPos]         =
float(Values[Pos0and1and2and3])/float(Values[Pos0and2and3])
                                          except:

      eff1[group][LEDStatus][latchPos][thPos]         = -1
                                          try:

      eff2[group][LEDStatus][latchPos][thPos]         =
float(Values[Pos0and1and2and3])/float(Values[Pos0and1and3])
                                          except:

      eff2[group][LEDStatus][latchPos][thPos]         = -1
                                          try:

      eff3[group][LEDStatus][latchPos][thPos]         =
float(Values[Pos0and1and2and3])/float(Values[Pos0and1and2])
                                          except:

      eff3[group][LEDStatus][latchPos][thPos]         = -1
                                          if
((eff0[group][LEDStatus][latchPos][thPos] == -1) or
(eff1[group][LEDStatus][latchPos][thPos] == - 1) or
(eff2[group][LEDStatus][latchPos]   [thPos] == -1) or
(eff3[group][LEDStatus][latchPos][thPos] == -1)):

      flux[group][LEDStatus][latchPos][thPos]         = -1
                                          else:
                                                try:

      flux[group][LEDStatus][latchPos][thPos]   =
float(Values[Pos0and1and2and3])/(DetArea*Time2Acq*eff0[group][LEDStatu
s][latchPos][thPos]*eff1[group][LEDStatus][latchPos][thPos]*eff2[group
][LEDStatus][latchPos][thPos]*eff3[group][LEDStatus][latchPos][thPos])
                                                except:

      flux[group][LEDStatus][latchPos][thPos]   = -1
```

```
      if(eff2[group][LEDStatus][latchPos][thPos] == -1):

      fluxSingle[group][LEDStatus][latchPos][thPos]   = -1
                                    else:
                                          try:

      fluxSingle[group][LEDStatus][latchPos][thPos]   =
float(PosCh2)/(DetArea*Time2Acq*eff2[group][LEDStatus][latchPos][thPos
])
                                          except:

      fluxSingle[group][LEDStatus][latchPos][thPos]   = -1
                                    str+="Th %.4fV - Latch %dns -
Group%c Ch0 Eff:\t%.3f\n"%(th, Latches[latchPos]*20, chr(group+65),
eff0[group][LEDStatus][latchPos][thPos])
                                    str+="Th %.4fV - Latch %dns -
Group%c Ch1 Eff:\t%.3f\n"%(th, Latches[latchPos]*20, chr(group+65),
eff1[group][LEDStatus][latchPos][thPos])
                                    str+="Th %.4fV - Latch %dns -
Group%c Ch2 Eff:\t%.3f\n"%(th, Latches[latchPos]*20, chr(group+65),
eff2[group][LEDStatus][latchPos][thPos])
                                    str+="Th %.4fV - Latch %dns -
Group%c Ch3 Eff:\t%.3f\n"%(th, Latches[latchPos]*20, chr(group+65),
eff3[group][LEDStatus][latchPos][thPos])
                                    str+="Th %.4fV - Latch %dns -
Group%c Flux:\t%.3f\n"%(th, Latches[latchPos]*20, chr(group+65),
flux[group][LEDStatus][latchPos][thPos])
                                    str+="Th %.4fV - Latch %dns -
Group%c Flux Single[Using Ch1]:\t%.3f\n"%(th, Latches[latchPos]*20,
chr(group+65), fluxSingle[group][LEDStatus][latchPos][thPos])
                                    scanOutFile = open(Filename,"a")

      scanOutFile.write("%.4f\t%.4f\t%d\t%d\t%.4f\t%.4f\t%.4f\t%.4f\t%
.4f\t%.4f\t%s"%(th, ThVoltageRead, Latches[latchPos], LEDStatus,
eff0[group][LEDStatus][latchPos][thPos],
eff1[group][LEDStatus][latchPos][thPos],
eff2[group][LEDStatus][latchPos][thPos],
eff3[group][LEDStatus][latchPos][thPos],
flux[group][LEDStatus][latchPos][thPos],
fluxSingle[group][LEDStatus][latchPos][thPos], lineRead))
                                    scanOutFile.close()
                              print(str)
                              if (LEDStatus == 1):
                                    print('INFO: Setting LEDs to OFF
...')
                                    str="REG;%d;%d;\n"%(LEDReg, 0)
                                    Daq.write(str.encode('utf-8'))
                                    ret = ""
                                    time.sleep(1)
                                    while (Daq.inWaiting()):
                                          ret+=
Daq.readline().decode('utf-8')
                                    #print('Returned string: %s'%(ret))
                                    if ('successfully' in ret):
                                          print('INFO: LED OFF!\n\n')
                              break
ps.write("OUTPUT 0\n")
Daq.close()
```

CRE4AT_ThScan_With&Without_LEDInj.py

```
##########################################################################
# Company:              CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:             CRE4AT_PlotThScan.py
# Author:               Diogo Ayres Rocha (dayres@cbpf.br)
# Date:                 01-01-2024
##########################################################################
# Revision History:
#   Date     Author             Rev.    Comments
# 01 JAN 24  Diogo Ayres Rocha  0001    File Created
##########################################################################

import ROOT
from ROOT import TCanvas, TGraph2DErrors
import pandas as pd
import numpy as np
import sys

AcqTime         = 120
ReadVoltError   = 0.0002

file            = sys.argv[1]
Channel2Plot    = sys.argv[2]
LedStatus       = int(sys.argv[3])

df      = pd.read_csv(file,sep='\t')
dftemp  = df[(df.LEDStatus==LedStatus)]

c       = TCanvas("CRE4AT ThScan","CRE4AT ThScan", 1280, 1200)
g=TGraph2DErrors(len(dftemp), np.asarray(dftemp.ThMeas).astype(float),
np.asarray(dftemp.Latch).astype(float)*20,
np.asarray(dftemp[Channel2Plot]).astype(float)/AcqTime,
np.asarray(dftemp.ThMeas).astype(float) * ReadVoltError,
np.zeros(len(dftemp)),
(np.asarray(dftemp[Channel2Plot]).astype(float)/AcqTime)**0.5)
g.Draw("surf1")
g.SetTitle("CRE4AT Threshold Scan - %s"%("No Light Injection"
if(LedStatus == 0) else "Light Injection - 10kHz"))
g.GetYaxis().SetTitle("Latch [ns]")
g.GetZaxis().SetTitle("Frequency [Hz]")
g.GetXaxis().SetTitle("Th [Volts]")
g.GetXaxis().SetTitleOffset(2)
g.GetYaxis().SetTitleOffset(2)
g.GetZaxis().SetTitleOffset(1.5)
g.Draw("surf1")
c.Print("CRE4AT_ThScan_%s_%s"%(Channel2Plot, "No Light Injection"
if(LedStatus == 0) else "Light Injection - 10kHz"))
```

Apêndice M

Programas de simulação Monte Carlo para geração do resultado para cada ZAF e de junção dos resultados em uma unica tabela

Este apêndice é composto pelo código desenvolvido em *python3*, com uso do *framework* ROOT/CERN na versão 6.26/04, que realiza a simulação Monte Carlo para cada fator do ângulo zenital no modelo de produção teórico dos raios cósmicos e um código para união de do resultado para cada fator em uma única tabela, nesta ordem.

Os códigos fazem uso de algumas bibliotecas *python* nas seguintes versões:

1. Numpy, versão 1.19.5;
2. Pandas, versão 1.1.5;
3. Datetime, versão 4.9.

```
#########################################################################
# Company:  CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename: macro-CRE4AT_groupsABC_IPANEMA_SingleZAFactor_Histograms.py
# Author:   Diogo Ayres Rocha (dayres@cbpf.br)
# Date:     10-08-2023
#########################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 10 Ago 23  Diogo Ayres Rocha  0001   File Created
#########################################################################

import math
import sys
import datetime
from ROOT import gRandom, TH1F, TH2F, TCanvas, gStyle, gPad, TFile

from math import *
import numpy as np

### main goal is to calculate the geometrical correction
### for Efficiency and flux measurement
### in 3 groups and 4 planes conditions for each group
### thickness of the detectors are included

n_value_model_prod = float(sys.argv[1])

timestamp   = datetime.datetime.now().strftime("%d%m%Y_%Hh%Mm%Ss")
result_file       =
"CRE4AT_IPANEMA_MAP_1Bevents_ZAFactor%.2f.dat"%(n_value_model_prod)
nEvent                  = int(3600*150*100*24) # 1h in seconds * 150
particles/(second * m²) * generation plane area  [ ~ (Enlargment*2) **
2 ] * 24h
Enlargment  = np.float64(500.0)

gRandom.SetSeed(0)

# the setup geommetry for each detector

# two separations to compare
z           = [15.5,16.5,17.5,18.5,2.7,3.7,4.7,5.7,29.3,30.3,31.3,32.3]

x           = [0.0,0.0,0.0,0.0,13.7,13.7,13.7,13.7,27.4,27.4,27.4,27.4]

y           = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]

ncellsx = [1] * 12

ncellsy = [1] * 12

deltax  = [10.0] * 12

deltay  = [69.9] * 12

width   = [1.0] * 12

NumDetectors = len(z)

area = []

# production plane
```

```
xminprod = -Enlargment
yminprod = -Enlargment
xmaxprod = (x[-1]-x[0])+Enlargment # 3 is the number of groups
ymaxprod = (ncellsy[0]*deltay[0])+Enlargment # 3 is the number of
groups

areaprod = (xmaxprod-xminprod)*(ymaxprod-yminprod)

# definitions,calculations and initializations
pi = math.pi
MaxProd = 0
for i in range(0,100000):
      phi_gen = pi*(2.0*(gRandom.Rndm())) #0 to 2*pi
      theta_gen = (pi/2.0)*gRandom.Rndm() #0 to pi/2
      kynematic = math.sin(theta_gen)
      prod      =
math.pow(cos(theta_gen),n_value_model_prod)*kynematic
      if (MaxProd < prod):
            MaxProd = prod
print("INFO: Max production value found: %.4f"%(MaxProd))

def GenerateMuons(value):

    aux  = 1
    prod = 0
    x_gen    = (xmaxprod-xminprod)*gRandom.Rndm()+xminprod
    y_gen    = (ymaxprod-yminprod)*gRandom.Rndm()+yminprod

    phi_gen = pi*(2.0*(gRandom.Rndm())) #0 to 2*pi
    while ( aux > prod ) :
        aux       = (MaxProd*1.1)*gRandom.Rndm() # Appling the max
value found with 10% increase
        theta_gen = (pi/2.0)*gRandom.Rndm() #0 to pi/2
        kynematic = math.sin(theta_gen)
        prod      = math.pow(cos(theta_gen),value)*kynematic

    return x_gen,y_gen,phi_gen,theta_gen

def MuonInDetector(idet, gen, xi, yi, zi, deltaxi, deltayi):

    radii  = np.float64(tan(gen[3])*zi[idet])
    x1     = np.float64(radii*cos(gen[2]) + gen[0])
    y1     = np.float64(radii*sin(gen[2]) + gen[1])

    radii  = np.float64(tan(gen[3])*(zi[idet]+width[idet]))
    x2     = np.float64(radii*cos(gen[2]) + gen[0])
    y2     = np.float64(radii*sin(gen[2]) + gen[1])

    icell=0
    while True :
        if ( ( (x1>=(xi[idet]+icell*deltaxi[idet]) ) and (
x1<(xi[idet]+icell*deltaxi[idet]+deltaxi[idet]) ) ) or
             ( (x2>=(xi[idet]+icell*deltaxi[idet]) ) and (
x2<(xi[idet]+icell*deltaxi[idet]+deltaxi[idet]) ) ) or
             (icell==ncellsx[idet]) )  :
            break
        icell+=1

    if (icell==ncellsx[idet]) :
        chx=-1
    else :
```

```
        chx=icell

    icell=0
    while True :
        if ( ( (y1>=(yi[idet]+icell*deltayi[idet])) and (
y1<(yi[idet]+icell*deltayi[idet]+deltayi[idet]) ) or
             ( (y2>=(yi[idet]+icell*deltayi[idet])) and (
y2<(yi[idet]+icell*deltayi[idet]+deltayi[idet]) ) or
            (icell==ncellsy[idet]) ):
            break
        icell+=1

    if (icell==ncellsy[idet]) :
        chy=-1
    else :
        chy=icell

    return chx,chy

############### the main

print("INFO--> Starting .... ")
print(" ")

print("INFO--> n_value_model_prod = %.2f"%(n_value_model_prod))
iev                = 0
CoinA012           = 0
CoinA013           = 0
CoinA023           = 0
CoinA123           = 0
CoinB012           = 0
CoinB013           = 0
CoinB023           = 0
CoinB123           = 0
CoinC012           = 0
CoinC013           = 0
CoinC023           = 0
CoinC123           = 0
Coin4A             = 0
Coin4B             = 0
Coin4C             = 0
Coin4_A01B01       = 0
Coin4_A01B12       = 0
Coin4_A01B23       = 0
Coin4_A12B01       = 0
Coin4_A12B12       = 0
Coin4_A12B23       = 0
Coin4_A23B01       = 0
Coin4_A23B12       = 0
Coin4_A23B23       = 0
Coin4_A01C01       = 0
Coin4_A01C12       = 0
Coin4_A01C23       = 0
Coin4_A12C01       = 0
Coin4_A12C12       = 0
Coin4_A12C23       = 0
Coin4_A23C01       = 0
Coin4_A23C12       = 0
Coin4_A23C23       = 0
Coin4_B01C01       = 0
Coin4_B01C12       = 0
```

```
Coin4_B01C23      = 0
Coin4_B12C01      = 0
Coin4_B12C12      = 0
Coin4_B12C23      = 0
Coin4_B23C01      = 0
Coin4_B23C12      = 0
Coin4_B23C23      = 0
in_det            = [0]*NumDetectors
in_any            = 0
h2A01B01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A01&B01", 92, -1, 91, 362, -1, 361)
h2A01B12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A01&B12", 92, -1, 91, 362, -1, 361)
h2A01B23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A01&B23", 92, -1, 91, 362, -1, 361)
h2A12B01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A12&B01", 92, -1, 91, 362, -1, 361)
h2A12B12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A12&B12", 92, -1, 91, 362, -1, 361)
h2A12B23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A12&B23", 92, -1, 91, 362, -1, 361)
h2A23B01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A23&B01", 92, -1, 91, 362, -1, 361)
h2A23B12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A23&B12", 92, -1, 91, 362, -1, 361)
h2A23B23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A23&B23", 92, -1, 91, 362, -1, 361)

h2A01C01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A01&C01", 92, -1, 91, 362, -1, 361)
h2A01C12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A01&C12", 92, -1, 91, 362, -1, 361)
h2A01C23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A01&C23", 92, -1, 91, 362, -1, 361)
h2A12C01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A12&C01", 92, -1, 91, 362, -1, 361)
h2A12C12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A12&C12", 92, -1, 91, 362, -1, 361)
h2A12C23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A12&C23", 92, -1, 91, 362, -1, 361)
h2A23C01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A23&C01", 92, -1, 91, 362, -1, 361)
h2A23C12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A23&C12", 92, -1, 91, 362, -1, 361)
h2A23C23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A23&C23", 92, -1, 91, 362, -1, 361)

h2B01C01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B01&C01", 92, -1, 91, 362, -1, 361)
h2B01C12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B01&C12", 92, -1, 91, 362, -1, 361)
h2B01C23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B01&C23", 92, -1, 91, 362, -1, 361)
h2B12C01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B12&C01", 92, -1, 91, 362, -1, 361)
h2B12C12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B12&C12", 92, -1, 91, 362, -1, 361)
h2B12C23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B12&C23", 92, -1, 91, 362, -1, 361)
h2B23C01          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B23&C01", 92, -1, 91, 362, -1, 361)
```

```
h2B23C12          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B23&C12", 92, -1, 91, 362, -1, 361)
h2B23C23          = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B23&C23", 92, -1, 91, 362, -1, 361)

h2AB              = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A&B", 92, -1, 91, 362, -1, 361)
h2AC              = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group A&C", 92, -1, 91, 362, -1, 361)
h2BC              = TH2F("Acceptance CRE4AT", "Histogram of CRE4AT
Detector Acceptance - Group B&C", 92, -1, 91, 362, -1, 361)
h2Gen             = TH1F("Distribution of #theta", "#theta Generated by
Monte Carlo Simulation for ZAfactor = %.2f"%(n_value_model_prod), 92,
-1, 91)
while ( iev < nEvent ) :
      if (iev%200000 == 0) :
            print("DATA GENERATION-->  iev = ",iev)

      # generating one muon track in production plane
      gen = GenerateMuons(n_value_model_prod)
      theta = (gen[3]*180)/(pi)
      phi         = (gen[2]*180)/(pi)
      h2Gen.Fill(theta)
      flagTrigger = [0]*NumDetectors

      for idet in range (0,NumDetectors) :

            ## first configuration
            channels = MuonInDetector(idet,gen,x,y,z,deltax,deltay)

            # coincidence between x and y of each detector
            if ((channels[0] != -1) and (channels[1] != -1)) :
                  in_det[idet]     += 1
                  flagTrigger[idet] = 1

      # logics
      if (flagTrigger[0]==1 or flagTrigger[1]==1 or flagTrigger[2]==1
or flagTrigger[3]==1 or flagTrigger[4]==1 or flagTrigger[5]==1 or
flagTrigger[6]==1 or flagTrigger[7]==1):
            in_any += 1

      # Coincidence 3
      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[2]==1):
            CoinA012 += 1
      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[3]==1):
            CoinA013 += 1
      if (flagTrigger[0]==1 and flagTrigger[2]==1 and
flagTrigger[3]==1):
            CoinA023 += 1
      if (flagTrigger[1]==1 and flagTrigger[2]==1 and
flagTrigger[3]==1):
            CoinA123 += 1

      if (flagTrigger[4]==1 and flagTrigger[5]==1 and
flagTrigger[6]==1):
            CoinB012 += 1
      if (flagTrigger[4]==1 and flagTrigger[5]==1 and
flagTrigger[7]==1):
            CoinB013 += 1
```

```
      if (flagTrigger[4]==1 and flagTrigger[6]==1 and
flagTrigger[7]==1):
            CoinB023 += 1
      if (flagTrigger[5]==1 and flagTrigger[6]==1 and
flagTrigger[7]==1):
            CoinB123 += 1

      if (flagTrigger[8]==1 and flagTrigger[9]==1 and
flagTrigger[10]==1):
            CoinC012 += 1
      if (flagTrigger[8]==1 and flagTrigger[9]==1 and
flagTrigger[11]==1):
            CoinC013 += 1
      if (flagTrigger[8]==1 and flagTrigger[10]==1 and
flagTrigger[11]==1):
            CoinC023 += 1
      if (flagTrigger[9]==1 and flagTrigger[10]==1 and
flagTrigger[11]==1):
            CoinC123 += 1

      # Coincidence 4
      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[2]==1 and flagTrigger[3]==1):
            Coin4A += 1
      if (flagTrigger[4]==1 and flagTrigger[5]==1 and
flagTrigger[6]==1 and flagTrigger[7]==1):
            Coin4B += 1
      if (flagTrigger[8]==1 and flagTrigger[9]==1 and
flagTrigger[10]==1 and flagTrigger[11]==1):
            Coin4C += 1

      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[4]==1 and flagTrigger[5]==1):
            Coin4_A01B01 +=1
            h2A01B01.Fill(theta,phi)
            h2AB.Fill(theta,phi)
      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[5]==1 and flagTrigger[6]==1):
            Coin4_A01B12 +=1
            h2A01B12.Fill(theta,phi)
            h2AB.Fill(theta,phi)
      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[6]==1 and flagTrigger[7]==1):
            Coin4_A01B23 +=1
            h2A01B23.Fill(theta,phi)
            h2AB.Fill(theta,phi)
      if (flagTrigger[1]==1 and flagTrigger[2]==1 and
flagTrigger[4]==1 and flagTrigger[5]==1):
            Coin4_A12B01 +=1
            h2A12B01.Fill(theta,phi)
            h2AB.Fill(theta,phi)
      if (flagTrigger[1]==1 and flagTrigger[2]==1 and
flagTrigger[5]==1 and flagTrigger[6]==1):
            Coin4_A12B12 +=1
            h2A12B12.Fill(theta,phi)
            h2AB.Fill(theta,phi)
      if (flagTrigger[1]==1 and flagTrigger[2]==1 and
flagTrigger[6]==1 and flagTrigger[7]==1):
            Coin4_A12B23 +=1
            h2A12B23.Fill(theta,phi)
            h2AB.Fill(theta,phi)
```

```
      if (flagTrigger[2]==1 and flagTrigger[3]==1 and
flagTrigger[4]==1 and flagTrigger[5]==1):
            Coin4_A23B01 +=1
            h2A23B01.Fill(theta,phi)
            h2AB.Fill(theta,phi)
      if (flagTrigger[2]==1 and flagTrigger[3]==1 and
flagTrigger[5]==1 and flagTrigger[6]==1):
            Coin4_A23B12 +=1
            h2A23B12.Fill(theta,phi)
            h2AB.Fill(theta,phi)
      if (flagTrigger[2]==1 and flagTrigger[3]==1 and
flagTrigger[6]==1 and flagTrigger[7]==1):
            Coin4_A23B23 +=1
            h2A23B23.Fill(theta,phi)
            h2AB.Fill(theta,phi)

      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[8]==1 and flagTrigger[9]==1):
            Coin4_A01C01 +=1
            h2A01C01.Fill(theta,phi)
            h2AC.Fill(theta,phi)
      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[9]==1 and flagTrigger[10]==1):
            Coin4_A01C12 +=1
            h2A01C12.Fill(theta,phi)
            h2AC.Fill(theta,phi)
      if (flagTrigger[0]==1 and flagTrigger[1]==1 and
flagTrigger[10]==1 and flagTrigger[11]==1):
            Coin4_A01C23 +=1
            h2A01C23.Fill(theta,phi)
            h2AC.Fill(theta,phi)
      if (flagTrigger[1]==1 and flagTrigger[2]==1 and
flagTrigger[8]==1 and flagTrigger[9]==1):
            Coin4_A12C01 +=1
            h2A12C01.Fill(theta,phi)
            h2AC.Fill(theta,phi)
      if (flagTrigger[1]==1 and flagTrigger[2]==1 and
flagTrigger[9]==1 and flagTrigger[10]==1):
            Coin4_A12C12 +=1
            h2A12C12.Fill(theta,phi)
            h2AC.Fill(theta,phi)
      if (flagTrigger[1]==1 and flagTrigger[2]==1 and
flagTrigger[10]==1 and flagTrigger[11]==1):
            Coin4_A12C23 +=1
            h2A12C23.Fill(theta,phi)
            h2AC.Fill(theta,phi)
      if (flagTrigger[2]==1 and flagTrigger[3]==1 and
flagTrigger[8]==1 and flagTrigger[9]==1):
            Coin4_A23C01 +=1
            h2A23C01.Fill(theta,phi)
            h2AC.Fill(theta,phi)
      if (flagTrigger[2]==1 and flagTrigger[3]==1 and
flagTrigger[9]==1 and flagTrigger[10]==1):
            Coin4_A23C12 +=1
            h2A23C12.Fill(theta,phi)
            h2AC.Fill(theta,phi)
      if (flagTrigger[2]==1 and flagTrigger[3]==1 and
flagTrigger[10]==1 and flagTrigger[11]==1):
            Coin4_A23C23 +=1
            h2A23C23.Fill(theta,phi)
            h2AC.Fill(theta,phi)
```

```
      if (flagTrigger[4]==1 and flagTrigger[5]==1 and
flagTrigger[8]==1 and flagTrigger[9]==1):
            Coin4_B01C01 +=1
            h2B01C01.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      if (flagTrigger[4]==1 and flagTrigger[5]==1 and
flagTrigger[9]==1 and flagTrigger[10]==1):
            Coin4_B01C12 +=1
            h2B01C12.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      if (flagTrigger[4]==1 and flagTrigger[5]==1 and
flagTrigger[10]==1 and flagTrigger[11]==1):
            Coin4_B01C23 +=1
            h2B01C23.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      if (flagTrigger[5]==1 and flagTrigger[6]==1 and
flagTrigger[8]==1 and flagTrigger[9]==1):
            Coin4_B12C01 +=1
            h2B12C01.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      if (flagTrigger[5]==1 and flagTrigger[6]==1 and
flagTrigger[9]==1 and flagTrigger[10]==1):
            Coin4_B12C12 +=1
            h2B12C12.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      if (flagTrigger[5]==1 and flagTrigger[6]==1 and
flagTrigger[10]==1 and flagTrigger[11]==1):
            Coin4_B12C23 +=1
            h2B12C23.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      if (flagTrigger[6]==1 and flagTrigger[7]==1 and
flagTrigger[8]==1 and flagTrigger[9]==1):
            Coin4_B23C01 +=1
            h2B23C01.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      if (flagTrigger[6]==1 and flagTrigger[7]==1 and
flagTrigger[9]==1 and flagTrigger[10]==1):
            Coin4_B23C12 +=1
            h2B23C12.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      if (flagTrigger[6]==1 and flagTrigger[7]==1 and
flagTrigger[10]==1 and flagTrigger[11]==1):
            Coin4_B23C23 +=1
            h2B23C23.Fill(theta,phi)
            h2BC.Fill(theta,phi)
      iev += 1

CheckFactor = (max(in_det)-min(in_det))/np.mean(in_det)
print("CHECK--> Check Logic verification --> %.5f"%(CheckFactor))
if (CheckFactor < 0.001):
      print("INFO: Logic verification pass successfully!")
else:
      print("WARNING: Logic verification higher than 0.1%!!!")

fGeo_Quad   = np.mean([Coin4A, Coin4B, Coin4C])
fGeo0_123   = np.mean([CoinA123, CoinB123, CoinC123])
fGeo1_023   = np.mean([CoinA023, CoinB023, CoinC023])
fGeo2_013   = np.mean([CoinA013, CoinB013, CoinC013])
fGeo3_012   = np.mean([CoinA012, CoinB012, CoinC012])
```

```
fGeo0 = fGeo0_123/fGeo_Quad
fGeo1 = fGeo1_023/fGeo_Quad
fGeo2 = fGeo2_013/fGeo_Quad
fGeo3 = fGeo3_012/fGeo_Quad
fFluxGeo = np.mean([in_det[0], in_det[4], in_det[8]])/fGeo_Quad

R_A01B01 = np.float64(Coin4_A01B01)/(np.float64(Coin4A+Coin4B))
R_A01B12 = np.float64(Coin4_A01B12)/(np.float64(Coin4A+Coin4B))
R_A01B23 = np.float64(Coin4_A01B23)/(np.float64(Coin4A+Coin4B))
R_A12B01 = np.float64(Coin4_A12B01)/(np.float64(Coin4A+Coin4B))
R_A12B12 = np.float64(Coin4_A12B12)/(np.float64(Coin4A+Coin4B))
R_A12B23 = np.float64(Coin4_A12B23)/(np.float64(Coin4A+Coin4B))
R_A23B01 = np.float64(Coin4_A23B01)/(np.float64(Coin4A+Coin4B))
R_A23B12 = np.float64(Coin4_A23B12)/(np.float64(Coin4A+Coin4B))
R_A23B23 = np.float64(Coin4_A23B23)/(np.float64(Coin4A+Coin4B))

R_A01C01 = np.float64(Coin4_A01C01)/(np.float64(Coin4A+Coin4C))
R_A01C12 = np.float64(Coin4_A01C12)/(np.float64(Coin4A+Coin4C))
R_A01C23 = np.float64(Coin4_A01C23)/(np.float64(Coin4A+Coin4C))
R_A12C01 = np.float64(Coin4_A12C01)/(np.float64(Coin4A+Coin4C))
R_A12C12 = np.float64(Coin4_A12C12)/(np.float64(Coin4A+Coin4C))
R_A12C23 = np.float64(Coin4_A12C23)/(np.float64(Coin4A+Coin4C))
R_A23C01 = np.float64(Coin4_A23C01)/(np.float64(Coin4A+Coin4C))
R_A23C12 = np.float64(Coin4_A23C12)/(np.float64(Coin4A+Coin4C))
R_A23C23 = np.float64(Coin4_A23C23)/(np.float64(Coin4A+Coin4C))

R_B01C01 = np.float64(Coin4_B01C01)/(np.float64(Coin4B+Coin4C))
R_B01C12 = np.float64(Coin4_B01C12)/(np.float64(Coin4B+Coin4C))
R_B01C23 = np.float64(Coin4_B01C23)/(np.float64(Coin4B+Coin4C))
R_B12C01 = np.float64(Coin4_B12C01)/(np.float64(Coin4B+Coin4C))
R_B12C12 = np.float64(Coin4_B12C12)/(np.float64(Coin4B+Coin4C))
R_B12C23 = np.float64(Coin4_B12C23)/(np.float64(Coin4B+Coin4C))
R_B23C01 = np.float64(Coin4_B23C01)/(np.float64(Coin4B+Coin4C))
R_B23C12 = np.float64(Coin4_B23C12)/(np.float64(Coin4B+Coin4C))
R_B23C23 = np.float64(Coin4_B23C23)/(np.float64(Coin4B+Coin4C))

print("DATA --
>\tfGeo0=%.5f\n\t\tfGeo1=%.5f\n\t\tfGeo2=%.5f\n\t\tfGeo3=%.5f\n\t\tfFl
uxGeo=%.5f\n\t\tR_A01B01=%.5f\n\t\tR_A01B12=%.5f\n\t\tR_A01B23=%.5f\n\
t\tR_A12B01=%.5f\n\t\tR_A12B12=%.5f\n\t\tR_A12B23=%.5f\n\t\tR_A23B01=%
.5f\n\t\tR_A23B12=%.5f\n\t\tR_A23B23=%.5f\n\t\tR_A01C01=%.5f\n\t\tR_A0
1C12=%.5f\n\t\tR_A01C23=%.5f\n\t\tR_A12C01=%.5f\n\t\tR_A12C12=%.5f\n\t
\tR_A12C23=%.5f\n\t\tR_A23C01=%.5f\n\t\tR_A23C12=%.5f\n\t\tR_A23C23=%.
5f\n\t\tR_B01C01=%.5f\n\t\tR_B01C12=%.5f\n\t\tR_B01C23=%.5f\n\t\tR_B12
C01=%.5f\n\t\tR_B12C12=%.5f\n\t\tR_B12C23=%.5f\n\t\tR_B23C01=%.5f\n\t\
tR_B23C12=%.5f\n\t\tR_B23C23=%.5f\n\n"%(fGeo0, fGeo1, fGeo2, fGeo3,
fFluxGeo, R_A01B01, R_A01B12, R_A01B23, R_A12B01, R_A12B12, R_A12B23,
R_A23B01, R_A23B12, R_A23B23, R_A01C01, R_A01C12, R_A01C23, R_A12C01,
R_A12C12, R_A12C23, R_A23C01, R_A23C12, R_A23C23, R_B01C01, R_B01C12,
R_B01C23, R_B12C01, R_B12C12, R_B12C23, R_B23C01, R_B23C12, R_B23C23))
outFile = open(result_file,"w")
outFile.write("ZenithAngleFactor\tCheckFactor\tfGeo0\tfGeo1\tfGeo2\tfG
eo3\tfFluxGeo\tR_A01B01\tR_A01B12\tR_A01B23\tR_A12B01\tR_A12B12\tR_A12
B23\tR_A23B01\tR_A23B12\tR_A23B23\tR_A01C01\tR_A01C12\tR_A01C23\tR_A12
C01\tR_A12C12\tR_A12C23\tR_A23C01\tR_A23C12\tR_A23C23\tR_B01C01\tR_B01
C12\tR_B01C23\tR_B12C01\tR_B12C12\tR_B12C23\tR_B23C01\tR_B23C12\tR_B23
C23\tCoin4A\tCoin4B\tCoin4C\tCoin4_A01B01\tCoin4_A01B12\tCoin4_A01B23\
tCoin4_A12B01\tCoin4_A12B12\tCoin4_A12B23\tCoin4_A23B01\tCoin4_A23B12\
tCoin4_A23B23\tCoin4_A01C01\tCoin4_A01C12\tCoin4_A01C23\tCoin4_A12C01\
tCoin4_A12C12\tCoin4_A12C23\tCoin4_A23C01\tCoin4_A23C12\tCoin4_A23C23\
tCoin4_B01C01\tCoin4_B01C12\tCoin4_B01C23\tCoin4_B12C01\tCoin4_B12C12\
```

```
tCoin4_B12C23\tCoin4_B23C01\tCoin4_B23C12\tCoin4_B23C23\tSingleCh0A\tS
ingleCh0B\tSingleCh0C\n")
outFile.write("%.3f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%
.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\
t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5
f\t%.5f\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%
d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\
n"%(n_value_model_prod, CheckFactor, fGeo0, fGeo1, fGeo2, fGeo3,
fFluxGeo, R_A01B01, R_A01B12, R_A01B23, R_A12B01, R_A12B12, R_A12B23,
R_A23B01, R_A23B12, R_A23B23, R_A01C01, R_A01C12, R_A01C23, R_A12C01,
R_A12C12, R_A12C23, R_A23C01, R_A23C12, R_A23C23, R_B01C01, R_B01C12,
R_B01C23, R_B12C01, R_B12C12, R_B12C23, R_B23C01, R_B23C12, R_B23C23,
Coin4A, Coin4B, Coin4C, Coin4_A01B01, Coin4_A01B12, Coin4_A01B23,
Coin4_A12B01, Coin4_A12B12, Coin4_A12B23, Coin4_A23B01, Coin4_A23B12,
Coin4_A23B23, Coin4_A01C01, Coin4_A01C12, Coin4_A01C23, Coin4_A12C01,
Coin4_A12C12, Coin4_A12C23, Coin4_A23C01, Coin4_A23C12, Coin4_A23C23,
Coin4_B01C01, Coin4_B01C12, Coin4_B01C23, Coin4_B12C01, Coin4_B12C12,
Coin4_B12C23, Coin4_B23C01, Coin4_B23C12, Coin4_B23C23, in_det[0],
in_det[4], in_det[8]))
outFile.close()

h2Gen.GetXaxis().SetTitle("#theta[Degrees]")
h2Gen.GetYaxis().SetTitle("Bins")

h2A01B01.SetFillColor(2)
h2A01B12.SetFillColor(3)
h2A01B23.SetFillColor(4)
h2A12B01.SetFillColor(5)
h2A12B12.SetFillColor(6)
h2A12B23.SetFillColor(7)
h2A23B01.SetFillColor(41)
h2A23B12.SetFillColor(40)
h2A23B23.SetFillColor(15)

h2A01C01.SetFillColor(2)
h2A01C12.SetFillColor(3)
h2A01C23.SetFillColor(4)
h2A12C01.SetFillColor(5)
h2A12C12.SetFillColor(6)
h2A12C23.SetFillColor(7)
h2A23C01.SetFillColor(41)
h2A23C12.SetFillColor(40)
h2A23C23.SetFillColor(15)

h2B01C01.SetFillColor(2)
h2B01C12.SetFillColor(3)
h2B01C23.SetFillColor(4)
h2B12C01.SetFillColor(5)
h2B12C12.SetFillColor(6)
h2B12C23.SetFillColor(7)
h2B23C01.SetFillColor(41)
h2B23C12.SetFillColor(40)
h2B23C23.SetFillColor(15)

h2A01B01.GetXaxis().SetTitle("#theta [Degrees]")
h2A01B01.GetYaxis().SetTitle("#phi [Degrees]")
h2A01B12.GetXaxis().SetTitle("#theta [Degrees]")
h2A01B12.GetYaxis().SetTitle("#phi [Degrees]")
h2A01B23.GetXaxis().SetTitle("#theta [Degrees]")
h2A01B23.GetYaxis().SetTitle("#phi [Degrees]")
h2A12B01.GetXaxis().SetTitle("#theta [Degrees]")
```

```
h2A12B01.GetYaxis().SetTitle("#phi [Degrees]")
h2A12B12.GetXaxis().SetTitle("#theta [Degrees]")
h2A12B12.GetYaxis().SetTitle("#phi [Degrees]")
h2A12B23.GetXaxis().SetTitle("#theta [Degrees]")
h2A12B23.GetYaxis().SetTitle("#phi [Degrees]")
h2A23B01.GetXaxis().SetTitle("#theta [Degrees]")
h2A23B01.GetYaxis().SetTitle("#phi [Degrees]")
h2A23B12.GetXaxis().SetTitle("#theta [Degrees]")
h2A23B12.GetYaxis().SetTitle("#phi [Degrees]")
h2A23B23.GetXaxis().SetTitle("#theta [Degrees]")
h2A23B23.GetYaxis().SetTitle("#phi [Degrees]")
h2AB.GetXaxis().SetTitle("#theta [Degrees]")
h2AB.GetYaxis().SetTitle("#phi [Degrees]")

h2A01C01.GetXaxis().SetTitle("#theta [Degrees]")
h2A01C01.GetYaxis().SetTitle("#phi [Degrees]")
h2A01C12.GetXaxis().SetTitle("#theta [Degrees]")
h2A01C12.GetYaxis().SetTitle("#phi [Degrees]")
h2A01C23.GetXaxis().SetTitle("#theta [Degrees]")
h2A01C23.GetYaxis().SetTitle("#phi [Degrees]")
h2A12C01.GetXaxis().SetTitle("#theta [Degrees]")
h2A12C01.GetYaxis().SetTitle("#phi [Degrees]")
h2A12C12.GetXaxis().SetTitle("#theta [Degrees]")
h2A12C12.GetYaxis().SetTitle("#phi [Degrees]")
h2A12C23.GetXaxis().SetTitle("#theta [Degrees]")
h2A12C23.GetYaxis().SetTitle("#phi [Degrees]")
h2A23C01.GetXaxis().SetTitle("#theta [Degrees]")
h2A23C01.GetYaxis().SetTitle("#phi [Degrees]")
h2A23C12.GetXaxis().SetTitle("#theta [Degrees]")
h2A23C12.GetYaxis().SetTitle("#phi [Degrees]")
h2A23C23.GetXaxis().SetTitle("#theta [Degrees]")
h2A23C23.GetYaxis().SetTitle("#phi [Degrees]")
h2AC.GetXaxis().SetTitle("#theta [Degrees]")
h2AC.GetYaxis().SetTitle("#phi [Degrees]")

h2B01C01.GetXaxis().SetTitle("#theta [Degrees]")
h2B01C01.GetYaxis().SetTitle("#phi [Degrees]")
h2B01C12.GetXaxis().SetTitle("#theta [Degrees]")
h2B01C12.GetYaxis().SetTitle("#phi [Degrees]")
h2B01C23.GetXaxis().SetTitle("#theta [Degrees]")
h2B01C23.GetYaxis().SetTitle("#phi [Degrees]")
h2B12C01.GetXaxis().SetTitle("#theta [Degrees]")
h2B12C01.GetYaxis().SetTitle("#phi [Degrees]")
h2B12C12.GetXaxis().SetTitle("#theta [Degrees]")
h2B12C12.GetYaxis().SetTitle("#phi [Degrees]")
h2B12C23.GetXaxis().SetTitle("#theta [Degrees]")
h2B12C23.GetYaxis().SetTitle("#phi [Degrees]")
h2B23C01.GetXaxis().SetTitle("#theta [Degrees]")
h2B23C01.GetYaxis().SetTitle("#phi [Degrees]")
h2B23C12.GetXaxis().SetTitle("#theta [Degrees]")
h2B23C12.GetYaxis().SetTitle("#phi [Degrees]")
h2B23C23.GetXaxis().SetTitle("#theta [Degrees]")
h2B23C23.GetYaxis().SetTitle("#phi [Degrees]")
h2BC.GetXaxis().SetTitle("#theta [Degrees]")
h2BC.GetYaxis().SetTitle("#phi [Degrees]")

f     =
TFile("./Hists_CRE4AT_ZAf%.2f.root"%(n_value_model_prod),"RECREATE")
h2Gen.Write("Hist_ThetaGen_ZAf%.2f"%(n_value_model_prod))

h2A01B01.Write("Hist2d_A01B01_ZAf%.2f"%(n_value_model_prod))
```

macro-CRE4AT_groupsABC_IPANEMA_SingleZAFactor_Histograms.py

```
h2A01B12.Write("Hist2d_A01B12_ZAf%.2f"%(n_value_model_prod))
h2A01B23.Write("Hist2d_A01B23_ZAf%.2f"%(n_value_model_prod))
h2A12B01.Write("Hist2d_A12B01_ZAf%.2f"%(n_value_model_prod))
h2A12B12.Write("Hist2d_A12B12_ZAf%.2f"%(n_value_model_prod))
h2A12B23.Write("Hist2d_A12B23_ZAf%.2f"%(n_value_model_prod))
h2A23B01.Write("Hist2d_A23B01_ZAf%.2f"%(n_value_model_prod))
h2A23B12.Write("Hist2d_A23B12_ZAf%.2f"%(n_value_model_prod))
h2A23B23.Write("Hist2d_A23B23_ZAf%.2f"%(n_value_model_prod))
h2AB.Write("Hist2d_A&B_ZAf%.2f"%(n_value_model_prod))

h2A01C01.Write("Hist2d_A01C01_ZAf%.2f"%(n_value_model_prod))
h2A01C12.Write("Hist2d_A01C12_ZAf%.2f"%(n_value_model_prod))
h2A01C23.Write("Hist2d_A01C23_ZAf%.2f"%(n_value_model_prod))
h2A12C01.Write("Hist2d_A12C01_ZAf%.2f"%(n_value_model_prod))
h2A12C12.Write("Hist2d_A12C12_ZAf%.2f"%(n_value_model_prod))
h2A12C23.Write("Hist2d_A12C23_ZAf%.2f"%(n_value_model_prod))
h2A23C01.Write("Hist2d_A23C01_ZAf%.2f"%(n_value_model_prod))
h2A23C12.Write("Hist2d_A23C12_ZAf%.2f"%(n_value_model_prod))
h2A23C23.Write("Hist2d_A23C23_ZAf%.2f"%(n_value_model_prod))
h2AC.Write("Hist2d_A&C_ZAf%.2f"%(n_value_model_prod))

h2B01C01.Write("Hist2d_B01C01_ZAf%.2f"%(n_value_model_prod))
h2B01C12.Write("Hist2d_B01C12_ZAf%.2f"%(n_value_model_prod))
h2B01C23.Write("Hist2d_B01C23_ZAf%.2f"%(n_value_model_prod))
h2B12C01.Write("Hist2d_B12C01_ZAf%.2f"%(n_value_model_prod))
h2B12C12.Write("Hist2d_B12C12_ZAf%.2f"%(n_value_model_prod))
h2B12C23.Write("Hist2d_B12C23_ZAf%.2f"%(n_value_model_prod))
h2B23C01.Write("Hist2d_B23C01_ZAf%.2f"%(n_value_model_prod))
h2B23C12.Write("Hist2d_B23C12_ZAf%.2f"%(n_value_model_prod))
h2B23C23.Write("Hist2d_B23C23_ZAf%.2f"%(n_value_model_prod))
h2BC.Write("Hist2d_B&C_ZAf%.2f"%(n_value_model_prod))
f.Close()
```

```
#######################################################################
# Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:           Concat_ZAFTable.py
# Author:             Diogo Ayres Rocha (dayres@cbpf.br)
# Date:               10-08-2023
#######################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 10 Aug 23  Diogo Ayres Rocha  0001   File Created
#######################################################################

import pandas as pd
import sys
import os
OutFilename = sys.argv[1]
files=os.listdir('./')
files.sort()
data = []
for file in files:
      if (('CRE4AT' in file) and ('.dat' in file)):
            data.append(pd.read_csv(file, sep='\t'))
Data = pd.concat(data)
Data.reset_index(drop=True, inplace=True)
Data.to_csv(OutFilename, sep='\t', float_format="%.5f", index=False)
```

Concat_ZAFTable.py

Apêndice N

Programa desenvolvido para realizar a comparação das distribuições de θ em função do valor de referência

Este apêndice é composto pelo código desenvolvido em *python3*, com uso do *framework* ROOT/CERN na versão 6.26/04, que realiza a comparação entre histogramas de θ, gerados para diferentes valores de ZAF.

```
#######################################################################
# Company:              CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:             CRE4AT_ThetaDiff_Analysis.py
# Author:               Diogo Ayres Rocha (dayres@cbpf.br)
# Date:                 29-12-2023
#######################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 29 Dec 23  Diogo Ayres Rocha  0001   File Created
#######################################################################

from ROOT import TFile, TH1F, TCanvas, gStyle, gROOT
import sys
FileRef     = sys.argv[1]
File2Diff   = sys.argv[2]
ZAfRef      = float(FileRef.split('_')[-
1].replace("ZAf","").replace(".root",""))
ZAf2Diff    = float(File2Diff.split('_')[-
1].replace("ZAf","").replace(".root",""))
fRef        = TFile(FileRef,"READ")
f2Diff      = TFile(File2Diff,"READ")
hRef        = fRef.Get("Hist_ThetaGen_ZAf%.2f"%(ZAfRef))
h2Diff      = f2Diff.Get("Hist_ThetaGen_ZAf%.2f"%(ZAf2Diff))
gStyle.SetOptStat(0)
gROOT.SetBatch(1)
hDiff       = h2Diff.Clone()
hDiff.Add(hRef, -1)
hDiff.Divide(hRef)
hDiff.SetTitle("Normalized Diference of #theta Distribution Generated
by Monte Carlo for ZAf=%.2f and ZAf=%.2f"%(ZAf2Diff, ZAfRef))
hDiff.GetXaxis().SetRangeUser(0,90)
hDiff.GetXaxis().SetTitle("#theta [Degrees]")
hDiff.GetYaxis().SetTitle("DiffBins Normalized")
hDiff.GetXaxis().SetRangeUser(0,90)
c1                = TCanvas("#theta Distribution","#theta
Distribution", 1280,800)
hDiff.Draw()
c1.Print("ThetaDiff_ZAf%.2f_ZAfRef=%.2f.png"%(ZAf2Diff, ZAfRef))
```

Apêndice O

Programas desenvolvidos para realizar a validação do método de procura do ZAF

Este apêndice é composto pelos códigos desenvolvidos em *python3*, com uso do *framework* ROOT/CERN, na versão 6.26/04, que realizam a análise dos diferentes testes executados para validação do método criado para busca do valor do ZAF.

Os programas serão expostos na seguinte ordem:

1. "FindZAF.py", responsável por realizar a procura do fator, utilizando a tabela de busca;
2. "CheckBestTime_FindZAf.py", reponsável por realizar a avaliação do método com diferentes períodos de tempo de aquisição;
3. "Check_ChangingPositionError_FindZAf.py", responsável por avaliar o erro causado na busca do ZAF em função do desvio do posicionamento dos grupos, em relação ao posicionamento simulado na tabela de busca.

Para execução desses códigos, algumas bibliotecas *python* são necessárias, nas seguintes versões:

1. Numpy, versão 1.19.5;
2. Pandas, versão 1.1.5;
3. tqdm, versão 4.66.1.

```
#######################################################################
# Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:           FindZAF.py
# Author:             Diogo Ayres Rocha (dayres@cbpf.br)
# Date:               08-06-2023
#######################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 08 Jun 23  Diogo Ayres Rocha  0001   File Created
# 27 Dec 23  Diogo Ayres Rocha  0002   Plot Ajustments
#######################################################################

import pandas as pd
import numpy as np
import sys, os
from ROOT import TCanvas, TGraph, TLatex, TF1, TPad, TPaveText

R_Columns = ["R_A01B01", "R_A01B12", "R_A01B23", "R_A12B01",
"R_A12B12", "R_A12B23", "R_A23B01", "R_A23B12", "R_A23B23",
"R_A01C01", "R_A01C12", "R_A01C23", "R_A12C01", "R_A12C12",
"R_A12C23", "R_A23C01", "R_A23C12", "R_A23C23", "R_B01C01",
"R_B01C12", "R_B01C23", "R_B12C01", "R_B12C12", "R_B12C23",
"R_B23C01", "R_B23C12" ,"R_B23C23"]

SearchFile  = sys.argv[1]
DataFile    = sys.argv[2]
OutFolder   = sys.argv[3]
ZAfSimulated      = DataFile.split('_')[-1].replace(".dat","")
ZAfSimulatedNum   = float(ZAfSimulated.replace("ZAfactor",""))

SearchData  = pd.read_csv(SearchFile, sep="\t")
DataDf            = pd.read_csv(DataFile, sep="\t")

ZAfFound    = 0
g                 = TGraph(len(SearchData))

for pos, N2find in enumerate(SearchData.ZenithAngleFactor.values):
      MapZenSelData     =
((SearchData.loc[SearchData["ZenithAngleFactor"] ==
N2find]).reset_index(drop=True))
      chi2Sum                 = ((MapZenSelData[R_Columns] -
DataDf[R_Columns])**2).values.sum()/27
      g.SetPoint(pos,N2find,chi2Sum)
      if (pos == 0):
            LowestNfSum = chi2Sum
            ZAfFound    = N2find
      if (chi2Sum < LowestNfSum):
            LowestNfSum = chi2Sum
            ZAfFound    = N2find
print("Nfactor found: %.2f"%(ZAfFound))
c           = TCanvas("Chi2Sum", "Chi2Sum", 1280, 800)
g.GetXaxis().SetTitle("Zenith Angle Factor")
g.GetYaxis().SetTitle("#chi^{2}")
g.SetMarkerSize(1)
g.SetMarkerStyle(21)
g.SetMarkerColor(2)
g.SetTitle("Zenith Angle Factor %.2f Search"%(ZAfSimulatedNum))
fPol2 = TF1("fPol2", "pol2")
fPol2.SetLineColor(4)
g.Fit(fPol2, "", "", ZAfFound-(20*0.01), ZAfFound+(20*0.01))
g.Draw("AP")
```

FindZAF.py

```
padZoom = TPad("p","p", 0.6,0.5,0.98,0.92)
padZoom.SetLeftMargin(0.05)
padZoom.SetRightMargin(0.01)
padZoom.SetBottomMargin(0.05)
padZoom.SetTopMargin(0.32)
padZoom.Draw()
padZoom.cd()
gZAfcp      = g.DrawClone("AP")
gZAfcp.SetTitle("")
gZAfcp.GetXaxis().SetRangeUser(ZAfFound-(20*0.01),ZAfFound+(20*0.01))
min         = g.GetYaxis().GetXmin()
max         = g.GetYaxis().GetXmax()
gZAfcp.GetYaxis().SetRangeUser(min-min/30,max/80)
gZAfcp.GetXaxis().SetLabelSize(0.05)
gZAfcp.GetYaxis().SetLabelSize(0.05)
gZAfcp.GetXaxis().SetTitle("")
gZAfcp.GetYaxis().SetTitle("")
gZAfcp.Draw("AP")
ZAf         = (-fPol2.GetParameter(1))/(2*fPol2.GetParameter(2))
ZAfErr      = ZAf *
((((fPol2.GetParError(1)/fPol2.GetParameter(1))**2)+((fPol2.GetParErro
r(2)/fPol2.GetParameter(2))**2))**(1/2))

pt = TPaveText(0,0.75,0.99,1)
pt.AddText("Simple Zenith Angle Factor Found = %.2f"%(ZAfFound))
pt.AddText("Fit Function:
%.6f*(ZA_{f})^{2}%.6f*ZA_{f}+%.6f"%(fPol2.GetParameter(2),
fPol2.GetParameter(1), fPol2.GetParameter(0)))
pt.AddText("Zenith Angle Factor Found by Fit = %.4f +- %.4f"%(ZAf,
ZAfErr))
pt.Paint("NDC")
pt.Draw()
c.Print(os.path.join(OutFolder,"ZAF_Find_Curve_%s.png"%(ZAfSimulated))
)T.png")
```

FindZAF.py

```
#######################################################################
# Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:           CheckBestTime_FindZAf.py
# Author:             Diogo Ayres Rocha (dayres@cbpf.br)
# Date:               29-10-2023
#######################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 29 Oct 23  Diogo Ayres Rocha  0001   File Created
# 27 Dec 23  Diogo Ayres Rocha  0002   Plot Ajustments
#######################################################################

import pandas as pd
import numpy as np
import sys, os
from ROOT import TCanvas, TGraph, TGraphErrors, TLatex, TPad,
TPaveText, TF1

MapFilename       = sys.argv[1]
DataFilename      = sys.argv[2]
outFolder         = sys.argv[3]
ZAfSimulated      = DataFilename.split('_')[-1].replace(".dat","")
ZAfSimulatedNum   = float(ZAfSimulated.replace("ZAfactor",""))
R_Columns = ["R_A01B01", "R_A01B12", "R_A01B23", "R_A12B01",
"R_A12B12", "R_A12B23", "R_A23B01", "R_A23B12", "R_A23B23",
"R_A01C01", "R_A01C12", "R_A01C23", "R_A12C01", "R_A12C12",
"R_A12C23", "R_A23C01", "R_A23C12", "R_A23C23", "R_B01C01",
"R_B01C12", "R_B01C23", "R_B12C01", "R_B12C12", "R_B12C23",
"R_B23C01", "R_B23C12" ,"R_B23C23"]

MapDf = pd.read_csv(MapFilename, sep="\t")
DataDf      = pd.read_csv(DataFilename, sep="\t")

X           = np.asarray(DataDf.Hour).astype(float)
Y           = np.zeros(len(X))
YFit  = np.zeros(len(X))
YFitErr     = np.zeros(len(X))

for gpos, hours in enumerate(X):
      DataTimeLine      = (DataDf.loc[DataDf["Hour"] ==
hours]).reset_index(drop=True)
      ZAf2Find                =
DataTimeLine.ZenithAngleFactor.values[0]
      ZAfFound                = 0
      gZAf                    =
TGraph(len(MapDf.ZenithAngleFactor.values))
      for pos, ZAf2find in enumerate(MapDf.ZenithAngleFactor.values):
            MapZenSelData     = (MapDf.loc[MapDf["ZenithAngleFactor"]
== ZAf2find]).reset_index(drop=True)
            chi2Sum                 = ((MapZenSelData[R_Columns] -
DataTimeLine[R_Columns])**2).values.sum()/27
            gZAf.SetPoint(pos,ZAf2find,chi2Sum)
            #print("Hour %d - CheckZAf: %.2f - chi2Sum: %.4f"%(hours,
ZAf2find, chi2Sum))
            if (pos == 0):
                  LowestNfSum = chi2Sum
                  ZAfFound    = ZAf2find
            if (chi2Sum < LowestNfSum):
                  LowestNfSum = chi2Sum
                  ZAfFound    = ZAf2find
```

```
      print("Nfactor found during %d hour(s) data: %.2f"%(hours,
ZAfFound))
      c     = TCanvas("Chi2Sum %dh"%(hours), "Chi2Sum %dh"%(hours),
1280, 800)
      gZAf.GetXaxis().SetTitle("Zenith Angle Factor")
      gZAf.GetYaxis().SetTitle("#chi^{2} Sum")
      gZAf.SetMarkerSize(1)
      gZAf.SetMarkerStyle(21)
      gZAf.SetMarkerColor(2)
      gZAf.SetTitle("Zenith Angle Factor %.2f Search for %d
Hour(s)"%(ZAfSimulatedNum, hours))
      fPol2 = TF1("fPol2", "pol2")
      fPol2.SetLineColor(4)
      gZAf.Fit(fPol2, "", "", ZAfFound-(20*0.01), ZAfFound+(20*0.01))
      gZAf.Draw("AP")
      padZoom = TPad("p","p", 0.6,0.5,0.98,0.92)
      padZoom.SetLeftMargin(0.05)
      padZoom.SetRightMargin(0.01)
      padZoom.SetBottomMargin(0.05)
      padZoom.SetTopMargin(0.32)
      padZoom.Draw()
      padZoom.cd()
      gZAfcp = gZAf.DrawClone("AP")
      gZAfcp.SetTitle("")
      gZAfcp.GetXaxis().SetRangeUser(ZAfFound-
(20*0.01),ZAfFound+(20*0.01))
      min = gZAf.GetYaxis().GetXmin()
      max = gZAf.GetYaxis().GetXmax()
      gZAfcp.GetYaxis().SetRangeUser(min-min/30,max/30)
      gZAfcp.GetXaxis().SetLabelSize(0.05)
      gZAfcp.GetYaxis().SetLabelSize(0.05)
      gZAfcp.GetXaxis().SetTitle("")
      gZAfcp.GetYaxis().SetTitle("")
      gZAfcp.Draw("AP")

      ZAf = (-fPol2.GetParameter(1))/(2*fPol2.GetParameter(2))
      YFit[gpos]        = abs(ZAf-ZAfSimulatedNum)/ZAfSimulatedNum
      ZAfErr      = ZAf *
((((fPol2.GetParError(1)/fPol2.GetParameter(1))**2)+((fPol2.GetParErro
r(2)/fPol2.GetParameter(2))**2))**(1/2))
      YFitErr[gpos]     = ZAfErr/ZAfSimulatedNum

      pt = TPaveText(0,0.75,0.99,1)
      pt.AddText("Simple Zenith Angle Factor Found = %.2f"%(ZAfFound))
      pt.AddText("Fit Function:
%.6f*(ZA_{f})^{2}%.6f*ZA_{f}+%.6f"%(fPol2.GetParameter(2),
fPol2.GetParameter(1), fPol2.GetParameter(0)))
      pt.AddText("Zenith Angle Factor Found by Fit = %.4f"%(ZAf))
      pt.Paint("NDC")
      pt.Draw()
      if (gpos == 0):
            PdfFilename = os.path.join(outFolder,
"ZAfFind_Curve_%s.pdf("%(ZAfSimulated))
      elif (gpos == (len(X) - 1)):
            PdfFilename = os.path.join(outFolder,
"ZAfFind_Curve_%s.pdf)"%(ZAfSimulated))
      else:
            PdfFilename = os.path.join(outFolder,
"ZAfFind_Curve_%s.pdf"%(ZAfSimulated))
      c.Print(PdfFilename)
      Y[gpos]=abs(ZAfFound-ZAfSimulatedNum)/ZAfSimulatedNum
```

CheckBestTime_FindZAf.py

```
c     = TCanvas("ZA_{f} Search - Check Best DataTime to Find (Minimum
Method)", "ZA_{f} Search - Check Best DataTime to Find (Minimum
Method)", 1280, 800)
g     = TGraph(len(X), X, Y)
g.GetXaxis().SetTitle("DataTime[Hours]")
g.GetYaxis().SetTitle("(|ZAf_{Found} -
ZAf_{Simulated}|)/ZAf_{Simulated}")
g.SetMarkerSize(1)
g.SetMarkerStyle(20)
g.SetMarkerColor(2)
g.SetTitle("ZA_{f} %.2f Search - Check Best DataTime to Find (Minimum
Method)"%(ZAfSimulatedNum))
g.Draw("AP")
c.Print(os.path.join(outFolder,"ZAfFind_Curve_HoursStudy_MinMethod_%s.
png"%(ZAfSimulated)))

c     = TCanvas("ZA_{f} Search - Check Best DataTime to Find (Fit
Method)", "ZA_{f} Search - Check Best DataTime to Find (Fit Method)",
1280, 800)
g     = TGraphErrors(len(X), X, YFit, np.zeros(len(X)), YFitErr)
g.GetXaxis().SetTitle("DataTime[Hours]")
g.GetYaxis().SetTitle("(|ZAf_{Found} -
ZAf_{Simulated}|)/ZAf_{Simulated}")
g.SetMarkerSize(1)
g.SetMarkerStyle(20)
g.SetMarkerColor(2)
g.SetTitle("ZA_{f} %.2f Search - Check Best DataTime to Find (Fit
Method)"%(ZAfSimulatedNum))
g.Draw("AP")
c.Print(os.path.join(outFolder,"ZAfFind_Curve_HoursStudy_FitMethod_%s.
png"%(ZAfSimulated)))
```

CheckBestTime_FindZAf.py

```
#######################################################################
# Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename:           Check_ChangingPositionError_FindZAf.py
# Author:             Diogo Ayres Rocha (dayres@cbpf.br)
# Date:               23-10-2023
#######################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 23 Oct 23  Diogo Ayres Rocha  0001   File Created
# 29 Dec 23  Diogo Ayres Rocha  0002   Plot Ajustments
#######################################################################

import pandas as pd
import numpy as np
import sys, os
from ROOT import TCanvas, TGraph, TLatex, TPad, TPaveText, TF1,
TProfile, gROOT, gStyle, TLegend
from tqdm import tqdm

gROOT.SetBatch(1)

MapFilename       = sys.argv[1]
DataFilename      = sys.argv[2]
outFolder         = sys.argv[3]
R_Columns = ["R_A01B01", "R_A01B12", "R_A01B23", "R_A12B01",
"R_A12B12", "R_A12B23", "R_A23B01", "R_A23B12", "R_A23B23",
"R_A01C01", "R_A01C12", "R_A01C23", "R_A12C01", "R_A12C12",
"R_A12C23", "R_A23C01", "R_A23C12", "R_A23C23", "R_B01C01",
"R_B01C12", "R_B01C23", "R_B12C01", "R_B12C12", "R_B12C23",
"R_B23C01", "R_B23C12" ,"R_B23C23"]

MapDf = pd.read_csv(MapFilename, sep="\t")
DataDf      = pd.read_csv(DataFilename, sep="\t")

X           = np.zeros(len(DataDf))
XWAbs = np.zeros(len(DataDf))
YMin  = np.zeros(len(X))
YFit  = np.zeros(len(X))
YFitErr     = np.zeros(len(X))

Run = DataDf["Run"].values

for gpos in tqdm(range(0, len(Run))):
      RunNum = Run[gpos]
      DifPosLine        = (DataDf.loc[DataDf["Run"] ==
RunNum]).reset_index(drop=True)
      ZAfSimulatedNum   = DifPosLine.ZenithAngleFactor.values[0]
      ZAfFound          = 0
      gZAf              = TGraph(len(MapDf.ZenithAngleFactor.values))
      for pos, ZAf2Find in enumerate(MapDf.ZenithAngleFactor.values):
            MapZenSelData     = (MapDf.loc[MapDf["ZenithAngleFactor"]
== ZAf2Find]).reset_index(drop=True)
            chi2Sum                 = ((MapZenSelData[R_Columns] -
DifPosLine[R_Columns])**2).values.sum()/27
            gZAf.SetPoint(pos,ZAf2Find,chi2Sum)
            #print("Hour %d - CheckNp: %.2f - chi2Sum: %.4f"%(hours,
ZAf2Find, chi2Sum))
            if (pos == 0):
                  LowestNfSum = chi2Sum
                  ZAfFound    = ZAf2Find
            if (chi2Sum < LowestNfSum):
```

```
                LowestNfSum = chi2Sum
                ZAfFound    = ZAf2Find
        print("ZAfactor found on Run%d: %.2f"%(RunNum, ZAfFound))
        X[gpos] = abs(DifPosLine["GroupA_dX"][0]) +
abs(DifPosLine["GroupA_dY"][0]) + abs(DifPosLine["GroupA_dZ"][0]) +
abs(DifPosLine["GroupB_dX"][0]) + abs(DifPosLine["GroupB_dY"][0]) +
abs(DifPosLine["GroupB_dZ"][0]) + abs(DifPosLine["GroupC_dX"][0]) +
abs(DifPosLine["GroupC_dY"][0]) + abs(DifPosLine["GroupC_dZ"][0])
        XWAbs[gpos] = DifPosLine["GroupA_dX"][0] +
DifPosLine["GroupA_dY"][0] + DifPosLine["GroupA_dZ"][0] +
DifPosLine["GroupB_dX"][0] + DifPosLine["GroupB_dY"][0] +
DifPosLine["GroupB_dZ"][0] + DifPosLine["GroupC_dX"][0] +
DifPosLine["GroupC_dY"][0] + DifPosLine["GroupC_dZ"][0]
        YMin[gpos] = abs(ZAfFound-ZAfSimulatedNum)/ZAfSimulatedNum
        c     = TCanvas("Chi2 Run%d"%(RunNum), "Chi2 Run%d"%(RunNum),
1280, 800)
        gZAf.GetXaxis().SetTitle("Zenith Angle Factor")
        gZAf.GetYaxis().SetTitle("#chi^{2}")
        gZAf.SetMarkerSize(1)
        gZAf.SetMarkerStyle(21)
        gZAf.SetMarkerColor(2)
        gZAf.SetTitle("Zenith Angle Factor %.2f Search for
Run%d"%(ZAfSimulatedNum, RunNum))
        fPol2 = TF1("fPol2", "pol2")
        fPol2.SetLineColor(4)
        gZAf.Fit(fPol2, "", "", ZAfFound-(20*0.01), ZAfFound+(20*0.01))
        gZAf.Draw("AP")
        padZoom = TPad("p","p", 0.6,0.5,0.98,0.92)
        padZoom.SetLeftMargin(0.05)
        padZoom.SetRightMargin(0.01)
        padZoom.SetBottomMargin(0.05)
        padZoom.SetTopMargin(0.32)
        padZoom.Draw()
        padZoom.cd()
        gZAfcp = gZAf.DrawClone("AP")
        gZAfcp.SetTitle("")
        gZAfcp.GetXaxis().SetRangeUser(ZAfFound-
(20*0.01),ZAfFound+(20*0.01))
        min = gZAf.GetYaxis().GetXmin()
        max = gZAf.GetYaxis().GetXmax()
        gZAfcp.GetYaxis().SetRangeUser(min-min/30,max/20)
        gZAfcp.GetXaxis().SetLabelSize(0.05)
        #gZAfcp.GetXaxis().SetTickSize(0.)
        gZAfcp.GetYaxis().SetLabelSize(0.05)
        #gZAfcp.GetYaxis().SetTickSize(0.)
        gZAfcp.GetXaxis().SetTitle("")
        gZAfcp.GetYaxis().SetTitle("")
        gZAfcp.Draw("AP")
        ZAf                     = (-
fPol2.GetParameter(1))/(2*fPol2.GetParameter(2))
        YFit[gpos]        = abs(ZAf-ZAfSimulatedNum)/ZAfSimulatedNum
        ZAfErr      = ZAf *
((((fPol2.GetParError(1)/fPol2.GetParameter(1))**2)+((fPol2.GetParErro
r(2)/fPol2.GetParameter(2))**2))**(1/2))
        YFitErr[gpos]     = ZAfErr/ZAfSimulatedNum

        pt = TPaveText(0,0.75,0.99,1)
        pt.AddText("Simple Zenith Angle Factor Found = %.2f"%(ZAfFound))
        pt.AddText("Fit Function:
%.6f*(ZA_{f})^{2}%.6f*ZA_{f}+%.6f"%(fPol2.GetParameter(2),
fPol2.GetParameter(1), fPol2.GetParameter(0)))
```

Check_ChangingPositionError_FindZAf.py

```
      pt.AddText("Zenith Angle Factor Found by Fit = %.4f +-
%.4f"%(ZAf,ZAfErr))
      pt.Paint("NDC")
      pt.Draw()
      #colocar o valor de zero da derivada da parabola

      if (gpos == 0):
            PdfFilename = os.path.join(outFolder,
"ZAfFindCurve_PosChange.pdf(")
      elif (gpos == (len(X) - 1)):
            PdfFilename = os.path.join(outFolder,
"ZAfFindCurve_PosChange.pdf)")
      else:
            PdfFilename = os.path.join(outFolder,
"ZAfFindCurve_PosChange.pdf")
      c.Print(PdfFilename)

gStyle.SetOptStat(0)
gROOT.SetBatch(0)

# Building plots for Absolut Sum of Error Positions

c1    = TCanvas("ZAf Search Fit by Inaccurate Position - Absolut Sum",
"ZAf Search Fit by Inaccurate Position - Absolut Sum", 1280, 800)
c1.SetBottomMargin(0.2)
tprofFit = TProfile("ZA_{f} Search by Inaccurate Position - Absolut
Sum","ZA_{f} Search by Inaccurate Position - Absolut
Sum",50,0,np.max(X),0,np.max(YFit))
tprofFit.GetXaxis().SetTitle("#sum_{group=0,axis=0}^{group=2,axis=2}#l
eft| PositionDifference_{(group,axis)} #right| [cm]")
tprofFit.GetYaxis().SetTitle("(|ZAf_{Found} -
ZAf_{Simulated}|)/ZAf_{Simulated}")
for pos in range(0,len(YFit)):
      tprofFit.Fill(X[pos], YFit[pos])

tprofFit.SetMarkerSize(1)
tprofFit.SetMarkerStyle(20)
tprofFit.SetMarkerColor(2)
tprofFit.SetLineColor(2)
tprofFit.SetLineWidth(1)
tprofFit.GetXaxis().SetTitleOffset(2)
tprofFitMean = tprofFit.GetMean(2)
tprofFitStd = tprofFit.GetRMS(2)
tprofFit.Draw()

tprof = TProfile("ZA_{f} Search by Inaccurate Position - Absolut
Sum","ZA_{f} Search by Inaccurate Position - Absolut
Sum",50,0,np.max(X),0,np.max(YMin))
tprof.GetXaxis().SetTitle("#sum_{group=0,axis=0}^{group=2,axis=2}#left
| PositionDifference_{(group,axis)} #right| [cm]")
tprof.GetYaxis().SetTitle("(|ZAf_{Found} -
ZAf_{Simulated}|)/ZAf_{Simulated}")
for pos in range(0,len(YFit)):
      tprof.Fill(X[pos], YMin[pos])

tprof.SetMarkerSize(1)
tprof.SetMarkerStyle(21)
tprof.SetMarkerColor(4)
tprof.SetLineColor(4)
tprof.SetLineWidth(1)
tprof.GetXaxis().SetTitleOffset(2)
```

```
tprofMean = tprof.GetMean(2)
tprofStd = tprof.GetRMS(2)
tprof.Draw("SAME")

legend = TLegend(0.15,0.7,0.48,0.88)
legend.AddEntry(tprofFit,"Search Using Fit Method","lep")
legend.AddEntry(tprofFit, "Mean -> %.5f"%(tprofFitMean), "")
legend.AddEntry(tprofFit, "Std -> %.5f"%(tprofFitStd), "")
legend.AddEntry(tprof,"Search Using Minimum Method","lep")
legend.AddEntry(tprof, "Mean -> %.5f"%(tprofMean), "")
legend.AddEntry(tprof, "Std -> %.5f"%(tprofStd), "")
legend.Draw()

c1.Print(os.path.join(outFolder,"ZAfFindCurve_ChangingPositionStudy.pn
g"))

c2    = TCanvas("ZAf Search Fit Error by Inaccurate Position - Absolut
Sum", "ZAf Search Fit Error by Inaccurate Position - Absolut Sum",
1280, 800)
c2.SetBottomMargin(0.2)
tprofFitErr = TProfile("ZA_{f} Search Fit Error by Inaccurate Position
- Absolut Sum","ZA_{f} Search Fit Error by Inaccurate Position -
Absolut Sum",100,0,np.max(X),0,np.max(YFitErr))
tprofFitErr.GetXaxis().SetTitle("#sum_{group=0,axis=0}^{group=2,axis=2
}#left| PositionDifference_{(group,axis)} #right| [cm]")
tprofFitErr.GetYaxis().SetTitle("ZAf Fit Error/ZAf_{Simulated}")
for pos in range(0,len(YFitErr)):
      tprofFitErr.Fill(X[pos], YFitErr[pos])
tprofFitErr.SetMaximum(0.6)
tprofFitErr.SetMarkerSize(1)
tprofFitErr.SetMarkerStyle(20)
tprofFitErr.SetMarkerColor(2)
tprofFitErr.SetLineColor(2)
tprofFitErr.SetLineWidth(1)
tprofFitErr.GetXaxis().SetTitleOffset(2)
tprofFitErrMean = tprof.GetMean(2)
tprofFitErrStd = tprof.GetRMS(2)
tprofFitErr.Draw()

legend = TLegend(0.15,0.76,0.48,0.88)
legend.AddEntry(tprofFitErr,"Search Using Fit Method - Fit
Error","lep")
legend.AddEntry(tprofFitErr, "Mean -> %.5f"%(tprofFitErrMean), "")
legend.AddEntry(tprofFitErr, "Std -> %.5f"%(tprofFitErrStd), "")
legend.Draw()

c2.Print(os.path.join(outFolder,"ZAfFindCurve_ChangingPositionStudy_Fi
tError.png"))

# Building plots for Sum (Not absolut values) of Error Positions

c3    = TCanvas("ZAf Search Fit by Inaccurate Position", "ZAf Search
Fit by Inaccurate Position", 1280, 800)
c3.SetBottomMargin(0.2)
tprofFit = TProfile("ZA_{f} Search by Inaccurate Position","ZA_{f}
Search by Inaccurate
Position",50,np.min(XWAbs),np.max(XWAbs),0,np.max(YFit))
tprofFit.GetXaxis().SetTitle("#sum_{group=0,axis=0}^{group=2,axis=2}
PositionDifference_{(group,axis)} [cm]")
tprofFit.GetYaxis().SetTitle("(|ZAf_{Found} -
ZAf_{Simulated}|)/ZAf_{Simulated}")
```

Check_ChangingPositionError_FindZAf.py

```
for pos in range(0,len(YFit)):
      tprofFit.Fill(XWAbs[pos], YFit[pos])

tprofFit.SetMarkerSize(1)
tprofFit.SetMarkerStyle(20)
tprofFit.SetMarkerColor(2)
tprofFit.SetLineColor(2)
tprofFit.SetLineWidth(1)
tprofFit.GetXaxis().SetTitleOffset(2)
tprofFitMean = tprofFit.GetMean(2)
tprofFitStd = tprofFit.GetRMS(2)
tprofFit.Draw()

tprof = TProfile("ZA_{f} Search by Inaccurate Position","ZA_{f} Search
by Inaccurate Position",50,np.min(XWAbs),np.max(XWAbs),0,np.max(YMin))
tprof.GetXaxis().SetTitle("#sum_{group=0,axis=0}^{group=2,axis=2}
PositionDifference_{(group,axis)} [cm]")
tprof.GetYaxis().SetTitle("(|ZAf_{Found} -
ZAf_{Simulated}|)/ZAf_{Simulated}")
for pos in range(0,len(YFit)):
      tprof.Fill(XWAbs[pos], YMin[pos])

tprof.SetMarkerSize(1)
tprof.SetMarkerStyle(21)
tprof.SetMarkerColor(4)
tprof.SetLineColor(4)
tprof.SetLineWidth(1)
tprof.GetXaxis().SetTitleOffset(2)
tprofMean = tprof.GetMean(2)
tprofStd = tprof.GetRMS(2)
tprof.Draw("SAME")

legend = TLegend(0.32,0.7,0.6,0.88)
legend.AddEntry(tprofFit,"Search Using Fit Method","lep")
legend.AddEntry(tprofFit, "Mean -> %.5f"%(tprofFitMean), "")
legend.AddEntry(tprofFit, "Std -> %.5f"%(tprofFitStd), "")
legend.AddEntry(tprof,"Search Using Minimum Method","lep")
legend.AddEntry(tprof, "Mean -> %.5f"%(tprofMean), "")
legend.AddEntry(tprof, "Std -> %.5f"%(tprofStd), "")
legend.Draw()

c3.Print(os.path.join(outFolder,"ZAfFindCurve_ChangingPositionStudy_Wi
thoutAbsolutValues.png"))

c4    = TCanvas("ZAf Search Fit Error by Inaccurate Position", "ZAf
Search Fit Error by Inaccurate Position", 1280, 800)
c4.SetBottomMargin(0.2)
tprofFitErr = TProfile("ZA_{f} Search Fit Error by Inaccurate
Position","ZA_{f} Search Fit Error by Inaccurate
Position",100,np.min(XWAbs),np.max(XWAbs),0,np.max(YFitErr)*1.5)
tprofFitErr.GetXaxis().SetTitle("#sum_{group=0,axis=0}^{group=2,axis=2
} PositionDifference_{(group,axis)} [cm]")
tprofFitErr.GetYaxis().SetTitle("ZAf Fit Error/ZAf_{Simulated}")
for pos in range(0,len(YFitErr)):
      tprofFitErr.Fill(XWAbs[pos], YFitErr[pos])
tprofFitErr.SetMaximum(0.6)
tprofFitErr.SetMarkerSize(1)
tprofFitErr.SetMarkerStyle(20)
tprofFitErr.SetMarkerColor(2)
tprofFitErr.SetLineColor(2)
tprofFitErr.SetLineWidth(1)
```

Check_ChangingPositionError_FindZAf.py

```
tprofFitErr.GetXaxis().SetTitleOffset(2)
tprofFitErrMean = tprof.GetMean(2)
tprofFitErrStd = tprof.GetRMS(2)
tprofFitErr.Draw()

legend = TLegend(0.32,0.76,0.6,0.88)
legend.AddEntry(tprofFitErr,"Search Using Fit Method - Fit
Error","lep")
legend.AddEntry(tprofFitErr, "Mean -> %.5f"%(tprofFitErrMean), "")
legend.AddEntry(tprofFitErr, "Std -> %.5f"%(tprofFitErrStd), "")
legend.Draw()

c4.Print(os.path.join(outFolder,"ZAfFindCurve_ChangingPositionStudy_Wi
thoutAbsolutValues_FitError.png"))
```

Check_ChangingPositionError_FindZAf.py

Apêndice P

Shell Script Responsável por Iniciar a Análise Diária dos Dados do Experimento CRE4AT

Este apêndice é composto pelo código desenvolvido em *Shell Script*, para realizar a análise diária dos dados de forma automatizada.

```
#########################################################################
####
# Company:  CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename: CRE4AT_Crontab_IPANEMA_SiPM.sh
# Author:   Diogo Ayres Rocha (dayres@cbpf.br)
# Date:     15-08-2023
#########################################################################
####
# Revision History:
#   Date     Author              Rev.    Comments
# 15 Ago 23  Diogo Ayres Rocha   0001    File Created
#########################################################################
####

#!/bin/bash
source /home/cre4at_cbpf/anaconda3/bin/activate pyroot
date=`date "+%Y%m%d"`
rsync -cravzp server:~/CRE4AT_EACF_SiPM/DATA/*${date}*
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/tmpDATA/
list=$(ls /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/tmpDATA/ | grep
"${date}")
if [ -z "$list" ] ;
then
      if [ -e /home/cre4at_cbpf/.dates2syncandplot_IPANEMA_SiPM.dat ]
      then
            date=`date "+%Y%m%d"`
            echo "${date}" | tee -a
/home/cre4at_cbpf/.dates2syncandplot_IPANEMA_SiPM.dat
      else
            date=`date --date="yesterday" "+%Y%m%d"`
            echo "${date}" | tee -a
/home/cre4at_cbpf/.dates2syncandplot_IPANEMA_SiPM.dat
            date=`date "+%Y%m%d"`
            echo "${date}" | tee -a
/home/cre4at_cbpf/.dates2syncandplot_IPANEMA_SiPM.dat
      fi
else
      mkdir /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date}
      mv /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/tmpDATA/*
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date}
      cd /home/cre4at_cbpf/python_scripts/
      DateListFileExists=false
      if [ -e /home/cre4at_cbpf/.dates2syncandplot_IPANEMA_SiPM.dat ]
      then
            DateListFileExists=true
            while read line
            do
                  date=$line
                  rsync -cravzp
server:~/CRE4AT_EACF_SiPM/DATA/*${date}*
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/tmpDATA/
                  list=$(ls
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/tmpDATA/ | grep
"${date}")
                  if [[ ! -z "$list" ]] ;
                  then
                        mkdir
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date}
                        mv
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/tmpDATA/*
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date}
```

```
                        echo "python
CRE4AT_DailyAnalysis_IPANEMA_SiPM.py -i
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date} -o
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/plots_history/${date} -m
/home/cre4at_cbpf//sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_IPANEMA_MAP_2Beve
nts.dat -r /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/ROOT_Files -t
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_Overall_Data.dat"
                        python CRE4AT_DailyAnalysis_IPANEMA_SiPM.py -i
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date} -o
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/plots_history/${date} -m
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_IPANEMA_MAP_2Beven
ts.dat -r /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/ROOT_Files -t
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_Overall_Data.dat
                fi
            done <
/home/cre4at_cbpf/.dates2syncandplot_IPANEMA_SiPM.dat
            rm /home/cre4at_cbpf/.dates2syncandplot_IPANEMA_SiPM.dat
      fi
      if [ "$DateListFileExists" = false ] ;
      then
            date=`date --date="yesterday" "+%Y%m%d"`
            rsync -cravzp server:~/CRE4AT_EACF_SiPM/DATA/*${date}*
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date}
            #rm -r
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/plots_history/${date}
            echo "python CRE4AT_DailyAnalysis_IPANEMA_SiPM.py -i
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date} -o
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/plots_history/${date} -m
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_IPANEMA_MAP_2Beven
ts.dat -r /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/ROOT_Files -t
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_Overall_Data.dat"
            python CRE4AT_DailyAnalysis_IPANEMA_SiPM.py -i
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date} -o
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/plots_history/${date} -m
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_IPANEMA_MAP_2Beven
ts.dat -r /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/ROOT_Files -t
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_Overall_Data.dat
      fi
      date=`date "+%Y%m%d"`
      echo "python CRE4AT_DailyAnalysis_IPANEMA_SiPM.py -i
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date} -o
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/plots_history/${date} -m
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_IPANEMA_MAP_2Beven
ts.dat -r /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/ROOT_Files"
      python CRE4AT_DailyAnalysis_IPANEMA_SiPM.py -i
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/DATA/${date} -o
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/plots_history/${date} -m
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/CRE4AT_IPANEMA_MAP_2Beven
ts.dat -r /home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/ROOT_Files
      rsync -cravzp
/home/cre4at_cbpf/sf_CRE4AT/CRE4AT_EACF_SiPM/plots_history/*20*
cre4atsite:/var/www/wp/cre4at/coleta-de-
dados/images/past/IPANEMA_SiPM/
fi
conda deactivate
```

CRE4AT_Crontab_IPANEMA_SiPM.sh

Apêndice Q

Programa de Análise Diária dos Dados do Experimento CRE4AT (pyROOT)

Este apêndice é composto pelo código desenvolvido em *python3*, com uso do *framework* ROOT/CERN, na versão 6.26/04, que realiza a análise diária dos dados recebidos do experimento CRE4AT.

O código faz uso de algumas bibliotecas *python*, nas seguintes versões:

1. Numpy, versão 1.19.5;
2. Pandas, versão 1.1.5;
3. Datetime, versão 4.9;
4. Argparse, versão 1.4.0;
5. tqdm, versão 4.66.1.

```
#########################################################################
# Company:  CENTRO BRASILEIRO DE PESQUISAS FISICAS - BRASIL
# Filename: CRE4AT_DailyAnalysis_IPANEMA_SiPM.py
# Author:   Diogo Ayres Rocha (dayres@cbpf.br)
# Date:     15-08-2023
#########################################################################
# Revision History:
#   Date     Author             Rev.   Comments
# 15 Ago 23  Diogo Ayres Rocha  0001   File Created
# 20 Sep 23  Diogo Ayres Rocha  0002   Improvements on Burst Identif
# 15 Oct 23  Diogo Ayres Rocha  0003   Inclusion Burst Global Method
# 25 Nov 23  Diogo Ayres Rocha  0004   Plots Labels Correc.
# 15 Mar 24  Diogo Ayres Rocha  0005   Enable Forbush Identification
#########################################################################

from ROOT import TCanvas, TGraphErrors, TMultiGraph, TH1F, TH1D, TF1,
gROOT, gStyle, TLegend, TPad, gPad, TLatex, TGraph, TPaveText, TFile,
TTree
import pandas as pd
import numpy as np
from numpy import array
from array import array
import sys, os
import argparse
from datetime import datetime
from tqdm import tqdm

#colors = [1, 2, 9, 4, 41, 6, 7, 39]
PIDIndex                = ['PID_Sensor1', 'PID_Sensor2',
'PID_Avg','PID_Flag']
PIDUnits                = ['#circC', '#circC', '#circC', 'Flag Value']
SlowIndex               = ['AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY',
'GyroZ', 'MagX', 'MagY', 'MagZ', 'Temp', 'Hum', 'Press', 'HV_Temp',
'HV_Voltage']
SlowUnits               = ['m/s^{2}', 'm/s^{2}','m/s^{2}', 'rad/s',
'rad/s', 'rad/s', '#muT', '#muT', '#muT', '#circC', '%', 'hPa',
'#circC', 'Voltage [V]']
SlowControlTitles = ['Accelerometer X Axis', 'Accelerometer Y Axis',
'Accelerometer Z Axis', 'Gyroscope X Axis', 'Gyroscope Y Axis',
'Gyroscope Z Axis', 'Magnetometer X Axis', 'Magnetometer Y Axis',
'Magnetometer Z Axis', 'Temperature', 'Humidity', 'Pressure', 'HV
Temperature', 'HV Voltage']
Counters2Plot           = ['ACh0', 'BCh0', 'CCh0', 'ACh1', 'BCh1',
'CCh1', 'ACh2', 'BCh2', 'CCh2', 'ACh3', 'BCh3', 'CCh3', 'A0&1',
'A0&2', 'A0&3', 'A1&2', 'A1&3', 'A2&3', 'B0&1', 'B0&2', 'B0&3',
'B1&2', 'B1&3', 'B2&3', 'C0&1', 'C0&2', 'C0&3', 'C1&2', 'C1&3',
'C2&3', 'A0&1&2', 'A0&1&3', 'A0&2&3', 'A1&2&3', 'A0&1&2&3', 'B0&1&2',
'B0&1&3', 'B0&2&3', 'B1&2&3', 'B0&1&2&3', 'C0&1&2', 'C0&1&3',
'C0&2&3', 'C1&2&3', 'C0&1&2&3']
R_Columns               = ["R_A01B01", "R_A01B12", "R_A01B23",
"R_A12B01", "R_A12B12", "R_A12B23", "R_A23B01", "R_A23B12",
"R_A23B23", "R_A01C01", "R_A01C12", "R_A01C23", "R_A12C01",
"R_A12C12", "R_A12C23", "R_A23C01", "R_A23C12", "R_A23C23",
"R_B01C01", "R_B01C12", "R_B01C23", "R_B12C01", "R_B12C12",
"R_B12C23", "R_B23C01", "R_B23C12" ,"R_B23C23"]
GroupsList              = ["A", "B", "C"]
Location                = "IPANEMA_SiPM"
AcqTime                       = 10 # Seconds
NumOfChsInGroups  = 4
DetArea                       = 0.1 * 0.69
InitialLag              = 5
```

```
threshold_burst         = 5.
threshold_forbush       = 5.
nbinFix                 = 200
limitSearchLoop         = 50
ZAFSearchPoints2Add     = 80

# Flags Functions (Burst and Forbursh)
def flag_burst_right(y, lag, threshold):
    yv = y[0:lag]
    flag_burst = np.zeros(len(y))
    flag_burst[0:lag] = np.ones(lag)
    avgFilter = [0]*(lag-1)
    stdFilter = [0]*(lag-1)
    avgFilter.append(np.mean(yv))
    stdFilter.append(np.std(yv))
    for i in tqdm(range(lag, len(y))):
        if y[i] == 0 and y[i+1] == 0 and y[i+2] == 0:
            avgFilter.append(avgFilter[i-1])
            stdFilter.append(stdFilter[i-1])

        elif (y[i] - avgFilter[i-1]) > (threshold * stdFilter[i-1]):
            if y[i] > avgFilter[i-1]:
                flag_burst[i] = 1
            avgFilter.append(avgFilter[i-1])
            stdFilter.append(stdFilter[i-1])
        else:
            yv[0:lag-1] = yv[1:lag]
            yv[lag-1] = y[i]
            avgFilter.append(np.mean(yv))
            stdFilter.append(np.std(yv))

    return dict(flag_burst = np.asarray(flag_burst), avgFilter =
np.asarray(avgFilter), stdFilter = np.asarray(stdFilter))

def flag_burst_left(y, lag, threshold):
    yv = y[len(y)-lag:len(y)]
    flag_burst = np.zeros(len(y))
    flag_burst[len(y)-lag:len(y)] = np.ones(lag)
    avgFilter = [0]*len(y)
    stdFilter = [0]*len(y)
    avgFilter[len(y)-lag+1] = np.mean(yv)
    stdFilter[len(y)-lag+1] = np.std(yv)
    for i in tqdm(range(len(y)-lag, 0, -1)):
        if y[i] == 0 and y[i-1] == 0 and y[i-2] == 0:
            avgFilter[i] = avgFilter[i+1]
            stdFilter[i] = stdFilter[i+1]
        elif (y[i] - avgFilter[i+1]) > (threshold * stdFilter[i+1]):
            if y[i] > avgFilter[i+1]:
                flag_burst[i] = 1
            avgFilter[i] = avgFilter[i+1]
            stdFilter[i] = stdFilter[i+1]
        else:
            yv[1:lag] = yv[0:lag-1]
            yv[0] = y[i]
            avgFilter[i] = np.mean(yv)
            stdFilter[i] = np.std(yv)

    return dict(flag_burst = np.asarray(flag_burst), avgFilter =
np.asarray(avgFilter), stdFilter = np.asarray(stdFilter))

def flag_forbush_right(y, lag, threshold):
```

```
    yv = y[0:lag]
    flag_forbush = np.zeros(len(y))
    flag_forbush[0:lag] = np.ones(lag)
    avgFilter = [0]*(lag-1)
    stdFilter = [0]*(lag-1)
    avgFilter.append(np.mean(yv))
    stdFilter.append(np.std(yv))
    for i in tqdm(range(lag, len(y))):
        if (y[i] - avgFilter[i-1]) < (-threshold * stdFilter[i-1]):
            if y[i] < avgFilter[i-1]:
                flag_forbush[i] = 1
            avgFilter.append(avgFilter[i-1])
            stdFilter.append(stdFilter[i-1])
        elif (y[i] - avgFilter[i-1]) > (threshold * stdFilter[i-1]):
            avgFilter.append(avgFilter[i-1])
            stdFilter.append(stdFilter[i-1])
        else:
            yv[0:lag-1] = yv[1:lag]
            yv[lag-1] = y[i]
            avgFilter.append(np.mean(yv))
            stdFilter.append(np.std(yv))

    return dict(flag_forbush = np.asarray(flag_forbush), avgFilter =
np.asarray(avgFilter), stdFilter = np.asarray(stdFilter))

def flag_forbush_left(y, lag, threshold):
    yv = y[len(y)-lag:len(y)]
    flag_forbush = np.zeros(len(y))
    flag_forbush[len(y)-lag:len(y)] = np.ones(lag)
    avgFilter = [0]*len(y)
    stdFilter = [0]*len(y)
    avgFilter[len(y)-lag+1] = np.mean(yv)
    stdFilter[len(y)-lag+1] = np.std(yv)
    for i in tqdm(range(len(y)-lag, 0, -1)):
        if (y[i] - avgFilter[i+1]) < (-threshold * stdFilter[i+1]):
            if y[i] < (avgFilter[i+1]):
                flag_forbush[i] = 1
            avgFilter[i] = avgFilter[i+1]
            stdFilter[i] = stdFilter[i+1]
        elif (y[i] - avgFilter[i+1]) > (threshold * stdFilter[i+1]):
            avgFilter[i] = avgFilter[i+1]
            stdFilter[i] = stdFilter[i+1]
        else:
            yv[1:lag] = yv[0:lag-1]
            yv[0] = y[i]
            avgFilter[i] = np.mean(yv)
            stdFilter[i] = np.std(yv)

    return dict(flag_forbush = np.asarray(flag_forbush), avgFilter =
np.asarray(avgFilter), stdFilter = np.asarray(stdFilter))

# Parameters parse
parser=argparse.ArgumentParser(
    description='Help for the offline data Analysis of CRE4T/CBPF',
    epilog="#-------------------------------------------------------
----------#")
parser.add_argument('--InputFolder','-i', type=str, default='./',
help='Select the input data folder with ".dat" files!')
parser.add_argument('--OutFolder','-o', type=str, default="./",
help='Output folder for the result images [As default ./]')
```

```
parser.add_argument('--MapFile','-m', type=str, help='Select the input
file with NFactor Map data!')
parser.add_argument('--TextAppendDailyFile','-t', type=str,
default='', help='Select the .dat input file to append daily
parameters to overall plots!')
parser.add_argument('--ROOTFolderOutput','-r', type=str, help='Select
the .root output folder')
args=parser.parse_args()

gROOT.SetBatch(1)

SaveOverallValues = True
if (args.TextAppendDailyFile == ''):
      SaveOverallValues = False
Data_temp=[]
filesList=[]
files = os.listdir(args.InputFolder)
files.sort()
for file in files:
    if "CRE4AT_%s"%(Location) in file:
        filesList.append(os.path.join(args.InputFolder,file))
for FileNum in range(0,len(filesList)):
    print('INFO: Appending Scale file %s...'%(filesList[FileNum]))
    Data_temp.append(pd.read_csv(filesList[FileNum], delimiter="\t"))
Data=pd.concat(Data_temp)
Data = Data[Data["LED_CALIB_Status"] == 0]
Data.reset_index(drop=True, inplace=True)
del(Data_temp)
del(filesList)

PIDAcquisition = ("PID_Flag" in Data.columns.values)
if(not(os.path.isdir(args.OutFolder))):
      os.mkdir(args.OutFolder)
if(not(os.path.isdir(os.path.join(args.OutFolder, "Counts")))):
      os.mkdir(os.path.join(args.OutFolder, "Counts"))
if(not(os.path.isdir(os.path.join(args.OutFolder,
"Efficiency_Raw")))):
      os.mkdir(os.path.join(args.OutFolder, "Efficiency_Raw"))
if(not(os.path.isdir(os.path.join(args.OutFolder, "Flux_Raw")))):
      os.mkdir(os.path.join(args.OutFolder, "Flux_Raw"))
if(not(os.path.isdir(os.path.join(args.OutFolder,
"Flux_Corrected")))):
      os.mkdir(os.path.join(args.OutFolder, "Flux_Corrected"))
if(not(os.path.isdir(os.path.join(args.OutFolder, "Burst-
Forbush_Search")))):
      os.mkdir(os.path.join(args.OutFolder, "Burst-Forbush_Search"))
if(not(os.path.isdir(os.path.join(args.OutFolder, "Slow_Control")))):
      os.mkdir(os.path.join(args.OutFolder, "Slow_Control"))
if(not(os.path.isdir(os.path.join(args.OutFolder,
"Zenith_Angle_Factor_Search")))):
      os.mkdir(os.path.join(args.OutFolder,
"Zenith_Angle_Factor_Search"))
if(not(os.path.isdir(os.path.join(args.OutFolder, "PID_Monitoring")))
and PIDAcquisition):
      os.mkdir(os.path.join(args.OutFolder, "PID_Monitoring"))

text = TPaveText(0.95,.02,0.99,0.99,"blNDC")
text.SetTextColor(17)
text.SetBorderSize(0)
text.SetFillColor(19)
text.SetFillStyle(0)
```

```
text.AddText("CRE4AT/CBPF - %s / EACF - %s"%(Location,
datetime.utcfromtimestamp(Data.Timestamp[0]).strftime('%d/%m/%Y')))
text.SetAllWith("","angle",90)
text.SetTextSize(.04)

Data["AcqTime"]   = np.ones(len(Data))*AcqTime

ErrX = np.zeros(Data.Timestamp.index.stop)

StartDateTime = min(Data.Timestamp)
StopDateTime  = max(Data.Timestamp)

# Raw Data Plotting - Counters
for counter in Counters2Plot:
    if ((Data['%s'%(counter)] == 0).all() == False):
        p = TGraphErrors(Data.Timestamp.index.stop,
np.asarray((Data.Timestamp).values).astype(np.float64),
np.asarray(Data['%s'%(counter)].values).astype(np.float64),
ErrX.astype(np.float64),
np.asarray((Data['%s'%(counter)]**(1/2)).values).astype(np.float64))
        p.SetMarkerStyle(21)
        p.SetMarkerSize(0.9)
        p.SetMarkerColor(2)
        p.SetTitle('TimePlot')
        p.GetXaxis().SetTitle('Time')
        p.GetYaxis().SetTitle('Count/10s')
        p.GetXaxis().SetLabelSize(0.05)
        p.GetXaxis().SetTitleSize(0.05)
        p.GetYaxis().SetLabelSize(0.05)
        p.GetYaxis().SetTitleSize(0.05)
        p.GetXaxis().SetTimeDisplay(1)
        if ((Data.Timestamp[Data.Timestamp.index.stop-1]-
Data.Timestamp[0]) <= 86400):
            p.GetXaxis().SetTimeFormat("%H:%M")
        else:
            p.GetXaxis().SetLabelOffset(0.03)
            p.GetXaxis().SetTitleOffset(1.5)
            p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
            p.GetXaxis().SetNdivisions(-510, 0)
        p.GetXaxis().SetTimeOffset(0,"gmt")
        p.SetMinimum(0)
        nbins = int(max(Data['%s'%(counter)].values)*1.005) -
int(min(Data['%s'%(counter)].values)*0.995)
        h = TH1F("Counts %s"%(counter), "Counts %s"%(counter), nbins,
int(min(Data['%s'%(counter)].values)*0.995),
int(max(Data['%s'%(counter)].values)*1.005))
        for value in Data['%s'%(counter)].values:
            h.Fill(value)
        h.GetXaxis().SetTitle('Counts/10s')
        h.GetYaxis().SetTitle('Events')
        h.GetXaxis().SetLabelSize(0.05)
        h.GetXaxis().SetTitleSize(0.05)
        h.GetYaxis().SetLabelSize(0.05)
        h.GetYaxis().SetTitleSize(0.05)

        c = TCanvas("Counts - %s"%(counter),"Counts -
%s"%(counter),1280,980)
        t = TPad("pad1","",0.02,0.02,0.95,0.98)
        t.Draw()
        text.Draw()
        t.Divide(1,2)
```

```
            t.cd(1)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            h.Draw()
            t.cd(2)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            p.Draw("AP")

            # Saving in PNG
            imageName = "Counts_%s.png"%(counter)
            c.Print(os.path.join(args.OutFolder,"Counts",imageName))

'''def GetErrEff(num, den):
      errNum      = num**(1/2)
      errDen      = den**(1/2)
      eff         = num/den
      ret         = eff*((errNum/num)**2+(errDen/den)**2)**(1/2)
      return ret'''


Eff   = []
for group in range(0,len(GroupsList)):
      Eff.append([])
      for ch in range(0,NumOfChsInGroups):
            Eff[group].append([])

# Raw Data Plotting - Efficiencies and Flux
for groupNum, group in enumerate(GroupsList):
    for ch in range(0,NumOfChsInGroups):
        ListOfChannelsNotWanted     =
list(set(list(range(0,NumOfChsInGroups)))-set([ch]))
        if (((Data['%s0&1&2&3'%(group)] == 0).all() == False) and
((Data['%s%d&%d&%d'%(group,ListOfChannelsNotWanted[0],ListOfChannelsNo
tWanted[1],ListOfChannelsNotWanted[2])] == 0).all() == False)):
            Eff[groupNum][ch]                                =
(np.asarray(Data['%s0&1&2&3'%(group)]).astype('float')/np.asarray(Data
['%s%d&%d&%d'%(group,ListOfChannelsNotWanted[0],ListOfChannelsNotWante
d[1],ListOfChannelsNotWanted[2])]).astype('float'))
            Eff[groupNum][ch]                                =
np.nan_to_num(Eff[groupNum][ch], nan=-1, posinf=-1, neginf=-1)
            Data["Eff_Raw_%cCh%d"%(group, ch)]       =
Eff[groupNum][ch]
            #ErrEff                                  =
GetErrEff(np.asarray(Data['%s0&1&2&3'%(group)]).astype('float'),
np.asarray(Data['%s%d&%d&%d'%(group,ListOfChannelsNotWanted[0],ListOfC
hannelsNotWanted[1],ListOfChannelsNotWanted[2])]).astype('float'))
            #ErrEff = ErrX
            c     = TCanvas('Eff Raw %sCh%s'%(group,ch),'Eff Raw
%sCh%s'%(group,ch),1280,980)
            p     = TGraph(len(Eff[groupNum][ch]),
np.asarray(Data.Timestamp.values).astype('float'), Eff[groupNum][ch])
            p.SetTitle("TimePlot")
            p.GetXaxis().SetTitle('Time')
            p.GetYaxis().SetTitle('Eff')
            p.SetMarkerStyle(21)
```

```
            p.SetMarkerSize(0.9)
            p.SetMarkerColor(2)
            p.GetXaxis().SetLabelSize(0.05)
            p.GetXaxis().SetTitleSize(0.05)
            p.GetYaxis().SetLabelSize(0.05)
            p.GetYaxis().SetTitleSize(0.05)
            p.GetXaxis().SetTimeDisplay(1)
            if ((Data.Timestamp[Data.Timestamp.index.stop-1]-
Data.Timestamp[0]) <= 86400):
                p.GetXaxis().SetTimeFormat("%H:%M")
            else:
                p.GetXaxis().SetLabelOffset(0.03)
                p.GetXaxis().SetTitleOffset(1.5)

p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
                p.GetXaxis().SetNdivisions(-510, 0)
            p.GetXaxis().SetTimeOffset(0,"gmt")
            h = TH1F('Efficiency Raw %sCh%s'%(group, ch), 'Efficiency
Raw %sCh%s'%(group, ch), nbinFix, min(Eff[groupNum][ch])*0.995,
max(Eff[groupNum][ch])*1.005)
            for value in Eff[groupNum][ch]:
                h.Fill(value)
            h.GetXaxis().SetTitle('Eff')
            h.GetYaxis().SetTitle('Events')
            h.GetXaxis().SetLabelSize(0.05)
            h.GetXaxis().SetTitleSize(0.05)
            h.GetYaxis().SetLabelSize(0.05)
            h.GetYaxis().SetTitleSize(0.05)

            t = TPad("pad1","",0.02,0.02,0.95,0.98)
            t.Draw()
            text.Draw()
            t.Divide(1,2)

            t.cd(1)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            h.Draw()
            t.cd(2)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            p.Draw("AP")

            # Saving in PNG
            imageName = "Eff_Raw_%cCh%d.png"%(group, ch)

c.Print(os.path.join(args.OutFolder,"Efficiency_Raw",imageName))
        else:
            Eff[groupNum][ch] = np.zeros(len(Data))
    if ((((Eff[groupNum][0] == 0).all()) == False) and
(((Eff[groupNum][1] == 0).all()) == False) and (((Eff[groupNum][2] ==
0).all()) == False) and (((Eff[groupNum][3] == 0).all()) == False)):
        Flux                    =
(np.asarray(Data['%s0&1&2&3'%(group)]).astype('float')/(DetArea*AcqTim
e*Eff[groupNum][0]*Eff[groupNum][1]*Eff[groupNum][2]*Eff[groupNum][3])
)
```

```
        Flux                    = np.nan_to_num(Flux, nan=-1, posinf=-1,
neginf=-1)
        Data["Flux_Raw_%c"%(group)] = Flux

        c = TCanvas('Flux Raw %c'%(group),'Flux Raw
%c'%(group),1280,980)
        p = TGraph(len(Flux),
np.asarray(Data.Timestamp.values).astype('float'),
np.asarray(Data["Flux_Raw_%c"%(group)]).astype(float))
        p.SetTitle("TimePlot")
        p.GetXaxis().SetTitle('Time')
        p.GetYaxis().SetTitle('Flux')
        p.SetMarkerStyle(21)
        p.SetMarkerSize(0.9)
        p.SetMarkerColor(2)
        p.GetXaxis().SetLabelSize(0.05)
        p.GetXaxis().SetTitleSize(0.05)
        p.GetYaxis().SetLabelSize(0.05)
        p.GetYaxis().SetTitleSize(0.05)
        p.GetXaxis().SetTimeDisplay(1)
        if ((Data.Timestamp[Data.Timestamp.index.stop-1]-
Data.Timestamp[0]) <= 86400):
            p.GetXaxis().SetTimeFormat("%H:%M")
        else:
            p.GetXaxis().SetLabelOffset(0.03)
            p.GetXaxis().SetTitleOffset(1.5)
            p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
            p.GetXaxis().SetNdivisions(-510, 0)
        p.GetXaxis().SetTimeOffset(0,"gmt")
        p.SetMinimum(0)

        h = TH1F('Flux Raw %c'%(group), 'Flux Raw %c'%(group),
nbinFix, min(Data["Flux_Raw_%c"%(group)])*0.995,
max(Data["Flux_Raw_%c"%(group)])*1.005)
        for value in Data["Flux_Raw_%c"%(group)].values:
            h.Fill(value)
        h.GetXaxis().SetTitle('Flux')
        h.GetYaxis().SetTitle('Events')
        h.GetXaxis().SetLabelSize(0.05)
        h.GetXaxis().SetTitleSize(0.05)
        h.GetYaxis().SetLabelSize(0.05)
        h.GetYaxis().SetTitleSize(0.05)

        t = TPad("pad1","",0.02,0.02,0.95,0.98)
        t.Draw()
        text.Draw()
        t.Divide(1,2)

        t.cd(1)
        gPad.SetRightMargin(0.12)
        gPad.SetLeftMargin(0.12)
        gPad.SetTopMargin(0.12)
        gPad.SetBottomMargin(0.18)
        h.Draw()
        t.cd(2)
        gPad.SetRightMargin(0.12)
        gPad.SetLeftMargin(0.12)
        gPad.SetTopMargin(0.12)
        gPad.SetBottomMargin(0.18)
        p.Draw("AP")
```

```
        # Saving in PNG
        imageName = "Flux_Raw_Group%c.png"%(group)
        c.Print(os.path.join(args.OutFolder,"Flux_Raw",imageName))


# Raw Data Plotting - Slow Control
Data["Err_AccX"] = (ErrX+0.0006)
Data["Err_AccY"] = (ErrX+0.0006)
Data["Err_AccZ"] = (ErrX+0.0006)
Data["Err_GyroX"] = (ErrX) # +0.15
Data["Err_GyroY"] = (ErrX) # +0.15
Data["Err_GyroZ"] = (ErrX) # +0.15
Data["Err_MagX"] = (ErrX+0.014)
Data["Err_MagY"] = (ErrX+0.014)
Data["Err_MagZ"] = (ErrX+0.014)
Data["Err_Temp"] = (ErrX+1)
Data["Err_Hum"] = (Data.Hum*0.03)
Data["Err_Press"] = (ErrX+1)
Data["Err_HV_Temp"] = (ErrX)
Data["Err_HV_Voltage"] = (ErrX)

for pos, SlowName in enumerate(SlowIndex):
      if ((Data[SlowName].isnull() | (Data[SlowName] == -10000)).all()
== False):
            NullDataFound     = False
            if (Data[SlowName].isnull().values.any()):
                  NullDataFound     = True
                  df_temp                 =
Data.dropna(subset=SlowName)
                  df_temp.reset_index(drop=True, inplace=True)
                  ErrX_tmp          =
np.zeros(df_temp.Timestamp.index.stop)
                  p                       =
TGraphErrors(df_temp.Timestamp.index.stop,
np.asarray((df_temp.Timestamp).values).astype(np.float64),
np.asarray(df_temp[SlowName].values).astype(np.float64),
ErrX_tmp.astype(np.float64),
np.asarray(df_temp["Err_%s"%(SlowName)]).astype(np.float64))
            else:
                  p                 =
TGraphErrors(Data.Timestamp.index.stop,
np.asarray((Data.Timestamp).values).astype(np.float64),
np.asarray(Data[SlowName].values).astype(np.float64),
ErrX.astype(np.float64),
np.asarray(Data["Err_%s"%(SlowName)]).astype(np.float64))
            p.SetMarkerStyle(21)
            p.SetMarkerSize(0.9)
            p.SetMarkerColor(2)
            p.SetTitle('TimePlot')
            p.GetXaxis().SetTitle('Time')
            p.GetYaxis().SetTitle(SlowUnits[pos])
            p.GetXaxis().SetLabelSize(0.05)
            p.GetXaxis().SetTitleSize(0.05)
            p.GetYaxis().SetLabelSize(0.05)
            p.GetYaxis().SetTitleSize(0.05)
            p.GetXaxis().SetTimeDisplay(1)
            if ((Data.Timestamp[Data.Timestamp.index.stop-1]-
Data.Timestamp[0]) <= 86400):
                  p.GetXaxis().SetTimeFormat("%H:%M")
            else:
                  p.GetXaxis().SetLabelOffset(0.03)
```

```
                  p.GetXaxis().SetTitleOffset(1.5)

      p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
                  p.GetXaxis().SetNdivisions(-506, 0)
            p.GetXaxis().SetTimeOffset(0,"gmt")
            #print('Plotting SlowControl %s Histogram' %
(SlowIndex[i]))
            if (NullDataFound):
                  h     = TH1F(SlowControlTitles[pos],
SlowControlTitles[pos], 130 if (max(df_temp[SlowName].values)-
min(df_temp[SlowName].values)) > 30 else 50,
min(df_temp[SlowName].values)*0.995,
max(df_temp[SlowName].values)*1.005)
                  for value in df_temp[SlowName].values:
                        h.Fill(value)
            else:
                  h     = TH1F(SlowControlTitles[pos],
SlowControlTitles[pos], 130 if (max(Data[SlowName].values)-
min(Data[SlowName].values)) > 30 else 50,
min(Data[SlowName].values)*0.995, max(Data[SlowName].values)*1.005)
                  for value in Data[SlowName].values:
                        h.Fill(value)
            h.GetXaxis().SetTitle(SlowUnits[pos])
            h.GetYaxis().SetTitle('Events')
            h.GetXaxis().SetLabelSize(0.05)
            h.GetXaxis().SetTitleSize(0.05)
            h.GetYaxis().SetLabelSize(0.05)
            h.GetYaxis().SetTitleSize(0.05)

            c =
TCanvas(SlowControlTitles[pos],SlowControlTitles[pos],1280,980)
            t = TPad("pad1","",0.02,0.02,0.95,0.98)
            t.Draw()
            text.Draw()
            t.Divide(1,2)

            t.cd(1)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.15)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            h.Draw()

            t.cd(2)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.15)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            p.Draw("AP")


            # Saving in PNG
            imageName = "Slow_%s.png"%(SlowName)

      c.Print(os.path.join(args.OutFolder,"Slow_Control",imageName))

if(PIDAcquisition):
      for pos in range(0, len(PIDIndex)-1):
            PIDName = PIDIndex[pos]
            if (Data[PIDName].isnull().all() == False):
                  NullDataFound     = False
```

```
                if (Data[PIDName].isnull().values.any()):
                        NullDataFound     = True
                        df_temp                 =
Data.dropna(subset=PIDName)
                        df_temp.reset_index(drop=True, inplace=True)
                        ErrX_tmp          =
np.zeros(df_temp.Timestamp.index.stop)
                        p                       =
TGraphErrors(df_temp.Timestamp.index.stop,
np.asarray((df_temp.Timestamp).values).astype(np.float64),
np.asarray(df_temp[PIDName].values).astype(np.float64),
ErrX_tmp.astype(np.float64),
np.asarray(ErrX_tmp+0.5).astype(np.float64))
                else:
                        p                 =
TGraphErrors(Data.Timestamp.index.stop,
np.asarray((Data.Timestamp).values).astype(np.float64),
np.asarray(Data[PIDName].values).astype(np.float64),
ErrX.astype(np.float64), np.asarray(ErrX+0.5).astype(np.float64))
                p.SetMarkerStyle(21)
                p.SetMarkerSize(0.9)
                p.SetMarkerColor(2)
                p.SetTitle('TimePlot')
                p.GetXaxis().SetTitle('Time')
                p.GetYaxis().SetTitle(PIDUnits[pos])
                p.GetXaxis().SetLabelSize(0.05)
                p.GetXaxis().SetTitleSize(0.05)
                p.GetYaxis().SetLabelSize(0.05)
                p.GetYaxis().SetTitleSize(0.05)
                p.GetXaxis().SetTimeDisplay(1)
                if ((Data.Timestamp[Data.Timestamp.index.stop-1]-
Data.Timestamp[0]) <= 86400):
                        p.GetXaxis().SetTimeFormat("%H:%M")
                else:
                        p.GetXaxis().SetLabelOffset(0.03)
                        p.GetXaxis().SetTitleOffset(1.5)

      p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
                        p.GetXaxis().SetNdivisions(-506, 0)
                p.GetXaxis().SetTimeOffset(0,"gmt")
                #print('Plotting SlowControl %s Histogram' %
(SlowIndex[i]))
                if (NullDataFound):
                        h     = TH1F(PIDName.replace("_", " "),
PIDName.replace("_", " "), 130 if (max(df_temp[PIDName].values)-
min(df_temp[PIDName].values)) > 30 else 50,
min(df_temp[PIDName].values)*0.995,
max(df_temp[PIDName].values)*1.005)
                        for value in df_temp[PIDName].values:
                                h.Fill(value)
                else:
                        h     = TH1F(PIDName.replace("_", " "),
PIDName.replace("_", " "), 130 if (max(Data[PIDName].values)-
min(Data[PIDName].values)) > 30 else 50,
min(Data[PIDName].values)*0.995, max(Data[PIDName].values)*1.005)
                        for value in Data[PIDName].values:
                                h.Fill(value)
                h.GetXaxis().SetTitle(PIDUnits[pos])
                h.GetYaxis().SetTitle('Events')
                h.GetXaxis().SetLabelSize(0.05)
                h.GetXaxis().SetTitleSize(0.05)
```

```
                    h.GetYaxis().SetLabelSize(0.05)
                    h.GetYaxis().SetTitleSize(0.05)

                    c = TCanvas(PIDName.replace("_", "
"),PIDName.replace("_", " "),1280,980)
                    t = TPad("pad1","",0.02,0.02,0.95,0.98)
                    t.Draw()
                    text.Draw()
                    t.Divide(1,2)

                    t.cd(1)
                    gPad.SetRightMargin(0.12)
                    gPad.SetLeftMargin(0.15)
                    gPad.SetTopMargin(0.12)
                    gPad.SetBottomMargin(0.18)
                    h.Draw()

                    t.cd(2)
                    gPad.SetRightMargin(0.12)
                    gPad.SetLeftMargin(0.15)
                    gPad.SetTopMargin(0.12)
                    gPad.SetBottomMargin(0.12)
                    p.Draw("AP")


                    # Saving in PNG
                    imageName = "%s.png"%(PIDName)

      c.Print(os.path.join(args.OutFolder,"PID_Monitoring",imageName))
      PIDName = PIDIndex[-1]
      pos         = len(PIDIndex) - 1
      if (Data[PIDName].isnull().all() == False):
            NullDataFound     = False
            if (Data[PIDName].isnull().values.any()):
                  NullDataFound     = True
                  df_temp                 = Data.dropna(subset=PIDName)
                  df_temp.reset_index(drop=True, inplace=True)
                  p                       =
TGraph(df_temp.Timestamp.index.stop,
np.asarray((df_temp.Timestamp).values).astype(np.float64),
np.asarray(df_temp[PIDName].values).astype(np.float64))
            else:
                  p                       =
TGraph(Data.Timestamp.index.stop,
np.asarray((Data.Timestamp).values).astype(np.float64),
np.asarray(Data[PIDName].values).astype(np.float64))
            p.SetMarkerStyle(21)
            p.SetMarkerSize(0.9)
            p.SetMarkerColor(2)
            p.SetTitle(PIDName.replace("_", " "))
            p.GetXaxis().SetTitle('Time')
            p.GetYaxis().SetTitle(PIDUnits[pos])
            p.GetXaxis().SetLabelSize(0.05)
            p.GetXaxis().SetTitleSize(0.05)
            p.GetYaxis().SetLabelSize(0.05)
            p.GetYaxis().SetTitleSize(0.05)
            p.GetXaxis().SetTimeDisplay(1)
            if ((Data.Timestamp[Data.Timestamp.index.stop-1]-
Data.Timestamp[0]) <= 86400):
                  p.GetXaxis().SetTimeFormat("%H:%M")
            else:
```

```
                  p.GetXaxis().SetLabelOffset(0.03)
                  p.GetXaxis().SetTitleOffset(1.5)

      p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
                  p.GetXaxis().SetNdivisions(-506, 0)
            p.GetXaxis().SetTimeOffset(0,"gmt")
            #print('Plotting SlowControl %s Histogram' %
(SlowIndex[i]))

            c = TCanvas(PIDName.replace("_", " "),PIDName.replace("_",
" "),1280,600)
            t = TPad("pad1","",0.02,0.02,0.95,0.98)
            t.Draw()
            text.Draw()
            t.Divide(1,1)

            t.cd(1)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.15)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.12)
            p.Draw("AP")


            # Saving in PNG
            imageName = "%s.png"%(PIDName)

      c.Print(os.path.join(args.OutFolder,"PID_Monitoring",imageName))

# Burst/Forbush Identification - Local Method

for groupNum, group in enumerate(GroupsList):
      for lag in range(InitialLag, limitSearchLoop+1):
            burst_flag_R =
flag_burst_right(Data["Flux_Raw_%c"%(group)].values, lag,
threshold_burst)
            burstPositionsRight =
np.where(burst_flag_R["flag_burst"][lag:-lag]==1)
            burst_flag_L =
flag_burst_left(Data["Flux_Raw_%c"%(group)].values, lag,
threshold_burst)
            burstPositionsLeft  =
np.where(burst_flag_L["flag_burst"][lag:-lag]==1)
            '''print("Burst Position found Right: ")
            print(burstPositionsRight[0])
            print("Burst Position found Left: ")
            print(burstPositionsLeft[0])
            print('Lag Used: %d'%(lag))'''
            if (np.array_equal(burstPositionsLeft[0],
burstPositionsRight[0])):
                  break
      if (lag == limitSearchLoop):
            lag = int(limitSearchLoop/2)
            burst_flag_R =
flag_burst_right(Data["Flux_Raw_%c"%(group)].values, lag,
threshold_burst)
            burst_flag_L =
flag_burst_left(Data["Flux_Raw_%c"%(group)].values, lag,
threshold_burst)
            burst_flag_F = np.logical_and(burst_flag_R['flag_burst'],
burst_flag_L['flag_burst'])
```

```
            Data['Det_%s_Burst_Counter'%(group)] = burst_flag_F * 1
      else:
            burst_flag_F =
np.logical_and(burst_flag_R['flag_burst'],burst_flag_L['flag_burst'])
            Data['Det_%s_Burst_Counter'%(group)] = burst_flag_F * 1

      for lag in range(InitialLag, limitSearchLoop+1):
            forbush_flag_R =
flag_forbush_right(Data["Flux_Raw_%c"%(group)].values, lag,
threshold_forbush)
            forbushPositionsRight =
np.where(forbush_flag_R["flag_forbush"][lag:-lag]==1)
            forbush_flag_L =
flag_forbush_left(Data["Flux_Raw_%c"%(group)].values, lag,
threshold_forbush)
            forbushPositionsLeft =
np.where(forbush_flag_L["flag_forbush"][lag:-lag]==1)
            if (np.array_equal(forbushPositionsLeft[0],
forbushPositionsRight[0])):
                  break
      if (lag == limitSearchLoop):
            lag = int(limitSearchLoop/2)
            forbush_flag_R =
flag_forbush_right(Data["Flux_Raw_%c"%(group)].values, lag,
threshold_forbush)
            forbush_flag_L =
flag_forbush_left(Data["Flux_Raw_%c"%(group)].values, lag,
threshold_forbush)
            forbush_flag_F =
np.logical_and(forbush_flag_R['flag_forbush'],forbush_flag_L['flag_for
bush'])
            Data['Det_%s_Forbush_Counter'%(group)] = forbush_flag_F * 1
      else:
            forbush_flag_F =
np.logical_and(forbush_flag_R['flag_forbush'],forbush_flag_L['flag_for
bush'])
            Data['Det_%s_Forbush_Counter'%(group)] = forbush_flag_F * 1

      if ((Data["Det_%c_Burst_Counter"%(group)] == 1).any()):
            tmpData = Data[Data["Det_%c_Burst_Counter"%(group)] == 0]
            tmpData.reset_index(drop=True, inplace=True)
            # Protection for channel with all zero efficiency (Dead
Channel)
            if ((((tmpData["Eff_Raw_%cCh0"%(group)] == 0).all()) ==
False) and (((tmpData["Eff_Raw_%cCh1"%(group)] == 0).all()) == False)
and (((tmpData["Eff_Raw_%cCh2"%(group)] == 0).all()) == False) and
(((tmpData["Eff_Raw_%cCh3"%(group)] == 0).all()) == False)):
                  c           = TCanvas('Flux Raw %c'%(group),'Flux Raw
%c'%(group),1280,1280)
                  p           =
TGraph(len(tmpData["Flux_Raw_%c"%(group)]),
np.asarray(tmpData.Timestamp.values).astype('float'),
np.asarray(tmpData["Flux_Raw_%c"%(group)].values).astype('float'))
                  p.SetTitle('Flux Raw Group %c - Without Burst - Local
Method'%(group))
                  p.GetXaxis().SetTitle('Time')
                  p.GetYaxis().SetTitle('Flux')
                  p.SetMarkerStyle(21)
                  p.SetMarkerSize(0.9)
                  p.SetMarkerColor(2)
                  p.GetXaxis().SetLabelSize(0.05)
```

```
                    p.GetXaxis().SetTitleSize(0.05)
                    p.GetYaxis().SetLabelSize(0.05)
                    p.GetYaxis().SetTitleSize(0.05)
                    p.GetXaxis().SetTimeDisplay(1)
                    if ((tmpData.Timestamp[tmpData.Timestamp.index.stop-
1]-tmpData.Timestamp[0]) <= 86400):
                          p.GetXaxis().SetTimeFormat("%H:%M")
                    else:
                          p.GetXaxis().SetLabelOffset(0.03)
                          p.GetXaxis().SetTitleOffset(1.5)

      p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
                          p.GetXaxis().SetNdivisions(-510, 0)
                    p.GetXaxis().SetTimeOffset(0,"gmt")
                    p.SetMinimum(0)
                    p.GetXaxis().SetLimits(StartDateTime,StopDateTime)

                    h = TH1F('Flux %c - Without Burst - Local
Method'%(group), 'Flux %c - Without Burst - Local Method'%(group),
nbinFix, min(np.asarray(tmpData["Flux_Raw_%c"%(group)]))*0.995,
max(np.asarray(tmpData["Flux_Raw_%c"%(group)]))*1.005)
                    for value in tmpData["Flux_Raw_%c"%(group)].values:
                        h.Fill(value)
                    h.GetXaxis().SetTitle('Flux')
                    h.GetYaxis().SetTitle('Events')
                    h.GetXaxis().SetLabelSize(0.05)
                    h.GetXaxis().SetTitleSize(0.05)
                    h.GetYaxis().SetLabelSize(0.05)
                    h.GetYaxis().SetTitleSize(0.05)

                    t = TPad("pad1","",0.02,0.02,0.95,0.98)
                    t.Draw()
                    text.Draw()
                    t.Divide(1,3)

                    t.cd(1)
                    gPad.SetRightMargin(0.12)
                    gPad.SetLeftMargin(0.12)
                    gPad.SetTopMargin(0.12)
                    gPad.SetBottomMargin(0.18)
                    h.Draw()

                    t.cd(2)
                    gPad.SetRightMargin(0.12)
                    gPad.SetLeftMargin(0.12)
                    gPad.SetTopMargin(0.12)
                    gPad.SetBottomMargin(0.18)
                    p.Draw("AP")

                    tmpData = Data[Data["Det_%c_Burst_Counter"%(group)]
== 1]
                    tmpData.reset_index(drop=True, inplace=True)
                    pLocalBurst =
TGraph(len(tmpData["Flux_Raw_%c"%(group)]),
np.asarray(tmpData.Timestamp.values).astype('float'),
np.asarray(tmpData["Flux_Raw_%c"%(group)].values).astype('float'))
                    pLocalBurst.SetTitle('Burst Group %c - Local
Method'%(group))
                    pLocalBurst.GetXaxis().SetTitle('Time')
                    pLocalBurst.GetYaxis().SetTitle('Flux')
                    pLocalBurst.SetMarkerStyle(21)
```

```
                  pLocalBurst.SetMarkerSize(0.9)
                  pLocalBurst.SetMarkerColor(2)
                  pLocalBurst.GetXaxis().SetLabelSize(0.05)
                  pLocalBurst.GetXaxis().SetTitleSize(0.05)
                  pLocalBurst.GetYaxis().SetLabelSize(0.05)
                  pLocalBurst.GetYaxis().SetTitleSize(0.05)
                  pLocalBurst.GetXaxis().SetTimeDisplay(1)
                  if ((tmpData.Timestamp[tmpData.Timestamp.index.stop-
1]-tmpData.Timestamp[0]) <= 86400):
                        pLocalBurst.GetXaxis().SetTimeFormat("%H:%M")
                  else:
                        pLocalBurst.GetXaxis().SetLabelOffset(0.03)
                        pLocalBurst.GetXaxis().SetTitleOffset(1.5)

      pLocalBurst.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M
}")
                        pLocalBurst.GetXaxis().SetNdivisions(-510, 0)
                  pLocalBurst.GetXaxis().SetTimeOffset(0,"gmt")
                  pLocalBurst.SetMinimum(0)

      pLocalBurst.GetXaxis().SetLimits(StartDateTime,StopDateTime)

                  t.cd(3)
                  gPad.SetRightMargin(0.12)
                  gPad.SetLeftMargin(0.12)
                  gPad.SetTopMargin(0.12)
                  gPad.SetBottomMargin(0.18)
                  pLocalBurst.Draw("AP")

                  # Saving in PNG
                  imageName =
"LocalMethod_BurstSearch_Group%c.png"%(group)
                  c.Print(os.path.join(args.OutFolder,"Burst-
Forbush_Search",imageName))

      if ((Data["Det_%s_Forbush_Counter"%(group)] == 1).any()):
            tmpData = Data[Data["Det_%s_Forbush_Counter"%(group)] == 1]
            tmpData.reset_index(drop=True, inplace=True)
            # Protection for channel with all zero efficiency (Dead
Channel)
            if ((((tmpData["Eff_Raw_%cCh0"%(group)] == 0).all()) ==
False) and (((tmpData["Eff_Raw_%cCh1"%(group)] == 0).all()) == False)
and (((tmpData["Eff_Raw_%cCh2"%(group)] == 0).all()) == False) and
(((tmpData["Eff_Raw_%cCh3"%(group)] == 0).all()) == False)):
                  c           = TCanvas('Flux Raw %c'%(group),'Flux Raw
%c'%(group),1280,500)
                  p           =
TGraph(len(tmpData["Flux_Raw_%c"%(group)]),
np.asarray(tmpData.Timestamp.values).astype('float'),
np.asarray(tmpData["Flux_Raw_%c"%(group)].values).astype('float'))
                  p.SetTitle('Forbush Group %c - Local Method'%(group))
                  p.GetXaxis().SetTitle('Time')
                  p.GetYaxis().SetTitle('Flux')
                  p.SetMarkerStyle(21)
                  p.SetMarkerSize(0.9)
                  p.SetMarkerColor(2)
                  p.GetXaxis().SetLabelSize(0.05)
                  p.GetXaxis().SetTitleSize(0.05)
                  p.GetYaxis().SetLabelSize(0.05)
                  p.GetYaxis().SetTitleSize(0.05)
                  p.GetXaxis().SetTimeDisplay(1)
```

```
                  if ((tmpData.Timestamp[tmpData.Timestamp.index.stop-
1]-tmpData.Timestamp[0]) <= 86400):
                        p.GetXaxis().SetTimeFormat("%H:%M")
                  else:
                        p.GetXaxis().SetLabelOffset(0.03)
                        p.GetXaxis().SetTitleOffset(1.5)

      p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
                        p.GetXaxis().SetNdivisions(-510, 0)
                  p.GetXaxis().SetTimeOffset(0,"gmt")
                  p.SetMinimum(0)
                  p.GetXaxis().SetLimits(StartDateTime,StopDateTime)

                  t = TPad("pad1","",0.02,0.02,0.95,0.98)
                  t.Draw()
                  text.Draw()
                  t.Divide(1,1)

                  t.cd(1)
                  gPad.SetRightMargin(0.12)
                  gPad.SetLeftMargin(0.12)
                  gPad.SetTopMargin(0.12)
                  gPad.SetBottomMargin(0.18)
                  p.Draw("AP")

                  # Saving in PNG
                  imageName = "LocalMethod_Forbush_Group%c.png"%(group)
                  c.Print(os.path.join(args.OutFolder,"Burst-
Forbush_Search",imageName))

# New Burst/Forbush Identification - Global Method

for groupNum, group in enumerate(GroupsList):
      gStyle.SetOptFit(1)
      yMax  = 1000
      yMin  = 0
      nbins = yMax - yMin
      ht = TH1D('Burst/Forbush Identification - Group %c - Global
Method'%(group), 'Burst/Forbush Identification - Group %c - Global
Method'%(group), nbins, yMin, yMax)
      for value in Data["Flux_Raw_%c"%(group)].values:
            ht.Fill(value)

      avg = ht.GetMean()
      std = ht.GetRMS()

      ht.GetXaxis().SetRangeUser(0,avg+5*std)
      ht.GetXaxis().SetTitle('Counts/10s')
      ht.GetYaxis().SetTitle('Events')
      ht.GetXaxis().SetLabelSize(0.05)
      ht.GetXaxis().SetTitleSize(0.05)
      ht.GetYaxis().SetLabelSize(0.05)
      ht.GetYaxis().SetTitleSize(0.05)

      avg = ht.GetMean()
      std = ht.GetRMS()
      fGaus = TF1("fGaus","gaus", yMin, yMax)
      fGaus.SetParameter(1, avg)
      fGaus.SetParameter(2, std)
      ht.Fit(fGaus, "", "", 0, avg+5*std)
```

```
      avg = fGaus.GetParameter(1)
      std = fGaus.GetParameter(2)
      factor = (Data["Flux_Raw_%c"%(group)].values - avg)/std
      Data["%c_Burst_Forbush_Factor_GlobalMethod"%(group)] = factor
      if (any(Data["%c_Burst_Forbush_Factor_GlobalMethod"%(group)] >
5)):
            c = TCanvas('Burst/Forbush Identification - Group %c -
Global Method'%(group),'Burst/Forbush Identification - Group %c -
Global Method'%(group),1280,1280)
            t = TPad("pad1","",0.02,0.02,0.95,0.98)
            t.Draw()
            text.Draw()
            t.Divide(1,3)

            t.cd(1)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            ht.Draw()

            tmpData =
Data[Data["%c_Burst_Forbush_Factor_GlobalMethod"%(group)] < 5]
            tmpData.reset_index(drop=True, inplace=True)
            p = TGraph(len(tmpData['Flux_Raw_%c'%(group)]),
np.asarray(tmpData.Timestamp.values).astype('float'),
np.asarray(tmpData['Flux_Raw_%c'%(group)].values).astype('float'))
            p.SetTitle('Flux Raw Group %c - Without Bursts - Global
Method'%(group))
            p.GetXaxis().SetTitle('Time')
            p.GetYaxis().SetTitle('Flux')
            p.SetMarkerStyle(21)
            p.SetMarkerSize(0.9)
            p.SetMarkerColor(2)
            p.GetXaxis().SetLabelSize(0.05)
            p.GetXaxis().SetTitleSize(0.05)
            p.GetYaxis().SetLabelSize(0.05)
            p.GetYaxis().SetTitleSize(0.05)
            p.GetXaxis().SetTimeDisplay(1)
            if ((tmpData.Timestamp[tmpData.Timestamp.index.stop-1]-
tmpData.Timestamp[0]) <= 86400):
                  p.GetXaxis().SetTimeFormat("%H:%M")
            else:
                  p.GetXaxis().SetLabelOffset(0.03)
                  p.GetXaxis().SetTitleOffset(1.5)

      p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
                  p.GetXaxis().SetNdivisions(-510, 0)
            p.GetXaxis().SetTimeOffset(0,"gmt")
            p.SetMinimum(0)
            p.GetXaxis().SetLimits(StartDateTime,StopDateTime)

            t.cd(2)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            p.Draw("AP")

            tmpData =
Data[Data["%c_Burst_Forbush_Factor_GlobalMethod"%(group)] > 5]
```

```
            tmpData.reset_index(drop=True, inplace=True)
            ErrX = np.zeros(len(tmpData['Flux_Raw_%c'%(group)]))
            pGlobalBurst = TGraph(len(tmpData['Flux_Raw_%c'%(group)]),
np.asarray(tmpData.Timestamp.values).astype('float'),
np.asarray(tmpData['Flux_Raw_%c'%(group)].values).astype('float'))
            pGlobalBurst.SetTitle("Burst Group %c - Global
Method"%(group))
            pGlobalBurst.GetXaxis().SetTitle('Time')
            pGlobalBurst.GetYaxis().SetTitle('Flux')
            pGlobalBurst.SetMarkerStyle(21)
            pGlobalBurst.SetMarkerSize(0.9)
            pGlobalBurst.SetMarkerColor(2)
            pGlobalBurst.GetXaxis().SetLabelSize(0.05)
            pGlobalBurst.GetXaxis().SetTitleSize(0.05)
            pGlobalBurst.GetYaxis().SetLabelSize(0.05)
            pGlobalBurst.GetYaxis().SetTitleSize(0.05)
            pGlobalBurst.GetXaxis().SetTimeDisplay(1)
            if ((StopDateTime-StartDateTime) <= 5*60):
                  pGlobalBurst.GetXaxis().SetLabelOffset(0.015)
                  pGlobalBurst.GetXaxis().SetTimeFormat("%H:%M:%S")
                  pGlobalBurst.GetXaxis().SetNdivisions(-510, 0)
            elif ((StopDateTime-StartDateTime) <= 86400):
                  pGlobalBurst.GetXaxis().SetLabelOffset(0.015)
                  pGlobalBurst.GetXaxis().SetTimeFormat("%H:%M")
            else:
                  pGlobalBurst.GetXaxis().SetLabelOffset(0.03)
                  pGlobalBurst.GetXaxis().SetTitleOffset(1.5)

      pGlobalBurst.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%
M}")
                  pGlobalBurst.GetXaxis().SetNdivisions(-510, 0)
            pGlobalBurst.GetXaxis().SetTimeOffset(0,"gmt")

      pGlobalBurst.GetXaxis().SetLimits(StartDateTime,StopDateTime)
            pGlobalBurst.SetMinimum(0)

            t.cd(3)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            pGlobalBurst.Draw("AP")

            # Saving in PNG
            imageName = "GlobalMethod_BurstSearch_Group%c.png"%(group)
            c.Print(os.path.join(args.OutFolder,"Burst-
Forbush_Search",imageName))

      else:
            c = TCanvas('Burst/Forbush Identification - Group %c -
Global Method'%(group),'Burst/Forbush Identification - Group %c -
Global Method'%(group),1280,580)
            t = TPad("pad1","",0.02,0.02,0.95,0.98)
            t.Draw()
            text.Draw()
            t.Divide(1,1)

            t.cd(1)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
```

```
            gPad.SetBottomMargin(0.18)
            ht.Draw()
            c.Update()

            # Saving in PNG
            imageName = "GlobalMethod_BurstSearch_Group%c.png"%(group)
            c.Print(os.path.join(args.OutFolder,"Burst-
Forbush_Search",imageName))

      if (any(Data["%c_Burst_Forbush_Factor_GlobalMethod"%(group)] < -
5)):
            tmpData =
Data[Data["%c_Burst_Forbush_Factor_GlobalMethod"%(group)] < -5]
            tmpData.reset_index(drop=True, inplace=True)
            p = TGraph(len(tmpData['Flux_Raw_%c'%(group)]),
np.asarray(tmpData.Timestamp.values).astype('float'),
np.asarray(tmpData['Flux_Raw_%c'%(group)].values).astype('float'))
            p.SetTitle('Forbush Group %c - Global Method'%(group))
            p.GetXaxis().SetTitle('Time')
            p.GetYaxis().SetTitle('Flux')
            p.SetMarkerStyle(21)
            p.SetMarkerSize(0.9)
            p.SetMarkerColor(2)
            p.GetXaxis().SetLabelSize(0.05)
            p.GetXaxis().SetTitleSize(0.05)
            p.GetYaxis().SetLabelSize(0.05)
            p.GetYaxis().SetTitleSize(0.05)
            p.GetXaxis().SetTimeDisplay(1)
            if ((tmpData.Timestamp[tmpData.Timestamp.index.stop-1]-
tmpData.Timestamp[0]) <= 86400):
                  p.GetXaxis().SetTimeFormat("%H:%M")
            else:
                  p.GetXaxis().SetLabelOffset(0.03)
                  p.GetXaxis().SetTitleOffset(1.5)

      p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
                  p.GetXaxis().SetNdivisions(-510, 0)
            p.GetXaxis().SetTimeOffset(0,"gmt")
            p.SetMinimum(0)
            p.GetXaxis().SetLimits(StartDateTime,StopDateTime)

            t = TPad("pad1","",0.02,0.02,0.95,0.98)
            t.Draw()
            text.Draw()
            t.Divide(1,1)

            t.cd(1)
            gPad.SetRightMargin(0.12)
            gPad.SetLeftMargin(0.12)
            gPad.SetTopMargin(0.12)
            gPad.SetBottomMargin(0.18)
            p.Draw("AP")

            # Saving in PNG
            imageName = "GlobalMethod_Forbush_Group%c.png"%(group)
            c.Print(os.path.join(args.OutFolder,"Burst-
Forbush_Search",imageName))

# Getting Daily Efficiency for Each Channel
DailyEff    = []
DailyEffRms = []
```

```
tmpData = Data[((Data["A_Burst_Forbush_Factor_GlobalMethod"] < 5) &
(Data["B_Burst_Forbush_Factor_GlobalMethod"] < 5) &
(Data["C_Burst_Forbush_Factor_GlobalMethod"] < 5))]
for groupNum, group in enumerate(GroupsList):
      DailyEff.append([])
      DailyEffRms.append([])
      for ch in range(0,NumOfChsInGroups):
            DailyEff[groupNum].append(0.0)
            DailyEffRms[groupNum].append(0.0)
            DailyEff[groupNum][ch]        =
tmpData["Eff_Raw_%cCh%d"%(group, ch)].mean()
            DailyEffRms[groupNum][ch]     =
tmpData["Eff_Raw_%cCh%d"%(group, ch)].std()

tmpData = Data[((Data["Det_A_Burst_Counter"] == 0) &
(Data["Det_B_Burst_Counter"] == 0) & (Data["Det_C_Burst_Counter"] ==
0))]
tmpData = tmpData.sum()

Flux_A                  =
(tmpData["A0&1&2&3"]/(DailyEff[0][0]*DailyEff[0][1]*DailyEff[0][2]*Dai
lyEff[0][3]*DetArea*tmpData["AcqTime"]))
Flux_B                  =
(tmpData["B0&1&2&3"]/(DailyEff[1][0]*DailyEff[1][1]*DailyEff[1][2]*Dai
lyEff[1][3]*DetArea*tmpData["AcqTime"]))
Flux_C                  =
(tmpData["C0&1&2&3"]/(DailyEff[2][0]*DailyEff[2][1]*DailyEff[2][2]*Dai
lyEff[2][3]*DetArea*tmpData["AcqTime"]))

Flux_A01_B01      =
(tmpData["A01&B01"]/(DailyEff[0][0]*DailyEff[0][1]*DailyEff[1][0]*Dail
yEff[1][1]*DetArea*tmpData["AcqTime"]))
Flux_A01_B12      =
(tmpData["A01&B12"]/(DailyEff[0][0]*DailyEff[0][1]*DailyEff[1][1]*Dail
yEff[1][2]*DetArea*tmpData["AcqTime"]))
Flux_A01_B23      =
(tmpData["A01&B23"]/(DailyEff[0][0]*DailyEff[0][1]*DailyEff[1][2]*Dail
yEff[1][3]*DetArea*tmpData["AcqTime"]))
Flux_A12_B01      =
(tmpData["A12&B01"]/(DailyEff[0][1]*DailyEff[0][2]*DailyEff[1][0]*Dail
yEff[1][1]*DetArea*tmpData["AcqTime"]))
Flux_A12_B12      =
(tmpData["A12&B12"]/(DailyEff[0][1]*DailyEff[0][2]*DailyEff[1][1]*Dail
yEff[1][2]*DetArea*tmpData["AcqTime"]))
Flux_A12_B23      =
(tmpData["A12&B23"]/(DailyEff[0][1]*DailyEff[0][2]*DailyEff[1][2]*Dail
yEff[1][3]*DetArea*tmpData["AcqTime"]))
Flux_A23_B01      =
(tmpData["A23&B01"]/(DailyEff[0][2]*DailyEff[0][3]*DailyEff[1][0]*Dail
yEff[1][1]*DetArea*tmpData["AcqTime"]))
Flux_A23_B12      =
(tmpData["A23&B12"]/(DailyEff[0][2]*DailyEff[0][3]*DailyEff[1][1]*Dail
yEff[1][2]*DetArea*tmpData["AcqTime"]))
Flux_A23_B23      =
(tmpData["A23&B23"]/(DailyEff[0][2]*DailyEff[0][3]*DailyEff[1][2]*Dail
yEff[1][3]*DetArea*tmpData["AcqTime"]))

Flux_A01_C01      =
(tmpData["A01&C01"]/(DailyEff[0][0]*DailyEff[0][1]*DailyEff[2][0]*Dail
yEff[2][1]*DetArea*tmpData["AcqTime"]))
```

```
Flux_A01_C12      =
(tmpData["A01&C12"]/(DailyEff[0][0]*DailyEff[0][1]*DailyEff[2][1]*Dail
yEff[2][2]*DetArea*tmpData["AcqTime"]))
Flux_A01_C23      =
(tmpData["A01&C23"]/(DailyEff[0][0]*DailyEff[0][1]*DailyEff[2][2]*Dail
yEff[2][3]*DetArea*tmpData["AcqTime"]))
Flux_A12_C01      =
(tmpData["A12&C01"]/(DailyEff[0][1]*DailyEff[0][2]*DailyEff[2][0]*Dail
yEff[2][1]*DetArea*tmpData["AcqTime"]))
Flux_A12_C12      =
(tmpData["A12&C12"]/(DailyEff[0][1]*DailyEff[0][2]*DailyEff[2][1]*Dail
yEff[2][2]*DetArea*tmpData["AcqTime"]))
Flux_A12_C23      =
(tmpData["A12&C23"]/(DailyEff[0][1]*DailyEff[0][2]*DailyEff[2][2]*Dail
yEff[2][3]*DetArea*tmpData["AcqTime"]))
Flux_A23_C01      =
(tmpData["A23&C01"]/(DailyEff[0][2]*DailyEff[0][3]*DailyEff[2][0]*Dail
yEff[2][1]*DetArea*tmpData["AcqTime"]))
Flux_A23_C12      =
(tmpData["A23&C12"]/(DailyEff[0][2]*DailyEff[0][3]*DailyEff[2][1]*Dail
yEff[2][2]*DetArea*tmpData["AcqTime"]))
Flux_A23_C23      =
(tmpData["A23&C23"]/(DailyEff[0][2]*DailyEff[0][3]*DailyEff[2][2]*Dail
yEff[2][3]*DetArea*tmpData["AcqTime"]))

Flux_B01_C01      =
(tmpData["B01&C01"]/(DailyEff[1][0]*DailyEff[1][1]*DailyEff[2][0]*Dail
yEff[2][1]*DetArea*tmpData["AcqTime"]))
Flux_B01_C12      =
(tmpData["B01&C12"]/(DailyEff[1][0]*DailyEff[1][1]*DailyEff[2][1]*Dail
yEff[2][2]*DetArea*tmpData["AcqTime"]))
Flux_B01_C23      =
(tmpData["B01&C23"]/(DailyEff[1][0]*DailyEff[1][1]*DailyEff[2][2]*Dail
yEff[2][3]*DetArea*tmpData["AcqTime"]))
Flux_B12_C01      =
(tmpData["B12&C01"]/(DailyEff[1][1]*DailyEff[1][2]*DailyEff[2][0]*Dail
yEff[2][1]*DetArea*tmpData["AcqTime"]))
Flux_B12_C12      =
(tmpData["B12&C12"]/(DailyEff[1][1]*DailyEff[1][2]*DailyEff[2][1]*Dail
yEff[2][2]*DetArea*tmpData["AcqTime"]))
Flux_B12_C23      =
(tmpData["B12&C23"]/(DailyEff[1][1]*DailyEff[1][2]*DailyEff[2][2]*Dail
yEff[2][3]*DetArea*tmpData["AcqTime"]))
Flux_B23_C01      =
(tmpData["B23&C01"]/(DailyEff[1][2]*DailyEff[1][3]*DailyEff[2][0]*Dail
yEff[2][1]*DetArea*tmpData["AcqTime"]))
Flux_B23_C12      =
(tmpData["B23&C12"]/(DailyEff[1][2]*DailyEff[1][3]*DailyEff[2][1]*Dail
yEff[2][2]*DetArea*tmpData["AcqTime"]))
Flux_B23_C23      =
(tmpData["B23&C23"]/(DailyEff[1][2]*DailyEff[1][3]*DailyEff[2][2]*Dail
yEff[2][3]*DetArea*tmpData["AcqTime"]))

R_A01_B01         = (Flux_A01_B01/(Flux_A + Flux_B))
R_A01_B12         = (Flux_A01_B12/(Flux_A + Flux_B))
R_A01_B23         = (Flux_A01_B23/(Flux_A + Flux_B))
R_A12_B01         = (Flux_A12_B01/(Flux_A + Flux_B))
R_A12_B12         = (Flux_A12_B12/(Flux_A + Flux_B))
R_A12_B23         = (Flux_A12_B23/(Flux_A + Flux_B))
R_A23_B01         = (Flux_A23_B01/(Flux_A + Flux_B))
R_A23_B12         = (Flux_A23_B12/(Flux_A + Flux_B))
```

```
R_A23_B23         = (Flux_A23_B23/(Flux_A + Flux_B))

R_A01_C01         = (Flux_A01_C01/(Flux_A + Flux_C))
R_A01_C12         = (Flux_A01_C12/(Flux_A + Flux_C))
R_A01_C23         = (Flux_A01_C23/(Flux_A + Flux_C))
R_A12_C01         = (Flux_A12_C01/(Flux_A + Flux_C))
R_A12_C12         = (Flux_A12_C12/(Flux_A + Flux_C))
R_A12_C23         = (Flux_A12_C23/(Flux_A + Flux_C))
R_A23_C01         = (Flux_A23_C01/(Flux_A + Flux_C))
R_A23_C12         = (Flux_A23_C12/(Flux_A + Flux_C))
R_A23_C23         = (Flux_A23_C23/(Flux_A + Flux_C))

R_B01_C01         = (Flux_B01_C01/(Flux_B + Flux_C))
R_B01_C12         = (Flux_B01_C12/(Flux_B + Flux_C))
R_B01_C23         = (Flux_B01_C23/(Flux_B + Flux_C))
R_B12_C01         = (Flux_B12_C01/(Flux_B + Flux_C))
R_B12_C12         = (Flux_B12_C12/(Flux_B + Flux_C))
R_B12_C23         = (Flux_B12_C23/(Flux_B + Flux_C))
R_B23_C01         = (Flux_B23_C01/(Flux_B + Flux_C))
R_B23_C12         = (Flux_B23_C12/(Flux_B + Flux_C))
R_B23_C23         = (Flux_B23_C23/(Flux_B + Flux_C))

R_df = pd.DataFrame({'R_A01B01': [R_A01_B01],
                     'R_A01B12': [R_A01_B12],
                     'R_A01B23': [R_A01_B23],
                     'R_A12B01': [R_A12_B01],
                     'R_A12B12': [R_A12_B12],
                     'R_A12B23': [R_A12_B23],
                            'R_A23B01': [R_A23_B01],
                     'R_A23B12': [R_A23_B12],
                     'R_A23B23': [R_A23_C23],
                            'R_A01C01': [R_A01_C01],
                     'R_A01C12': [R_A01_C12],
                     'R_A01C23': [R_A01_C23],
                     'R_A12C01': [R_A12_C01],
                     'R_A12C12': [R_A12_C12],
                     'R_A12C23': [R_A12_C23],
                            'R_A23C01': [R_A23_C01],
                     'R_A23C12': [R_A23_C12],
                     'R_A23C23': [R_A23_C23],
                            'R_B01C01': [R_B01_C01],
                     'R_B01C12': [R_B01_C12],
                     'R_B01C23': [R_B01_C23],
                     'R_B12C01': [R_B12_C01],
                     'R_B12C12': [R_B12_C12],
                     'R_B12C23': [R_B12_C23],
                            'R_B23C01': [R_B23_C01],
                     'R_B23C12': [R_B23_C12],
                     'R_B23C23': [R_B23_C23]})

MapData     = pd.read_csv(args.MapFile, sep="\t")
ZAfFound          = 0
g                 = TGraphErrors(len(MapData.ZenithAngleFactor.values))
Chi2SumArray = []
for pos, N2find in enumerate(MapData.ZenithAngleFactor.values):
      MapZenSelData     = (MapData.loc[MapData["ZenithAngleFactor"] ==
N2find]).reset_index(drop=True)
      chi2Sum     = ((MapZenSelData[R_Columns] - R_df) **
2).values.sum() / 27
      Chi2SumArray.append(chi2Sum)
      g.SetPoint(pos,N2find,chi2Sum)
```

CRE4AT_DailyAnalysis_IPANEMA_SiPM.py

```
        g.SetPointError(pos, 0.01, 0.00001)
        if (pos == 0):
                LowestNfSum = chi2Sum
                ZAfFound          = N2find
        if (chi2Sum < LowestNfSum):
                LowestNfSum = chi2Sum
                ZAfFound          = N2find
gStyle.SetOptFit(0)
c      = TCanvas("Chi2Sum", "Chi2Sum", 1280, 800)
g.GetXaxis().SetTitle("Zenith Angle Factor")
g.GetYaxis().SetTitle("#chi^{2}")
g.SetMarkerSize(1)
g.SetMarkerStyle(21)
g.SetMarkerColor(2)
g.SetTitle("Zenith Angle Factor Search")
fPol2 = TF1("fPol2", "pol2")
fPol2.SetLineColor(4)
g.Fit(fPol2, "", "", ZAfFound-(ZAFSearchPoints2Add*0.01),
ZAfFound+(ZAFSearchPoints2Add*0.01))
t = TPad("pad1","",0,0,0.95,1)
text.Draw()
t.Draw()
t.cd()
gPad.SetRightMargin(0.05)
gPad.SetLeftMargin(0.12)
gPad.SetTopMargin(0.12)
gPad.SetBottomMargin(0.12)
g.Draw("AP")
padZoom = TPad("p","p", 0.6,0.5,0.98,0.92)
padZoom.SetLeftMargin(0.05)
padZoom.SetRightMargin(0.01)
padZoom.SetBottomMargin(0.05)
padZoom.SetTopMargin(0.32)
padZoom.Draw()
padZoom.cd()
gPad.SetLeftMargin(0.1)
gcp = g.DrawClone("AP")
gcp.SetTitle("")
gcp.GetXaxis().SetRangeUser(ZAfFound-
(ZAFSearchPoints2Add*0.01),ZAfFound+(ZAFSearchPoints2Add*0.01))
min = g.GetYaxis().GetXmin()
max = g.GetYaxis().GetXmax()
gcp.GetYaxis().SetRangeUser(np.min(Chi2SumArray)-min/4,
np.min(Chi2SumArray)+min/4)
gcp.GetXaxis().SetLabelSize(0.05)
gcp.GetYaxis().SetLabelSize(0.05)
gcp.GetXaxis().SetTitle("")
gcp.GetYaxis().SetTitle("")
gcp.Draw("AP")
ZAf         = (-fPol2.GetParameter(1))/(2*fPol2.GetParameter(2))
ZAfErr      = ZAf *
((((fPol2.GetParError(1)/fPol2.GetParameter(1))**2)+((fPol2.GetParErro
r(2)/fPol2.GetParameter(2))**2))**(1/2))
ChiNDF      = (fPol2.GetChisquare())/(fPol2.GetNDF())
print ("*******************************\n\nZAF found by Fit:
%.4f\nZAF Fit Error: %.8f\nChi2/NDF:
%.8f\n\n*******************************"%(ZAf, ZAfErr, ChiNDF))
pt = TPaveText(0,0.75,0.99,1)
pt.AddText("Simple Zenith Angle Factor Found = %.2f"%(ZAfFound))
```

```
pt.AddText("Fit Function:
%.6f*(ZA_{f})^{2}%.6f*ZA_{f}+%.6f"%(fPol2.GetParameter(2),
fPol2.GetParameter(1), fPol2.GetParameter(0)))
pt.AddText("Zenith Angle Factor Found by Fit = %.4f +- %.4f"%(ZAf,
ZAfErr))
pt.Paint("NDC")
pt.Draw()
# Saving in PNG
imageName = "ZenitalAngleFactor_Search.png"
c.Print(os.path.join(args.OutFolder,"Zenith_Angle_Factor_Search",image
Name))

EffCorrection     = []
# Define NpFactor Corretion
MapZenSelData     = (MapData.loc[MapData["ZenithAngleFactor"] ==
round(ZAf, 2)]).reset_index(drop=True)
EffCorrection.append(MapZenSelData.fGeo0.values[0])
EffCorrection.append(MapZenSelData.fGeo1.values[0])
EffCorrection.append(MapZenSelData.fGeo2.values[0])
EffCorrection.append(MapZenSelData.fGeo3.values[0])
FluxCorrection    = MapZenSelData.fFluxGeo.values[0]

# Correcting Efficiencies and Flux and plotting Flux Corrected
gStyle.SetOptFit(1)
DailyFluxCorrected            = []
DailyFluxCorrectedErr   = []
for groupNum, group in enumerate(GroupsList):
      DailyFluxCorrected.append([])
      DailyFluxCorrectedErr.append([])
      for ch in range(0,NumOfChsInGroups):
            Data["Eff_%cCh%d"%(group, ch)]            =
Data["Eff_Raw_%cCh%d"%(group, ch)] * EffCorrection[ch]
      Data["Flux_%c"%(group)] =
(Data["%c0&1&2&3"%(group)]*FluxCorrection)/((DailyEff[groupNum][0]*Eff
Correction[0])*(DailyEff[groupNum][1]*EffCorrection[1])*(DailyEff[grou
pNum][2]*EffCorrection[2])*(DailyEff[groupNum][3]*EffCorrection[3])*De
tArea*Data["AcqTime"])

tmpData = Data[((Data["Det_A_Burst_Counter"] == 0) &
(Data["Det_B_Burst_Counter"] == 0) & (Data["Det_C_Burst_Counter"] ==
0))]

for groupNum, group in enumerate(GroupsList):
      p = TGraph(len(tmpData["Flux_%c"%(group)]),
np.asarray(tmpData.Timestamp.values).astype('float'),
np.asarray(tmpData["Flux_%c"%(group)]).astype(float))
      p.SetTitle("TimePlot")
      p.GetXaxis().SetTitle('Time')
      p.GetYaxis().SetTitle('Flux')
      p.SetMarkerStyle(21)
      p.SetMarkerSize(0.9)
      p.SetMarkerColor(2)
      p.GetXaxis().SetLabelSize(0.05)
      p.GetXaxis().SetTitleSize(0.05)
      p.GetYaxis().SetLabelSize(0.05)
      p.GetYaxis().SetTitleSize(0.05)
      p.GetXaxis().SetTimeDisplay(1)
      if ((Data.Timestamp[Data.Timestamp.index.stop-1]-
Data.Timestamp[0]) <= 86400):
            p.GetXaxis().SetTimeFormat("%H:%M")
      else:
```

```
            p.GetXaxis().SetLabelOffset(0.03)
            p.GetXaxis().SetTitleOffset(1.5)
            p.GetXaxis().SetTimeFormat("#splitline{%d/%m/%y}{%H:%M}")
            p.GetXaxis().SetNdivisions(-510, 0)
      p.GetXaxis().SetTimeOffset(0,"gmt")
      p.SetMinimum(0)

      h = TH1F('Flux Group %c'%(group), 'Flux Group %c'%(group),
int(nbinFix/4), np.min(tmpData["Flux_%c"%(group)])*0.995,
np.max(tmpData["Flux_%c"%(group)])*1.005)
      for value in tmpData["Flux_%c"%(group)].values:
            h.Fill(value)
      h.GetXaxis().SetTitle('Flux')
      h.GetYaxis().SetTitle('Events')
      h.GetXaxis().SetLabelSize(0.05)
      h.GetXaxis().SetTitleSize(0.05)
      h.GetYaxis().SetLabelSize(0.05)
      h.GetYaxis().SetTitleSize(0.05)
      fGaus = TF1("fGaus", "gaus")
      h.Fit(fGaus)
      DailyFluxCorrected[groupNum]  = fGaus.GetParameter(1)
      DailyFluxCorrectedErr[groupNum]     = fGaus.GetParError(1)

      cf = TCanvas('Flux %c'%(group), 'Flux %c'%(group), 1280, 980)

      t = TPad("pad1","",0.02,0.02,0.95,0.98)
      t.Draw()
      text.Draw()
      t.Divide(1,2)

      t.cd(1)
      gPad.SetRightMargin(0.12)
      gPad.SetLeftMargin(0.12)
      gPad.SetTopMargin(0.12)
      gPad.SetBottomMargin(0.18)
      h.Draw()

      t.cd(2)
      gPad.SetRightMargin(0.12)
      gPad.SetLeftMargin(0.12)
      gPad.SetTopMargin(0.12)
      gPad.SetBottomMargin(0.18)
      p.Draw("AP")

      # Saving in PNG
      imageName = "Flux_Corrected_Group%c.png"%(group)
      cf.Print(os.path.join(args.OutFolder,"Flux_Corrected",imageName)
)

# Creating Daily ROOT File
fROOT = TFile(os.path.join(args.ROOTFolderOutput,
"CRE4AT_%s_%s.root"%(Location,
datetime.utcfromtimestamp(Data.Timestamp[0]).strftime('%Y%m%d'))),
'recreate')
# Making Tree
tcre4at = TTree('CRE4AT_%s'%(Location), 'CRE4AT_%s'%(Location))

# Creating Arrays
Timestamp = array('I', [0])
GPS_Timestamp = array('d', [0])
RTC_Timestamp = array('I', [0])
```

```
ACh0 = array('I', [0])
ACh1 = array('I', [0])
ACh2 = array('I', [0])
ACh3 = array('I', [0])
BCh0 = array('I', [0])
BCh1 = array('I', [0])
BCh2 = array('I', [0])
BCh3 = array('I', [0])
CCh0 = array('I', [0])
CCh1 = array('I', [0])
CCh2 = array('I', [0])
CCh3 = array('I', [0])

A0and1 = array('I', [0])
A0and2 = array('I', [0])
A0and3 = array('I', [0])
A1and2 = array('I', [0])
A1and3 = array('I', [0])
A2and3 = array('I', [0])
B0and1 = array('I', [0])
B0and2 = array('I', [0])
B0and3 = array('I', [0])
B1and2 = array('I', [0])
B1and3 = array('I', [0])
B2and3 = array('I', [0])
C0and1 = array('I', [0])
C0and2 = array('I', [0])
C0and3 = array('I', [0])
C1and2 = array('I', [0])
C1and3 = array('I', [0])
C2and3 = array('I', [0])

A0and1and2 = array('I', [0])
A0and1and3 = array('I', [0])
A0and2and3 = array('I', [0])
A1and2and3 = array('I', [0])
B0and1and2 = array('I', [0])
B0and1and3 = array('I', [0])
B0and2and3 = array('I', [0])
B1and2and3 = array('I', [0])
C0and1and2 = array('I', [0])
C0and1and3 = array('I', [0])
C0and2and3 = array('I', [0])
C1and2and3 = array('I', [0])

A0and1and2and3 = array('I', [0])
B0and1and2and3 = array('I', [0])
C0and1and2and3 = array('I', [0])

A01andB01 = array('I', [0])
A01andB12 = array('I', [0])
A01andB23 = array('I', [0])
A12andB01 = array('I', [0])
A12andB12 = array('I', [0])
A12andB23 = array('I', [0])
A23andB01 = array('I', [0])
A23andB12 = array('I', [0])
A23andB23 = array('I', [0])
A01andC01 = array('I', [0])
A01andC12 = array('I', [0])
```

```
A01andC23 = array('I', [0])
A12andC01 = array('I', [0])
A12andC12 = array('I', [0])
A12andC23 = array('I', [0])
A23andC01 = array('I', [0])
A23andC12 = array('I', [0])
A23andC23 = array('I', [0])
B01andC01 = array('I', [0])
B01andC12 = array('I', [0])
B01andC23 = array('I', [0])
B12andC01 = array('I', [0])
B12andC12 = array('I', [0])
B12andC23 = array('I', [0])
B23andC01 = array('I', [0])
B23andC12 = array('I', [0])
B23andC23 = array('I', [0])

AccX = array('f', [0])
AccY = array('f', [0])
AccZ = array('f', [0])
GyroX = array('f', [0])
GyroY = array('f', [0])
GyroZ = array('f', [0])
MagX = array('f', [0])
MagY = array('f', [0])
MagZ = array('f', [0])
Temp = array('f', [0])
Hum = array('f', [0])
Press = array('f', [0])
HV_Temp = array('f', [0])
HV_Voltage = array('f', [0])
LED_CALIB_Status = array('I', [0])

Eff_Raw_ACh0 = array('f', [0])
Eff_Raw_ACh1 = array('f', [0])
Eff_Raw_ACh2 = array('f', [0])
Eff_Raw_ACh3 = array('f', [0])
Eff_Raw_BCh0 = array('f', [0])
Eff_Raw_BCh1 = array('f', [0])
Eff_Raw_BCh2 = array('f', [0])
Eff_Raw_BCh3 = array('f', [0])
Eff_Raw_CCh0 = array('f', [0])
Eff_Raw_CCh1 = array('f', [0])
Eff_Raw_CCh2 = array('f', [0])
Eff_Raw_CCh3 = array('f', [0])

Flux_Raw_A = array('f', [0])
Flux_Raw_B = array('f', [0])
Flux_Raw_C = array('f', [0])

Flux_A = array('f', [0])
Flux_B = array('f', [0])
Flux_C = array('f', [0])

Burst_LocalMethod_A = array('I', [0])
Burst_LocalMethod_B = array('I', [0])
Burst_LocalMethod_C = array('I', [0])

BurstForbushFactor_GlobalMethod_A = array('f', [0])
BurstForbushFactor_GlobalMethod_B = array('f', [0])
BurstForbushFactor_GlobalMethod_C = array('f', [0])
```

```
if(PIDAcquisition):
      PID_Sensor1 = array('f', [0])
      PID_Sensor2 = array('f', [0])
      PID_Avg     = array('f', [0])
      PID_Flag    = array('i', [0])

# Adding Branches
tcre4at.Branch('Timestamp', Timestamp, 'Timestamp/I')
tcre4at.Branch('GPS_Timestamp', GPS_Timestamp, 'GPS_Timestamp/D')
tcre4at.Branch('RTC_Timestamp', RTC_Timestamp, 'RTC_Timestamp/I')

tcre4at.Branch('ACh0', ACh0, 'ACh0/I')
tcre4at.Branch('ACh1', ACh1, 'ACh1/I')
tcre4at.Branch('ACh2', ACh2, 'ACh2/I')
tcre4at.Branch('ACh3', ACh3, 'ACh3/I')
tcre4at.Branch('BCh0', BCh0, 'BCh0/I')
tcre4at.Branch('BCh1', BCh1, 'BCh1/I')
tcre4at.Branch('BCh2', BCh2, 'BCh2/I')
tcre4at.Branch('BCh3', BCh3, 'BCh3/I')
tcre4at.Branch('CCh0', CCh0, 'CCh0/I')
tcre4at.Branch('CCh1', CCh1, 'CCh1/I')
tcre4at.Branch('CCh2', CCh2, 'CCh2/I')
tcre4at.Branch('CCh3', CCh3, 'CCh3/I')

tcre4at.Branch('A0and1', A0and1, 'A0and1/I')
tcre4at.Branch('A0and2', A0and2, 'A0and2/I')
tcre4at.Branch('A0and3', A0and3, 'A0and3/I')
tcre4at.Branch('A1and2', A1and2, 'A1and2/I')
tcre4at.Branch('A1and3', A1and3, 'A1and3/I')
tcre4at.Branch('A2and3', A2and3, 'A2and3/I')
tcre4at.Branch('B0and1', B0and1, 'B0and1/I')
tcre4at.Branch('B0and2', B0and2, 'B0and2/I')
tcre4at.Branch('B0and3', B0and3, 'B0and3/I')
tcre4at.Branch('B1and2', B1and2, 'B1and2/I')
tcre4at.Branch('B1and3', B1and3, 'B1and3/I')
tcre4at.Branch('B2and3', B2and3, 'B2and3/I')
tcre4at.Branch('C0and1', C0and1, 'C0and1/I')
tcre4at.Branch('C0and2', C0and2, 'C0and2/I')
tcre4at.Branch('C0and3', C0and3, 'C0and3/I')
tcre4at.Branch('C1and2', C1and2, 'C1and2/I')
tcre4at.Branch('C1and3', C1and3, 'C1and3/I')
tcre4at.Branch('C2and3', C2and3, 'C2and3/I')

tcre4at.Branch('A0and1and2', A0and1and2, 'A0and1and2/I')
tcre4at.Branch('A0and1and3', A0and1and3, 'A0and1and3/I')
tcre4at.Branch('A0and2and3', A0and2and3, 'A0and2and3/I')
tcre4at.Branch('A1and2and3', A1and2and3, 'A1and2and3/I')
tcre4at.Branch('B0and1and2', B0and1and2, 'B0and1and2/I')
tcre4at.Branch('B0and1and3', B0and1and3, 'B0and1and3/I')
tcre4at.Branch('B0and2and3', B0and2and3, 'B0and2and3/I')
tcre4at.Branch('B1and2and3', B1and2and3, 'B1and2and3/I')
tcre4at.Branch('C0and1and2', C0and1and2, 'C0and1and2/I')
tcre4at.Branch('C0and1and3', C0and1and3, 'C0and1and3/I')
tcre4at.Branch('C0and2and3', C0and2and3, 'C0and2and3/I')
tcre4at.Branch('C1and2and3', C1and2and3, 'C1and2and3/I')

tcre4at.Branch('A0and1an2and3', A0and1and2and3, 'A0and1an2and3/I')
tcre4at.Branch('B0and1an2and3', B0and1and2and3, 'B0and1an2and3/I')
tcre4at.Branch('C0and1an2and3', C0and1and2and3, 'C0and1an2and3/I')
```

```
tcre4at.Branch('A01andB01', A01andB01, 'A01andB01/I')
tcre4at.Branch('A01andB12', A01andB12, 'A01andB12/I')
tcre4at.Branch('A01andB23', A01andB23, 'A01andB23/I')
tcre4at.Branch('A12andB01', A12andB01, 'A12andB01/I')
tcre4at.Branch('A12andB12', A12andB12, 'A12andB12/I')
tcre4at.Branch('A12andB23', A12andB23, 'A12andB23/I')
tcre4at.Branch('A23andB01', A23andB01, 'A23andB01/I')
tcre4at.Branch('A23andB12', A23andB12, 'A23andB12/I')
tcre4at.Branch('A23andB23', A23andB23, 'A23andB23/I')
tcre4at.Branch('A01andC01', A01andC01, 'A01andC01/I')
tcre4at.Branch('A01andC12', A01andC12, 'A01andC12/I')
tcre4at.Branch('A01andC23', A01andC23, 'A01andC23/I')
tcre4at.Branch('A12andC01', A12andC01, 'A12andC01/I')
tcre4at.Branch('A12andC12', A12andC12, 'A12andC12/I')
tcre4at.Branch('A12andC23', A12andC23, 'A12andC23/I')
tcre4at.Branch('A23andC01', A23andC01, 'A23andC01/I')
tcre4at.Branch('A23andC12', A23andC12, 'A23andC12/I')
tcre4at.Branch('A23andC23', A23andC23, 'A23andC23/I')
tcre4at.Branch('B01andC01', B01andC01, 'B01andC01/I')
tcre4at.Branch('B01andC12', B01andC12, 'B01andC12/I')
tcre4at.Branch('B01andC23', B01andC23, 'B01andC23/I')
tcre4at.Branch('B12andC01', B12andC01, 'B12andC01/I')
tcre4at.Branch('B12andC12', B12andC12, 'B12andC12/I')
tcre4at.Branch('B12andC23', B12andC23, 'B12andC23/I')
tcre4at.Branch('B23andC01', B23andC01, 'B23andC01/I')
tcre4at.Branch('B23andC12', B23andC12, 'B23andC12/I')
tcre4at.Branch('B23andC23', B23andC23, 'B23andC23/I')

tcre4at.Branch('AccX', AccX, 'AccX/F')
tcre4at.Branch('AccY', AccY, 'AccY/F')
tcre4at.Branch('AccZ', AccZ, 'AccZ/F')
tcre4at.Branch('GyroX', GyroX, 'GyroX/F')
tcre4at.Branch('GyroY', GyroY, 'GyroY/F')
tcre4at.Branch('GyroZ', GyroZ, 'GyroZ/F')
tcre4at.Branch('MagX', MagX, 'MagX/F')
tcre4at.Branch('MagY', MagY, 'MagY/F')
tcre4at.Branch('MagZ', MagZ, 'MagZ/F')
tcre4at.Branch('Temp', Temp, 'Temp/F')
tcre4at.Branch('Hum', Hum, 'Hum/F')
tcre4at.Branch('Press', Press, 'Press/F')
tcre4at.Branch('HV_Temp', HV_Temp, 'HV_Temp/F')
tcre4at.Branch('HV_Voltage', HV_Voltage, 'HV_Voltage/F')
tcre4at.Branch('LED_CALIB_Status', LED_CALIB_Status,
'LED_CALIB_Status/I')

tcre4at.Branch('Eff_Raw_ACh0', Eff_Raw_ACh0, 'Eff_Raw_ACh0/F')
tcre4at.Branch('Eff_Raw_ACh1', Eff_Raw_ACh1, 'Eff_Raw_ACh1/F')
tcre4at.Branch('Eff_Raw_ACh2', Eff_Raw_ACh2, 'Eff_Raw_ACh2/F')
tcre4at.Branch('Eff_Raw_ACh3', Eff_Raw_ACh3, 'Eff_Raw_ACh3/F')
tcre4at.Branch('Eff_Raw_BCh0', Eff_Raw_BCh0, 'Eff_Raw_BCh0/F')
tcre4at.Branch('Eff_Raw_BCh1', Eff_Raw_BCh1, 'Eff_Raw_BCh1/F')
tcre4at.Branch('Eff_Raw_BCh2', Eff_Raw_BCh2, 'Eff_Raw_BCh2/F')
tcre4at.Branch('Eff_Raw_BCh3', Eff_Raw_BCh3, 'Eff_Raw_BCh3/F')
tcre4at.Branch('Eff_Raw_CCh0', Eff_Raw_CCh0, 'Eff_Raw_CCh0/F')
tcre4at.Branch('Eff_Raw_CCh1', Eff_Raw_CCh1, 'Eff_Raw_CCh1/F')
tcre4at.Branch('Eff_Raw_CCh2', Eff_Raw_CCh2, 'Eff_Raw_CCh2/F')
tcre4at.Branch('Eff_Raw_CCh3', Eff_Raw_CCh3, 'Eff_Raw_CCh3/F')

tcre4at.Branch('Flux_Raw_A', Flux_Raw_A, 'Flux_Raw_A/F')
tcre4at.Branch('Flux_Raw_B', Flux_Raw_B, 'Flux_Raw_B/F')
tcre4at.Branch('Flux_Raw_C', Flux_Raw_C, 'Flux_Raw_C/F')
```

```
tcre4at.Branch('Flux_A', Flux_A, 'Flux_A/F')
tcre4at.Branch('Flux_B', Flux_B, 'Flux_B/F')
tcre4at.Branch('Flux_C', Flux_C, 'Flux_C/F')

tcre4at.Branch('Burst_LocalMethod_A', Burst_LocalMethod_A,
'Burst_LocalMethod_A/I')
tcre4at.Branch('Burst_LocalMethod_B', Burst_LocalMethod_B,
'Burst_LocalMethod_B/I')
tcre4at.Branch('Burst_LocalMethod_C', Burst_LocalMethod_C,
'Burst_LocalMethod_C/I')

tcre4at.Branch('BurstForbushFactor_GlobalMethod_A',
BurstForbushFactor_GlobalMethod_A,
'BurstForbushFactor_GlobalMethod_A/F')
tcre4at.Branch('BurstForbushFactor_GlobalMethod_B',
BurstForbushFactor_GlobalMethod_B,
'BurstForbushFactor_GlobalMethod_B/F')
tcre4at.Branch('BurstForbushFactor_GlobalMethod_C',
BurstForbushFactor_GlobalMethod_C,
'BurstForbushFactor_GlobalMethod_C/F')

if(PIDAcquisition):
      tcre4at.Branch('PID_Sensor1', PID_Sensor1, 'PID_Sensor1/F')
      tcre4at.Branch('PID_Sensor2', PID_Sensor2, 'PID_Sensor2/F')
      tcre4at.Branch('PID_Avg', PID_Avg, 'PID_Avg/F')
      tcre4at.Branch('PID_Flag', PID_Flag, 'PID_Flag/i')

for line in tqdm(range(0, len(Data)), desc="Saving Data (ROOT File)"):
      Timestamp[0]            = np.array(Data['Timestamp'][line],
dtype=np.int64)
      GPS_Timestamp[0]  = np.array(Data['GPS_timestamp'][line],
dtype=np.int64)
      RTC_Timestamp[0]  = np.array(Data['RTC_timestamp'][line],
dtype=np.int64)

      ACh0[0]     = np.array(Data.ACh0[line], dtype=int)
      ACh1[0]     = np.array(Data.ACh1[line], dtype=int)
      ACh2[0]     = np.array(Data.ACh2[line], dtype=int)
      ACh3[0]     = np.array(Data.ACh3[line], dtype=int)
      BCh0[0]     = np.array(Data.BCh0[line], dtype=int)
      BCh1[0]     = np.array(Data.BCh1[line], dtype=int)
      BCh2[0]     = np.array(Data.BCh2[line], dtype=int)
      BCh3[0]     = np.array(Data.BCh3[line], dtype=int)
      CCh0[0]     = np.array(Data.CCh0[line], dtype=int)
      CCh1[0]     = np.array(Data.CCh1[line], dtype=int)
      CCh2[0]     = np.array(Data.CCh2[line], dtype=int)
      CCh3[0]     = np.array(Data.CCh3[line], dtype=int)

      A0and1[0] = np.array(Data['A0&1'][line], dtype=int)
      A0and2[0] = np.array(Data['A0&2'].values[line], dtype=int)
      A0and3[0] = np.array(Data['A0&3'].values[line], dtype=int)
      A1and2[0] = np.array(Data['A1&2'].values[line], dtype=int)
      A1and3[0] = np.array(Data['A1&3'].values[line], dtype=int)
      A2and3[0] = np.array(Data['A2&3'].values[line], dtype=int)
      B0and1[0] = np.array(Data['B0&1'].values[line], dtype=int)
      B0and2[0] = np.array(Data['B0&2'].values[line], dtype=int)
      B0and3[0] = np.array(Data['B0&3'].values[line], dtype=int)
      B1and2[0] = np.array(Data['B1&2'].values[line], dtype=int)
      B1and3[0] = np.array(Data['B1&3'].values[line], dtype=int)
      B2and3[0] = np.array(Data['B2&3'].values[line], dtype=int)
```

```
C0and1[0] = np.array(Data['C0&1'].values[line], dtype=int)
C0and2[0] = np.array(Data['C0&2'].values[line], dtype=int)
C0and3[0] = np.array(Data['C0&3'].values[line], dtype=int)
C1and2[0] = np.array(Data['C1&2'].values[line], dtype=int)
C1and3[0] = np.array(Data['C1&3'].values[line], dtype=int)
C2and3[0] = np.array(Data['C2&3'].values[line], dtype=int)

A0and1and2[0] = np.array(Data['A0&1&2'][line], dtype=int)
A0and1and3[0] = np.array(Data['A0&1&3'][line], dtype=int)
A0and2and3[0] = np.array(Data['A0&2&3'][line], dtype=int)
A1and2and3[0] = np.array(Data['A1&2&3'][line], dtype=int)
B0and1and2[0] = np.array(Data['B0&1&2'][line], dtype=int)
B0and1and3[0] = np.array(Data['B0&1&3'][line], dtype=int)
B0and2and3[0] = np.array(Data['B0&2&3'][line], dtype=int)
B1and2and3[0] = np.array(Data['B1&2&3'][line], dtype=int)
C0and1and2[0] = np.array(Data['C0&1&2'][line], dtype=int)
C0and1and3[0] = np.array(Data['C0&1&3'][line], dtype=int)
C0and2and3[0] = np.array(Data['C0&2&3'][line], dtype=int)
C1and2and3[0] = np.array(Data['C1&2&3'][line], dtype=int)

A0and1and2and3[0] = np.array(Data['A0&1&2&3'][line], dtype=int)
B0and1and2and3[0] = np.array(Data['B0&1&2&3'][line], dtype=int)
C0and1and2and3[0] = np.array(Data['C0&1&2&3'][line], dtype=int)

A01andB01[0] = np.array(Data['A01&B01'][line], dtype=int)
A01andB12[0] = np.array(Data['A01&B12'][line], dtype=int)
A01andB23[0] = np.array(Data['A01&B23'][line], dtype=int)
A12andB01[0] = np.array(Data['A12&B01'][line], dtype=int)
A12andB12[0] = np.array(Data['A12&B12'][line], dtype=int)
A12andB23[0] = np.array(Data['A12&B23'][line], dtype=int)
A23andB01[0] = np.array(Data['A23&B01'][line], dtype=int)
A23andB12[0] = np.array(Data['A23&B12'][line], dtype=int)
A23andB23[0] = np.array(Data['A23&B23'][line], dtype=int)
A01andC01[0] = np.array(Data['A01&C01'][line], dtype=int)
A01andC12[0] = np.array(Data['A01&C12'][line], dtype=int)
A01andC23[0] = np.array(Data['A01&C23'][line], dtype=int)
A12andC01[0] = np.array(Data['A12&C01'][line], dtype=int)
A12andC12[0] = np.array(Data['A12&C12'][line], dtype=int)
A12andC23[0] = np.array(Data['A12&C23'][line], dtype=int)
A23andC01[0] = np.array(Data['A23&C01'][line], dtype=int)
A23andC12[0] = np.array(Data['A23&C12'][line], dtype=int)
A23andC23[0] = np.array(Data['A23&C23'][line], dtype=int)
B01andC01[0] = np.array(Data['B01&C01'][line], dtype=int)
B01andC12[0] = np.array(Data['B01&C12'][line], dtype=int)
B01andC23[0] = np.array(Data['B01&C23'][line], dtype=int)
B12andC01[0] = np.array(Data['B12&C01'][line], dtype=int)
B12andC12[0] = np.array(Data['B12&C12'][line], dtype=int)
B12andC23[0] = np.array(Data['B12&C23'][line], dtype=int)
B23andC01[0] = np.array(Data['B23&C01'][line], dtype=int)
B23andC12[0] = np.array(Data['B23&C12'][line], dtype=int)
B23andC23[0] = np.array(Data['B23&C23'][line], dtype=int)

AccX[0] = np.array(Data['AccX'][line], dtype=float)
AccY[0] = np.array(Data['AccY'][line], dtype=float)
AccZ[0] = np.array(Data['AccZ'][line], dtype=float)
GyroX[0] = np.array(Data['GyroX'][line], dtype=float)
GyroY[0] = np.array(Data['GyroY'][line], dtype=float)
GyroZ[0] = np.array(Data['GyroZ'][line], dtype=float)
MagX[0] = np.array(Data['MagX'][line], dtype=float)
MagY[0] = np.array(Data['MagY'][line], dtype=float)
MagZ[0] = np.array(Data['MagZ'][line], dtype=float)
```

```
        Temp[0] = np.array(Data['Temp'][line], dtype=float)
        Hum[0] = np.array(Data['Hum'][line], dtype=float)
        Press[0] = np.array(Data['Press'][line], dtype=float)
        HV_Temp[0] = np.array(Data['HV_Temp'][line], dtype=float)
        HV_Voltage[0] = np.array(Data['HV_Voltage'][line], dtype=float)
        LED_CALIB_Status[0] = np.array(Data['LED_CALIB_Status'][line],
dtype=int)

        Eff_Raw_ACh0[0] = np.array(Data['Eff_Raw_ACh0'][line],
dtype=float)
        Eff_Raw_ACh1[0] = np.array(Data['Eff_Raw_ACh1'][line],
dtype=float)
        Eff_Raw_ACh2[0] = np.array(Data['Eff_Raw_ACh2'][line],
dtype=float)
        Eff_Raw_ACh3[0] = np.array(Data['Eff_Raw_ACh3'][line],
dtype=float)
        Eff_Raw_BCh0[0] = np.array(Data['Eff_Raw_BCh0'][line],
dtype=float)
        Eff_Raw_BCh1[0] = np.array(Data['Eff_Raw_BCh1'][line],
dtype=float)
        Eff_Raw_BCh2[0] = np.array(Data['Eff_Raw_BCh2'][line],
dtype=float)
        Eff_Raw_BCh3[0] = np.array(Data['Eff_Raw_BCh3'][line],
dtype=float)
        Eff_Raw_CCh0[0] = np.array(Data['Eff_Raw_CCh0'][line],
dtype=float)
        Eff_Raw_CCh1[0] = np.array(Data['Eff_Raw_CCh1'][line],
dtype=float)
        Eff_Raw_CCh2[0] = np.array(Data['Eff_Raw_CCh2'][line],
dtype=float)
        Eff_Raw_CCh3[0] = np.array(Data['Eff_Raw_CCh3'][line],
dtype=float)

        Flux_Raw_A[0] = np.array(Data['Flux_Raw_A'][line], dtype=float)
        Flux_Raw_B[0] = np.array(Data['Flux_Raw_B'][line], dtype=float)
        Flux_Raw_C[0] = np.array(Data['Flux_Raw_C'][line], dtype=float)

        Flux_A[0] = np.array(Data['Flux_A'][line], dtype=float)
        Flux_B[0] = np.array(Data['Flux_B'][line], dtype=float)
        Flux_C[0] = np.array(Data['Flux_C'][line], dtype=float)

        Burst_LocalMethod_A[0] =
np.array(Data['Det_A_Burst_Counter'][line], dtype=int)
        Burst_LocalMethod_B[0] =
np.array(Data['Det_B_Burst_Counter'][line], dtype=int)
        Burst_LocalMethod_C[0] =
np.array(Data['Det_C_Burst_Counter'][line], dtype=int)

        BurstForbushFactor_GlobalMethod_A[0] =
np.array(Data['A_Burst_Forbush_Factor_GlobalMethod'][line],
dtype=float)
        BurstForbushFactor_GlobalMethod_B[0] =
np.array(Data['B_Burst_Forbush_Factor_GlobalMethod'][line],
dtype=float)
        BurstForbushFactor_GlobalMethod_C[0] =
np.array(Data['C_Burst_Forbush_Factor_GlobalMethod'][line],
dtype=float)

        if(PIDAcquisition):
```

```
            PID_Sensor1[0]    = -10000 if
(np.isnan(Data['PID_Sensor1'][line])) else
np.array(Data['PID_Sensor1'][line], dtype=float)
            PID_Sensor2[0]    = -10000 if
(np.isnan(Data['PID_Sensor2'][line])) else
np.array(Data['PID_Sensor2'][line], dtype=float)
            PID_Avg[0]        = -10000 if
(np.isnan(Data['PID_Avg'][line])) else np.array(Data['PID_Avg'][line],
dtype=float)
            PID_Flag[0]       = 0 if (np.isnan(Data['PID_Flag'][line]))
else np.array(Data['PID_Flag'][line], dtype=int)

      tcre4at.Fill()
fROOT.Write()
fROOT.Close()

print("\n\n\nZenital Nfactor found by Simple Method:
%.2f\n"%(ZAfFound))
print("\n\n\nZenital Nfactor found by Fit: %.4f\n"%(ZAf))
print("Daily Efficiencies Corrected Found:\n\tACh0: %.2f\n\tACh1:
%.2f\n\tACh2: %.2f\n\tACh3: %.2f\n\tBCh0: %.2f\n\tBCh1: %.2f\n\tBCh2:
%.2f\n\tBCh3: %.2f\n\tCCh0: %.2f\n\tCCh1: %.2f\n\tCCh2: %.2f\n\tCCh3:
%.2f\n"%(DailyEff[0][0]*EffCorrection[0],DailyEff[0][1]*EffCorrection[
1],DailyEff[0][2]*EffCorrection[2],DailyEff[0][3]*EffCorrection[3],Dai
lyEff[1][0]*EffCorrection[0],DailyEff[1][1]*EffCorrection[1],DailyEff[
1][2]*EffCorrection[2],DailyEff[1][3]*EffCorrection[3],DailyEff[2][0]*
EffCorrection[0],DailyEff[2][1]*EffCorrection[1],DailyEff[2][2]*EffCor
rection[2],DailyEff[2][3]*EffCorrection[3]))
print("Daily Flux Corrected Found:\n\tA: %.5f +- %.5f\n\tB: %.2f +-
%.5f\n\tC: %.2f +- %.5f\n"%(DailyFluxCorrected[0],
DailyFluxCorrectedErr[0], DailyFluxCorrected[1],
DailyFluxCorrectedErr[1], DailyFluxCorrected[2],
DailyFluxCorrectedErr[2]))
if (SaveOverallValues):
      if(os.path.isfile(args.TextAppendDailyFile)):
            f = open(args.TextAppendDailyFile, "r")
            filecontent = f.readlines()
            f.close()
            if(any(str(Data.Timestamp[0]) in line for line in
filecontent)):
                  pos = np.where(list(str(Data.Timestamp[0]) in
mystring for mystring in filecontent))[0][0]
                  filecontent[pos] =
"%d\t%.2f\t%.4f\t%.8f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\
t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4
f\t%.2f\t%.4f\t%.2f\t%.4f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\n"%(Data
.Timestamp[0],ZAfFound,ZAf,ZAfErr,DailyEff[0][0]*EffCorrection[0],Dail
yEffRms[0][0]*EffCorrection[0],DailyEff[0][1]*EffCorrection[1],DailyEf
fRms[0][1]*EffCorrection[1],DailyEff[0][2]*EffCorrection[2],DailyEffRm
s[0][2]*EffCorrection[2],DailyEff[0][3]*EffCorrection[3],DailyEffRms[0
][3]*EffCorrection[3],DailyEff[1][0]*EffCorrection[0],DailyEffRms[1][0
]*EffCorrection[0],DailyEff[1][1]*EffCorrection[1],DailyEffRms[1][1]*E
ffCorrection[1],DailyEff[1][2]*EffCorrection[2],DailyEffRms[1][2]*EffC
orrection[2],DailyEff[1][3]*EffCorrection[3],DailyEffRms[1][3]*EffCorr
ection[3],DailyEff[2][0]*EffCorrection[0],DailyEffRms[2][0]*EffCorrect
ion[0],DailyEff[2][1]*EffCorrection[1],DailyEffRms[2][1]*EffCorrection
[1],DailyEff[2][2]*EffCorrection[2],DailyEffRms[2][2]*EffCorrection[2]
,DailyEff[2][3]*EffCorrection[3],DailyEffRms[2][3]*EffCorrection[3],Da
ilyFluxCorrected[0], DailyFluxCorrectedErr[0],DailyFluxCorrected[1],
DailyFluxCorrectedErr[1], DailyFluxCorrected[2],
DailyFluxCorrectedErr[2])
```

```
                f = open(args.TextAppendDailyFile, "w")
                for line in filecontent:
                      f.write(line)
                f.close()
          else:
                f = open(args.TextAppendDailyFile, "a")

      f.write("%d\t%.2f\t%.4f\t%.8f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4
f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%
.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\
t%.5f\n"%(Data.Timestamp[0],ZAfFound,ZAf,ZAfErr,DailyEff[0][0]*EffCorr
ection[0],DailyEffRms[0][0]*EffCorrection[0],DailyEff[0][1]*EffCorrect
ion[1],DailyEffRms[0][1]*EffCorrection[1],DailyEff[0][2]*EffCorrection
[2],DailyEffRms[0][2]*EffCorrection[2],DailyEff[0][3]*EffCorrection[3]
,DailyEffRms[0][3]*EffCorrection[3],DailyEff[1][0]*EffCorrection[0],Da
ilyEffRms[1][0]*EffCorrection[0],DailyEff[1][1]*EffCorrection[1],Daily
EffRms[1][1]*EffCorrection[1],DailyEff[1][2]*EffCorrection[2],DailyEff
Rms[1][2]*EffCorrection[2],DailyEff[1][3]*EffCorrection[3],DailyEffRms
[1][3]*EffCorrection[3],DailyEff[2][0]*EffCorrection[0],DailyEffRms[2]
[0]*EffCorrection[0],DailyEff[2][1]*EffCorrection[1],DailyEffRms[2][1]
*EffCorrection[1],DailyEff[2][2]*EffCorrection[2],DailyEffRms[2][2]*Ef
fCorrection[2],DailyEff[2][3]*EffCorrection[3],DailyEffRms[2][3]*EffCo
rrection[3],DailyFluxCorrected[0],
DailyFluxCorrectedErr[0],DailyFluxCorrected[1],
DailyFluxCorrectedErr[1], DailyFluxCorrected[2],
DailyFluxCorrectedErr[2]))
                f.close()
      else:
            f = open(args.TextAppendDailyFile, "w")

      f.write("Timestamp\tZAF\tZAFfit\tZAFfitErr\tEffA0\tEffA0std\tEff
A1\tEffA1std\tEffA2\tEffA2std\tEffA3\tEffA3std\tEffB0\tEffB0std\tEffB1
\tEffB1std\tEffB2\tEffB2std\tEffB3\tEffB3std\tEffC0\tEffC0std\tEffC1\t
EffC1std\tEffC2\tEffC2std\tEffC3\tEffC3std\tFluxA\tFluxAstd\tFluxB\tFl
uxBstd\tFluxC\tFluxCstd\n")

      f.write("%d\t%.2f\t%.4f\t%.8f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4
f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%
.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.2f\t%.4f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\
t%.5f\n"%(Data.Timestamp[0],ZAfFound,ZAf,ZAfErr,DailyEff[0][0]*EffCorr
ection[0],DailyEffRms[0][0]*EffCorrection[0],DailyEff[0][1]*EffCorrect
ion[1],DailyEffRms[0][1]*EffCorrection[1],DailyEff[0][2]*EffCorrection
[2],DailyEffRms[0][2]*EffCorrection[2],DailyEff[0][3]*EffCorrection[3]
,DailyEffRms[0][3]*EffCorrection[3],DailyEff[1][0]*EffCorrection[0],Da
ilyEffRms[1][0]*EffCorrection[0],DailyEff[1][1]*EffCorrection[1],Daily
EffRms[1][1]*EffCorrection[1],DailyEff[1][2]*EffCorrection[2],DailyEff
Rms[1][2]*EffCorrection[2],DailyEff[1][3]*EffCorrection[3],DailyEffRms
[1][3]*EffCorrection[3],DailyEff[2][0]*EffCorrection[0],DailyEffRms[2]
[0]*EffCorrection[0],DailyEff[2][1]*EffCorrection[1],DailyEffRms[2][1]
*EffCorrection[1],DailyEff[2][2]*EffCorrection[2],DailyEffRms[2][2]*Ef
fCorrection[2],DailyEff[2][3]*EffCorrection[3],DailyEffRms[2][3]*EffCo
rrection[3],DailyFluxCorrected[0],
DailyFluxCorrectedErr[0],DailyFluxCorrected[1],
DailyFluxCorrectedErr[1], DailyFluxCorrected[2],
DailyFluxCorrectedErr[2]))
            f.close()
```

Apêndice R

Manual de colagem do parafuso óptico

Este apêndice contém o manual de instruções para realizar a colagem da fibra *Wavelength Shifter* ao parafuso óptico.

Centro Brasileiro de Pesquisas Físicas
COHEP

Manual para Processo de Colagem das Fibras Wavelength Shifter

Diogo Ayres Rocha

Rio de Janeiro
2020

Preparação da Cola

A cola óptica é formada por 2 componentes que, quando misturados na proporção correta, farão a união da fibra *wavelength shifter* (WSL) ao parafuso ótico. Os dois compostos da cola podem ser vistos abaixo na figura 1.

Figura 1: Cola Óptica

O composto do frasco maior possui uma densidade maior, por isso, ao usar a seringa para retirar a quantidade necessária, não insira a agulha. A proporção utilizada será de 1/3 do componente do frasco menor e 2/3 do franco maior. Em torno de 5 a 6 parafusos poderão ser colados com a menor quantidade de cola que é possível misturar, em virtude da menor unidade disponível na seringa que pode ser utilizada. Não esqueça de deixar perto algumas folhas de papel toalha para limpeza rápida em caso de respingos de cola. **Não esqueça de utilizar luva durante o processo.**

O processo pode iniciar por qualquer um dos componentes e, para fim desse manual, o primeiro componente será o de menor densidade:

1. Com o auxilio da seringa com a agulha, insira a menor medida disponível do composto como pode ser visto na figura 2.

2. Despeje o composto no recipiente pequeno disponível como o mostrado na figura 3.

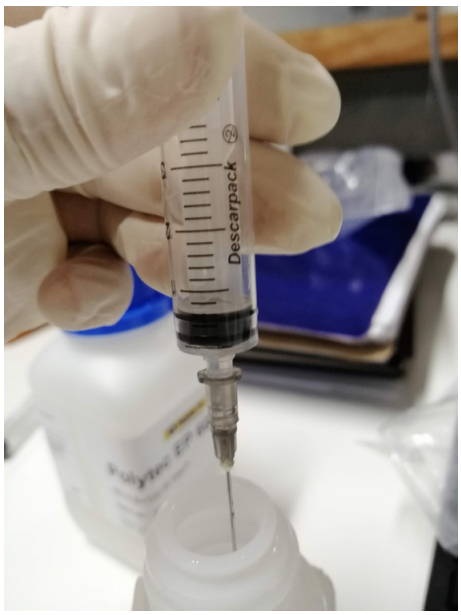


Figura 2: Menor medida do componente 1

Figura 3: recipiente pequeno com primeiro componente da cola

3. Utilizando outra seringa disponível, como já explicado, sem uso da agulha, succione duas vezes o nível utilizado no outro composto como pode ser visto na figura 4.

4. Utilize a própria ponta da seringa para mexer e uniformizar a mistura da cola, observando a figura 5.

5. Utilize uma nova seringa com a agulha para maior precisão ao posicionar a cola no parafuso. Contudo, observe que a cola será bem densa, com isso, pode-se succionar a cola com a seringa sem a agulha e colocá-la após isso.

Figura 4: Medida do componente 2 (2 vezes o primeiro composto)

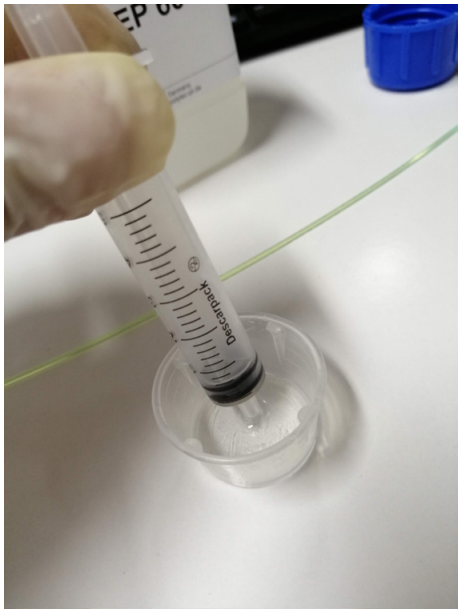


Figura 5: Mistura da cola com a seringa

Com a seringa pronta para colocar a cola, será a hora de preparar as fibras e parafusos para recebem-na, como será explicado na próxima sessão.

Aplicando a Cola

Para preparar a fibra será necessário um pedaço de papel e fita.

1. Primeiramente corte um pedaço pequeno de papel e enrole em volta da fibra deixando o mais justo possível, observando a figura 6. Esse papel será importante para não deixar a cola escorrer pela fibra e para retirar a cola residual após o processo de secagem.

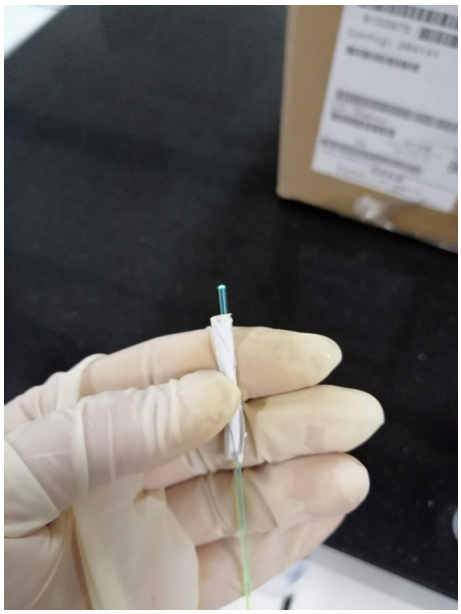


Figura 6: Preparando a fibra WSL com o papel

2. Insira o parafuso até tocar no papel e passe uma fita para fixar o papel à rosca do parafuso. A fibra não precisa ultrapassar o parafuso, pois ainda será possível escorregá-la através dele. Veja a figura 7.

Figura 7: Fibra com parafuso preso ao papel

3. Deixe a fibra um pouco abaixo do furo e com o auxílio da seringa injete um pouco de cola dentro do orifício, deixando passar um pouco para região superior. Observe a figura 8.

Figura 8: Parafuso pronto para receber a cola

4. Após o posicionamento da cola, empurre levemente a fibra deixando-a passar do nível da cabeça do parafuso alguns milímetros (em torno de 8mm), como pode ser visto na figura 9.

Figura 9: Parafuso com cola e fibra passando

5. Puxe levemente a fibra para baixo novamente e esse processo irá trazer um pouco de cola para o interior do parafuso uniformizando melhor o processo de colagem. Isso deixará a superfície lisa ao retirar o excesso de cola após a usinagem. Deixe aproximadamente 2mm de fibra além da cabeça do parafuso, que será eliminada com a usinagem.

6. Ao prender o parafuso para deixar a cola secando, deixe-o em pé para evitar que a cola escorra. No caso, foi utilizado um pequeno torno mecânico do laboratório, como pode ser visto na figura 10.

Figura 10: Parafuso preso no torno em pé para evitar que a cola escorresse

7. Para evitar que a fibra escorregue, faça pressão contra alguma superfície no restante da fibra que está para baixo. Nesse caso, foi utilizado uma fita isolante para prender à mesa, porém, para não danificar a fibra e/ou deixar cola nela, na região onde a fita tocaria na fibra foi inserido um pedaço de papel. Veja a figura 11.

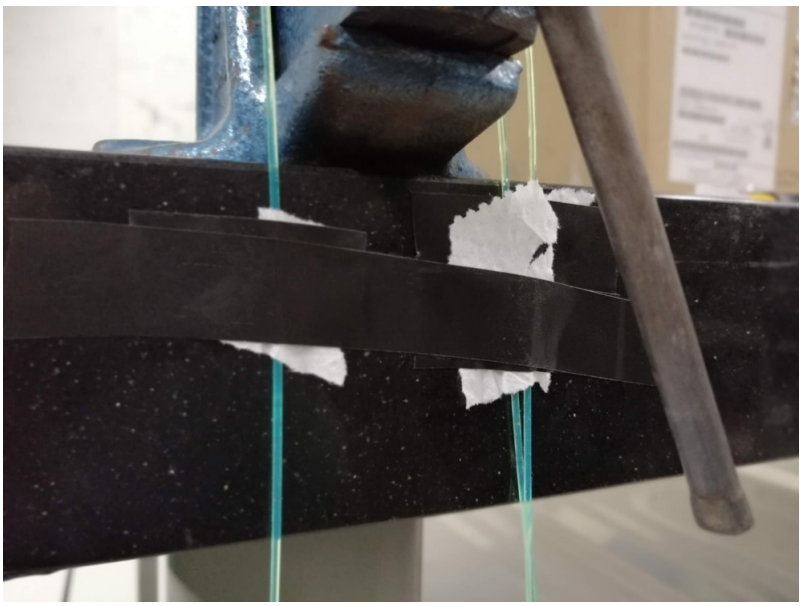


Figura 11: Fibra pressionada à mesa com fita isolante e papel para protegê-la da cola

8. Ao final do processo de secagem, que dura em torno de 24h, retire a fita que prende o papel ao parafuso e, com cuidado, rotacione o papel segurando no parafuso. Isso irá eliminar o resto de cola que escorreu para baixo deixando a fibra limpa.

Não esqueça de lavar as seringas e o recipiente da cola logo após sua utilização, para evitar que a cola seque e perda o material.

Apêndice S

Nota Técnica SciFi

Este apêndice contém a nota técnica criada como resultado do procedimento de teste automatizado, desenvolvido para a validação das novas eletrônicas do sistema de traço com tecnologia de fibras cintilantes, SciFi (LHCb/CERN), após a produção em massa.

FEE Test System

Read-Out-Box Test System Protocol for SciFi detector

Technical Note

CERN-LHCb-INT-2024-004

André Massafferri Rodrigues, Antônio Pellegrino, Daniel Berninghoff, Diogo Ayres Rocha, Ulisses Carneiro, Wilco Vink, Xiaoxue Han, Lucas Falcão

February 15, 2024

1 Introduction

After mass production, the 280 read-out front-end boxes (RoB) foreseen in SciFi detector need to undergo a process of quality assurance before installation and commissioning. The goal of the automatic test system is to fully qualify the overall individual RoBs, by generating the diagnostic in the format of a text file (logfile), a pdf file containing the main set of plots (summary) as well as uploading the full data in the format of root-file and relevant parameters in the SciFi Production DataBase (ProdDb).

The RoB is a unit device with 2048 input channels designed to read the SiPM signals of the SciFi module detector. It is composed of two independent Half RoBs, housing one Master board (MB), four Cluster boards (CB), and four PACIFIC boards (PB) with 256 input channels. The main component of PB is 64 channels ASIC named PACIFIC. The chip is responsible for amplification and digitization, as well as provides 8-bit DACs for precise trimming of the SiPM high-voltage and the analog threshold for the comparators. The CB contains two Microsemi FPGAs responsible for 128 channels each, and one slow control adapter (SCA) ASIC. The MB is composed of four DataGBT ASICs responsible for data transmission, one MasterGBT as a control interface, one SCA ASIC, DC-DC power provided by FeastMPs, and one Housekeeping FPGA. It is connected to the data acquisition system through eight unidirectional data links and one bidirectional control link. For the slow control, the RoB contains ten SCA ASICs, each one with one SPI, 16 I^2C, and 32 GPIO lines, four 8-bit DACs, one JTag chain, and one 12-bit ADC with 32 multiplexed inputs, to allow voltage measurements with a designed 100 μA current source.

For data acquisition and control back-end, this project uses a small unit, named miniDAQ - basically the same device developed for the LHCb standard of Upgrade I - based on a PCIe40 card mounted on a server PC. That device contains an Arria 10 GX FPGA (10AX115S3F45E2SG) with 1.15 million cells that emulate the CERN GBTx data frame through 48 bidirectional optical links running at 4.8 Gbps allowing high-speed data acquisition and full control of the RoB.

The Test Bench comprises one injection system, specially designed for this project, the RoB under test, one miniDAQ for data acquisition with monitor/keyboard/mouse, and requires one high precision high voltage module and a 4-channels@10A power supply to provide two lines of +12 V for the injection system and two lines for +6.5 V for the RoB. The injection system contains one control board which is connected to the miniDAQ by one TX/RX link and eight injector modules with 256 channels each, able to individually inject calibrated charge pulse in each one of the 2048 RoB inputs through the standard SiPM connectors. This high-density connector has a maximum number of insertions equal to 50, much less than the total number of RoBs to be tested. To enhance the lifetime of this connector, a low-cost and disposable adapter board was added between the RoB and injector connectors. The injector system proved to have a high uniformity concerning the timing arrival of the pulses generated, allowing to perform a precise alignment employing a single delay parameter with 2.5 ns LSB resolution.

The Test procedure is fully automatized by the WINCC control panel, which can be used by non-experts.

This document intends to describe the test sequence and its control panel and provide practical

information for the user on the preparation and finalization of the test procedure concerning RoB handling.

2 The RoB Test Sequence

The RoB Test Procedure is divided into steps, described below. It is also included, within boxes, the criteria used as `Error` condition.

2.1 Download PB configuration Files and RoB IDs

The first step is to download the configuration files from ProdDb, specific for each PB, and get the production and Hardware IDs for all MB, CB, and PB on the RoB under test.

2.2 Current Verification I

Verification of the RoB current before configuration, measured on the Power Supply used in the Test procedure.

`Error` condition
RoB current $<$ 350mA

2.3 Optical Control Link syncronization

Before the configuration, the error lock counters on miniDAQ firmware are reset. After the RoB configuration (described in the next step), the counters will check if any errors are found on the control fibers.

2.4 RoB configuration

In the MB, we configure the MasterGBT, DataGBTs and MasterSCA, as well as enable the FeastMP to power CB and PB. In the CB, we configure the SCA.

2.5 Check LEDs status

After the successful configuration, the software verifies the RoB LEDs status, searching for any configuration problem on DataGBTs.

2.6 Light Power Verification of Optical Fiber Links

Verification of the light intensity of the optical links.

`Error` condition
Light intensity criteria $<$ 255 μW

2.7 House-keeping FPGA

The procedure loads the firmware of the two FPGAs located in MBs, verifies the read/write dummy registers, and records the firmware version.

2.8 Cluster Board FPGA

The procedure loads the firmware of sixteen FPGAs located in CBs, verifies the read/write dummy registers, and records the firmware version.

2.9 RoB configuration

After the firmware of HK and CB FPGAs is loaded successfully, the configuration process is repeated to properly configure the PACIFICs ASICs on PB.

2.10 Optical Data Link syncronization

Checks if all DataLinks are properly locked. The process is the same as the control links.

2.11 Current Verification II and Voltage

Reads the RoB current after configuration and loading firmware. Also, it reads the voltages on the MB SCA ADCs and the power supply, stored in the logfile.

`Error` condition
RoB current < 7A

2.12 IDs verification

Reads the IDs of the MasterGBT as a way to unequivocally identify the MB. Upload the ID of the MasterSCA of the MB to the ProdDb.

This step compares the IDs of the CBs SCA with the ones written in the ProdDb.

The unique PB ID is hard-written on the chip. Through one-wire protocol in CB FPGA this step reads the intrinsical ID of the PBs to be stored on the ProdDb and compares with the ones written in the ProdDb[1].

2.13 Current Source Calibration

To keep the SiPM dark current rate under 15 MHz during the data-taking, due to the expected high neutron radiation conditions, the SiPM must be cooled down to -40 ^{0}C. For monitoring purposes, each SiPM connector has a PT1000 temperature sensor connected in series to a current source and read by an ADC on an SCA chip. In this step, we determine the current values produced by the

[1]The procedure is necessary to fix the ordering problem that occurred during the PB commitioning

eight current sources, utilizing a 0.01% precision 4.7k ohms resistor located in the connector-saver board (Figure 1). The current of the two connector-savers at the same PB is averaged and uploaded to ProdDb, see Fig. 2.

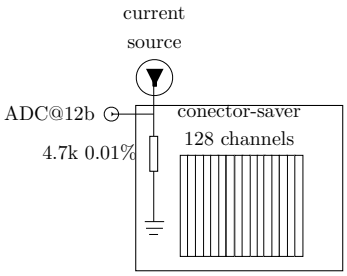


Figure 1: Representation of the current source in connector-saver.

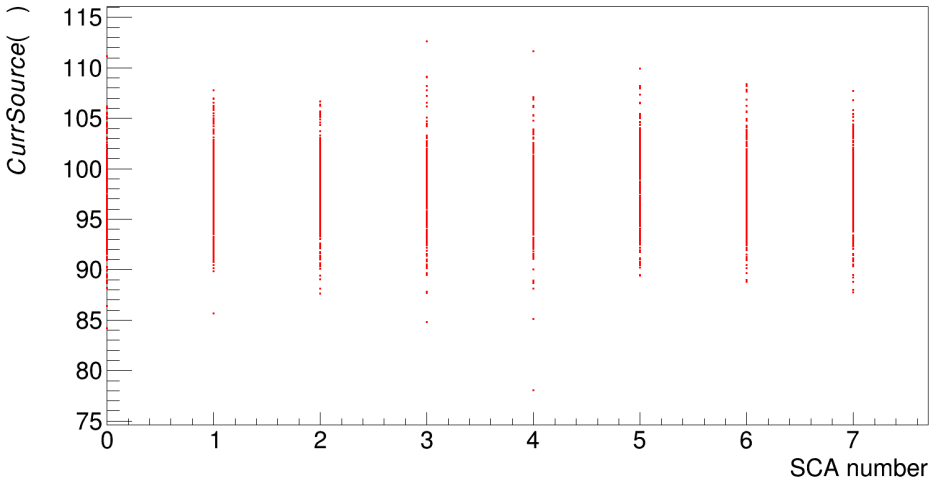


Figure 2: Current source plot of all channels for all RoBs tested.

`Error` condition
85 μA > current source > 110 μA

2.14 Temperature measurements

The same mechanism of the previous item is used to perform other temperature measurements. Once the current source values are determined, the test measures the temperatures of 56 sensors installed in the MBs, 16 sensors; CBs, 24 sensors; and PBs, 16 sensors.

`Error` condition
20 ^{0}C > Temperature > 45 ^{0}C
RMS between sensors > 5 ^{0}C

2.15 Calibration of HV measurement

Each HalfRoB receives 16 HV lines from an external HV power supply and the HV bias voltage of SIPM channels can be individually fine-tuned by a TrimDAC located in PB. During the data-taking, the HV bias will be monitored by reading the multiplexed channels of the ADC value in the SCA chip after a voltage divider, as shown in Figure 3. For calibration purposes, this step performs a linear fit of voltage measured in ADC according to the voltage applied by a high-precision power supplier in the range from 5 V to 70 V with 1 V step, averaging 20 times. Although the theoretical value of $V_{\text{bias}} = (72.43 \pm 0.09)$ * V_{ADC}, we modeled the fit result as a 1^{st} degree polynomial function (2 parameters) to account for ∼mV variations due grounding fluctuations, see Fig. 4 and 5. Both parameters are being saved in the database to be compared to further calibrations.

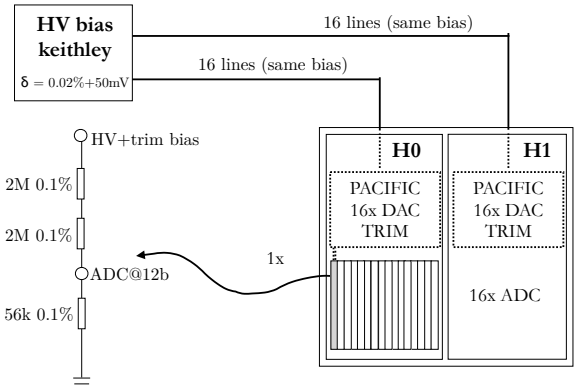


Figure 3: Block diagram of the relevant parts of the HV procedure.

2.16 Threshold-Scan (ThScan) Test

Two ThScan sequences in the full 8-bit DAC range are performed simultaneously in all 2048 channels, to determine the pedestal, measure indirectly the charge injected, and estimate the noise of every single channel. This procedure consists of a powerful tool to search for dead and noisy channels as well as to spot differences between the trim DAC settings of the two PACIFIC integrators. In this procedure, for each threshold value, we acquire data sequentially in 10k orbits, emulated

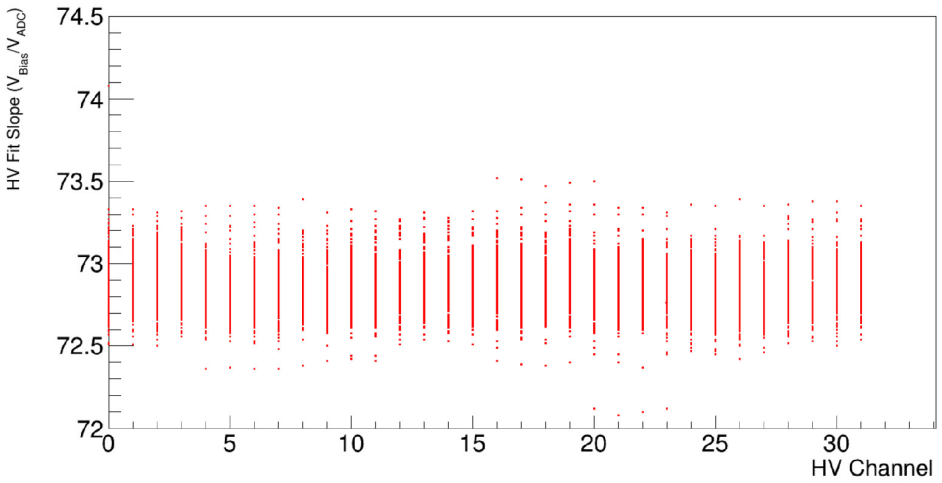


Figure 4: HV Fit Slope plot of all channels for all RoBs tested.

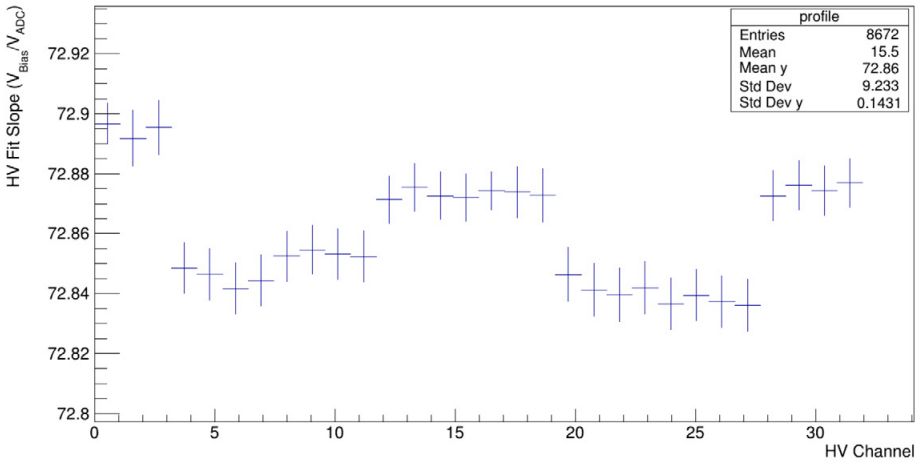


Figure 5: Profile of the HV Fit Slope plot of all channels for all RoBs tested.

in miniDAQ, in four different BXIDs sectors[2], to count pulses by using both PACIFIC integrators (even and odd BXIDs) in two conditions - with and without charge injection.

Both ThScan sequences differ only by the charge injected, previously calibrated. As a default, we use Q_{calib} equal to 0.1895 pC for ThScan1 and 0.43591 pC for ThScan2, both with 5 ns width. Therefore this Test produces eight S-curves for each one of the 2048 channels. The relevant transitions of S-curves, with and without charge, are, by construction, well detached. The results are spotted by merging/normalizing them, reducing by a factor of two the number of graphs. The

[2]In this test, DAQ is running in RAW mode, instead of the CLUSTER mode. Since in this case, the data frame houses 32 channels and one datalink is responsible for 128 channels, each sector must be composed of 5 consecutive BXIDs, the first one for CLUSTER mode data (default) and the others for the relevant data effectively needed for the test.

relevant numbers related to this procedure are shown in Table 1.

Table 1: BXIDs used in the orbits and default charge injected in each ThScans. The injector boards must be triggered 12 BXIDs prior to the acquisition.

(double) S-curve	integrator	pedestal BXID range	injection BXID range	injection charge(pC)
ThScan 1				
1	0	1000-1004	82-86	0.1895
2	1	3001-3005	2083-2087	0.1895
ThScan 2				
3	0	1000-1004	82-86	0.43591
4	1	3001-3005	2083-2087	0.43591

The four (double) S-curves are composed of six points of interest, as illustrated in Figure 6. The transition points in the x-axis are floating values in DAC units since they are obtained by the linear interpolations in the vicinity of the transition. The point `B` (75% crossing) corresponds to the threshold value in the absence of any external charge in the input of the RoB, known as the pedestal value. This value is a characteristic of the channel and it can vary subtly after power cycles. The point `E` (25% crossing), in the second S-curve, corresponds to the pedestal plus injection value, therefore the difference between `B` and `E` points corresponds to the charge effectively injected, Q_{inj}. The difference between `A` (99% crossing) and `C` (51% crossing) points is proportional to the noise produced by the RoB channel, called here Equivalent-Noise-like (ENL)[3]. The difference between `D` (49% crossing) and `F` (1% crossing) corresponds to the ENL in which the noise produced by the injector module is added. When a given transition is not found, the default value equals to -1 is set.

Figure 7 shows the distributions of the pedestal, Q_{inj} and ENL(pedestal), of ThScan 1, for integrator 0 of the RoB `4TSLPCEFB00069`. In this example, all channels are working as expected.

To be able to search for defects in the RoBs, it is important to estimate the uncertainty of the method. The most important contributions are the finite statistics (10k countings), the interpolation technique to extract values from S-curves, and the charge reproducibility of the injector modules in a given channel. For this purpose, we subtract the values obtained for pedestal, Q_{inj} and ENL extracted in two consecutive and identical tests, separated by one power cycle of the same RoB. The results obtained for the RoB `4TSLPCEFB00234` are shown in Figure 8.

In the Table 2 the `Error`, `Warning` and `Info` criteria are presented, followed by comments.

1. A dead channel is defined as a channel without a pedestal and charge. To embrace other critical cases we defined an absence of charge as half of the expected charge injected.
2. An open channel is the set of channels in which the pedestal is well determined but no charge is seen, see Figure 9.

[3]Since it is not obtained from any fit procedure, modeled by the Error function, ENL corresponds to approximately 3 times the usual *equivalent-noise in electronics textbooks.*

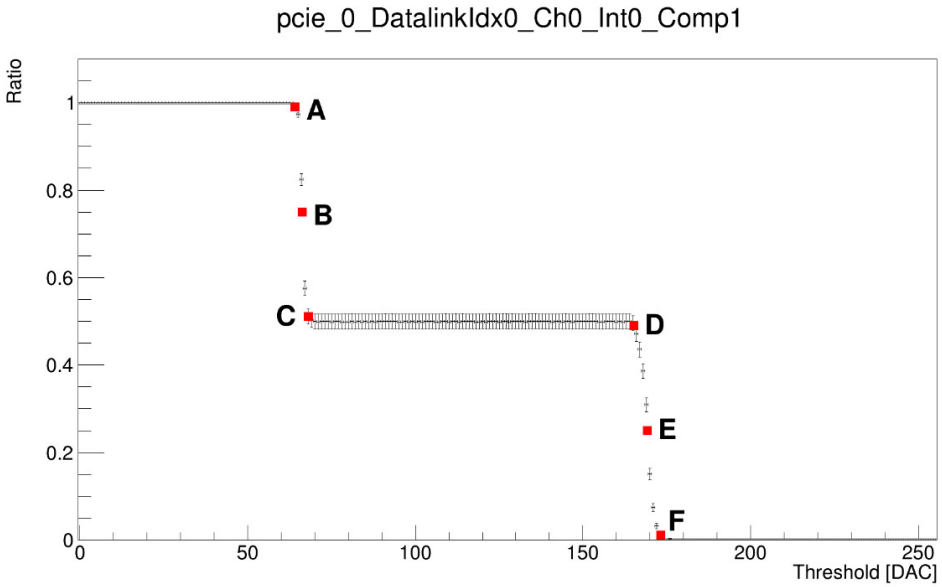


Figure 6: Four S-curves, six points of interest; B is the pedestal, E is the pedestal + charge. Q_{inj} measures effectively injected charge, channel pedestal varies subtly after power cycles.

3. This alarm is set to spot channels with pedestals outside the limits.
4. This alarm is set when only the pedestal has been identified.
5. The CB makes use of an FPGA to perform digital processing to form clusters from a single SiPM array. It computes the barycenter of the photon hits, improving the intrinsic spatial resolution of the geometrical layout of the sensors. The algorithm starts looking for a seed with more than 2.5 photo-electrons signal, neighbored by another channel with more than 1.5 photo-electrons. A single channel cluster can also be formed if the signal surpasses 4.5 photo-electrons. The commissioning with a light-injection system shows that 1.5 photo-electrons corresponds to approximately 15 DAC units, measured from the pedestal. Since the ENL is evaluated from the two transitions, the error criteria applied corresponds to half of the minimum value required by the experiment.
6. This warning is important to spot that the noise is outside the expected range.
7. This warning is important to spot any discontinuity between two HalfRoBs.
8. This criterion informs if abnormal behavior of the noise measurement is detected.
9. In principle, the pedestal between both integrators of the same channel should be identical. Any difference, greater than the precision of the measurement, must be informed.

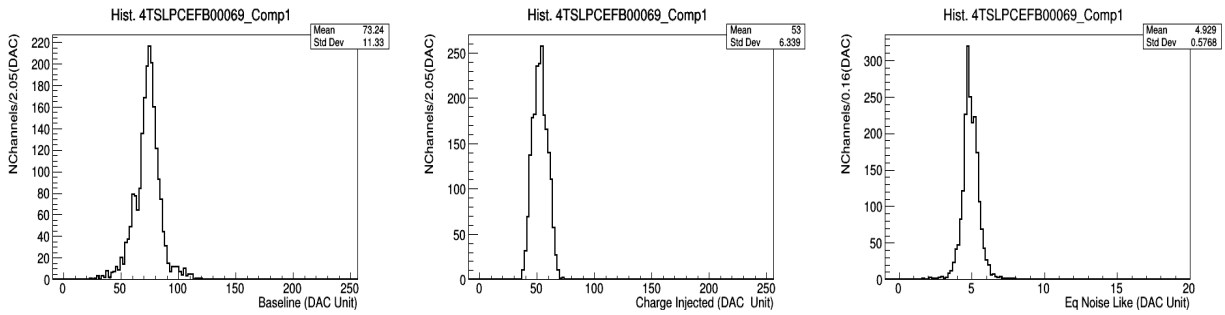


Figure 7: From left to right, distributions of (a) pedestal, (b) Q_{inj} and (c) ENL(pedestal), of ThScan 1, for integrator 0 of the RoB `4TSLPCEFB00069`.

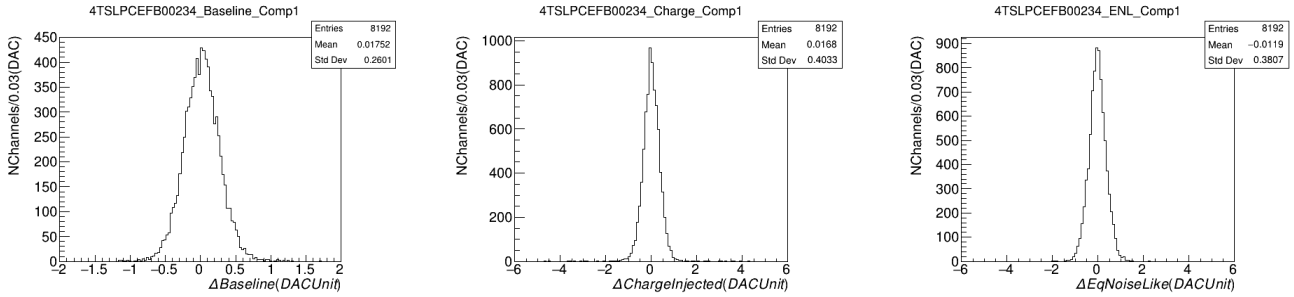


Figure 8: From left to right, distributions of the subtraction of the two consecutive tests for (a) pedestal, (b) Q_{inj} and (c) ENL(pedestal) of the RoB `4TSLPCEFB00234`.

2.17 Short-Circuit (SHCirc) Test

The scope of this test is to detect short circuits and crosstalks between neighbored channels, defined as a block of 128 channels that share the same SiPM connector as well as two PB ASICs. Even and odd of these channels are outputted from two different PACIFIC ASICs. In this sense, with the knowledge of the pedestal and ENL of all channels, in this step, 10k pulses with 0.43591 pC are injected in a given channel while all channels are counted. This procedure is performed consecutively in 128 channels within each block and simultaneously in the 16 blocks of the RoB. The threshold value used as default is 1.5 times the ENL found, with a limit of 10 DAC units. Figure 10 shows the short-circuit result for the RoB`4TSLPCEFB00208` without problems, for SiPM connector 0 and injection in channel 63.

`Error` condition
rate @injected channel < 0.95
short-circuit: rate @neighbor channel > 0.6
cross-talk: rate @neighbor channel > 0.1

Table 2: `Error`, `Warning` and `Info` criteria used in this step.

item	message	`Error` condition
1	dead channel	pedestal ≤ 0 and $Q_{meas} < (Q_{calib} / 2)$
2	open channel	pedestal > 0 and $Q_{meas} < (Q_{calib} / 2)$
3	low pedestal	pedestal = -1 and $Q_{meas} \neq$ -1
4	transition not found	pedestal $\neq$ -1 and Q_{meas} = -1
5	noise too high	ENL(pedestal) > 15
	`Warning` condition	
6		$15 >$ ENL(pedestal) > 9
7		| (mean Q_{meas}@H0) - (mean Q_{meas}@H1) | > 10
	`info` condition	
8		ENL(pedestal) $>$ ENL(pedestal+injection)
9		|pedestal $_{int0}$ - pedestal $_{int1}$ | > 1.5
	All values above are in DAC units	

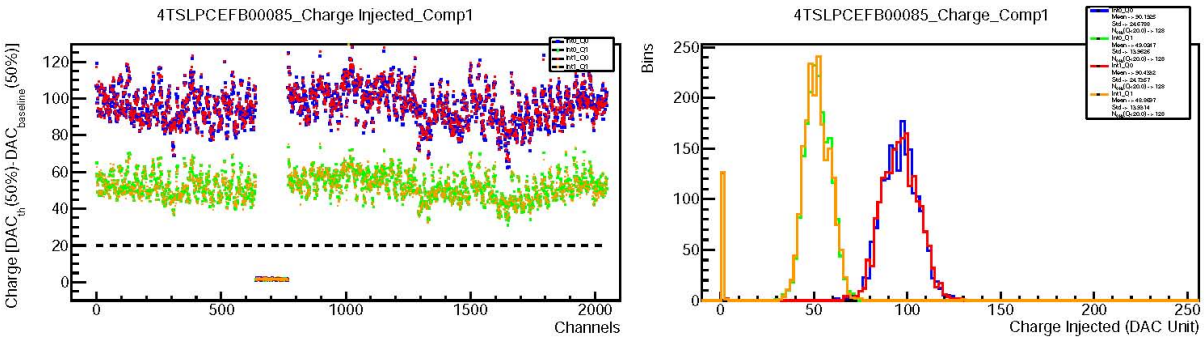


Figure 9: Group of 128 Open Channels on RoB 4TSLPCEFB00085

2.18 Link Stability Test

The system is composed of control and data links connected with miniDAQ using the GBTx Asic located at the Master Board, which has a communication error counter capability implemented, allowing the check any lost of clock. The goal of the Link Stability test is to verify the communication error due to clock failure on the control links, by simple comparison of the initial random values of the counters. The value is read out periodically during the test procedure.

`Error` condition
Any value different from initial
Execution every 1 hour and at the end of the procedure

2.19 PACIFIC Bit Error Rate (BER)

To verify the proper communication between the PACIFIC output and the back-end, the PACIFIC BER test is implemented by sending a known pattern sequence in a specific path. The path starts in

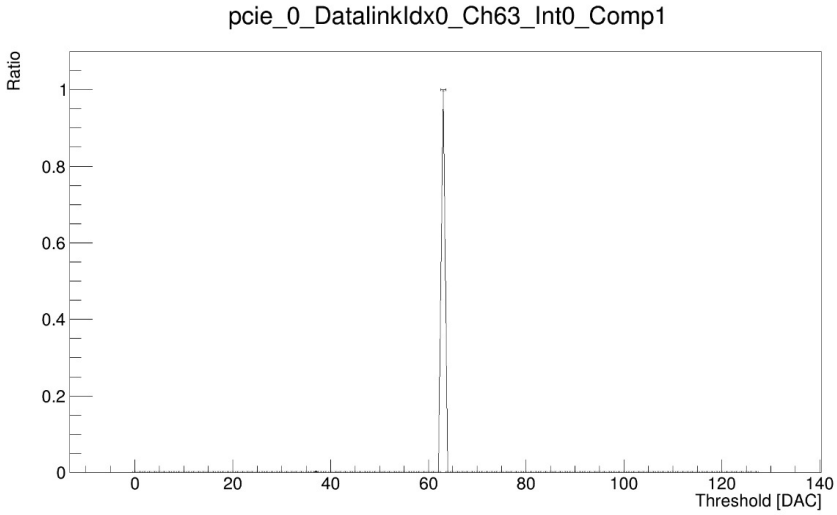


Figure 10: Result for the short-circuit of the RoB `4TSLPCEFB00208`, SiPM connector 0 and injection in channel 63.

CB FPGA. It sends digital data to the PACIFIC chip which, in turn, sends it back to the back-end, where any errors in the sequence are counted.

All tests involving PACIFIC asics are executed using the RAW data mode. In this mode, only a fraction of the data (32 channels) is sent to produce the data frame. The remaining bits are set to zero. Therefore, to fully investigate the digital part of both PACIFICs asics connected to each CB FPGA, the BER is executed 4 times.

`Error` condition
Any counter different from zero during 30 minutes for each quarter

3 Cluster Board Pseudo-Random Bit Sequence (PRBS)

This test is composed of a sequence of pseudo-random bits generated by the CB FPGA, emulating the pattern sequence created for the GBTx chip. The sequence is sent through the DataGBT links to the miniDAQ which counts each mismatch, indicating a potential problem in the transmission. Therefore, the GBT data frame and consequently all traces of the board are verified.

`Error` condition
Any counter different from zero during 2 hours

4 Brief summary of how to replace and test the RoB

This section outlines the process of replacing and testing RoBs.

- Installing a RoB

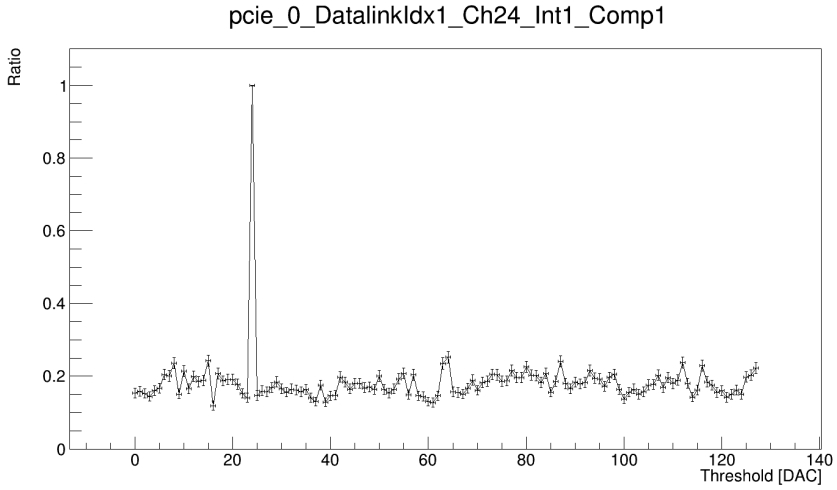


Figure 11: Noisy Channel on RoB 4TSLPCEFB00152

1. The RoB is positioned into the Tester by using the supporting screws.
2. The flat cables coming from the injector must be connected carefully, avoiding twisting motions.
3. All optical fibers are inspected before being connected by using the microscope.
4. The power supply cables can be connected without a specific sequence.

- Running the test: one must fill the blank space with the Rob ID. The FEE Test main panel (see Figure 12) allows the user to perform the following commands.

1. Button to open the power switch panel. This panel allows the user to switch and monitor the injector system and RoB device independently.
2. Reading the RoB Barcode. [4]
3. Starting Test sequence. It is also possible to select the processes individually.
4. Selection of the versions of the firmware to be used for HK and CB FPGAs. This option will be enabled if a new firmware version is available on the firmware folder.
5. Charge Selection for the injector in DACs unit (1 Byte). The expected charge (in pC) will be available on the side.
6. Equivalent Noise Like threshold for short circuit test.
7. Option to open the result files and plots after the test is completed. The logbook file of the test will be named according to the day and time of the test. One can also upload the results in ProdDb by using the panel.

[4]Needs use CAPS-LOCK button.

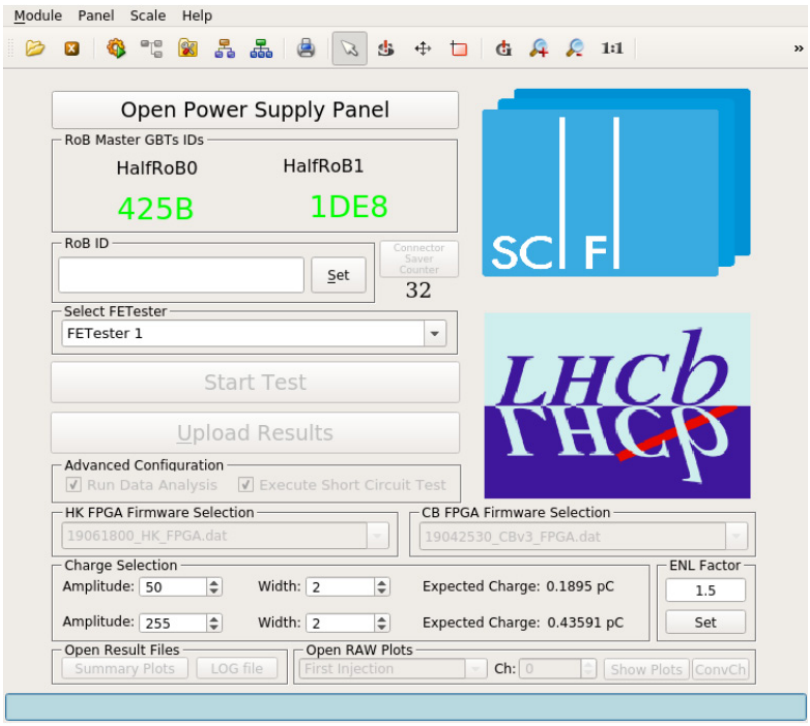


Figure 12: Main panel.

- Uninstalling the RoB

1. Turn off the RoB via the panel → Power Supply → Turn off RoB and confirm the lights are off.
2. Disconnect the optical fibers, the power supply cables, and the injector cables without a specific sequence.
3. Once removed, secure the RoB inside the case, fix the LIS cable with tape, and cover the connector with tape for protection.
4. Close the RoB with the metallic cover, place it between foam layers, and close it with the external cover.