CENTRO BRASILEIRO DE PESQUISAS FÍSICAS

COHEP - COORDENAÇÃO DE FÍSICA DE ALTAS ENERGIAS

Desenvolvimento de Experimento Antártico para Monitoramento de Raios Cósmicos

Leonardo Chaves Ruiz Guedes

Dissertação de mestrado apresentada ao Centro Brasileiro de Pesquisas Físicas como parte dos requisitos necessários à obtenção do título de Mestre em Física.

Orientador: Prof. Dr. André Massafferri Rodrigues

Rio de Janeiro - RJ

Julho, 2018

Agradecimentos

Agradeço primeiramente a Deus, por tudo que conquistei.

Ao meu amigo e orientador, Professor André Massafferri Rodrigues, não só pela oportunidade de trabalhar neste projeto, mas por toda paciência, confiança e participação na minha formação, desde o período de graduação e iniciação científica, também realizado no CBPF.

Aos meus colegas de trabalho, em especial Ulisses Carneiro e Marcos Koebcke, que participaram e ajudaram a viabilizar o desenvolvimento e conclusão deste projeto.

Ao Técnico Fernando Sousa, por toda sua experiência e suporte durante o desenvolvimento do projeto.

A todos os que colaboraram direta ou indiretamente para esta realização.

E, principalmente, aos meus pais, irmãs e namorada, que foram o meu suporte durante este período, pois, sem sua ajuda, não chegaria onde cheguei.

Resumo

O projeto CRE@AT (*Cosmic Rays Experiment at Antarctic*) é dedicado à medidas de raios cósmicos na superfície antártica, em que o mestrando trabalhou: (i) no desenvolvimento da eletrônica de *front-end* e auxilio no projeto e construção do experimento CREAT1, atualmente localizado no módulo avançado de pesquisa científica brasileiro Criosfera 1, na Antártica; (ii) no desenvolvimento do experimento CREAT2, em que foi necessário o *upgrade* a nível de *firmware*, *software* e *hardware*; e (iii) no desenvolvimento da eletrônica de *front-end*, a nível de *firmware*, bem como do *software* de aquisição e simulação para o experimento CREAT3, que tem como objetivo a medida da distribuição angular de raios cósmicos.

A escolha de instalar o experimento no continente antártico foi devido às características únicas do local. Sua atmosfera reduzida, campo magnético diferenciado, baixa umidade, ausência de elementos pesados e o mínimo de interferência do homem em seu ecossistema, tornam esse continente um excelente laboratório natural.

Através dos resultados obtidos pelo experimento, juntamente com medidas de cobertura de nuvens obtidas por satélites, estudaremos a existência de possíveis correlações entre o fluxo de raios cósmicos galácticos com a formação de nuvens. Por ser uma pesquisa que demanda estatística e diferentes amostragens, são necessários vários anos de aquisições para que tal fenômeno possa ser observado, além de sua relação ser mais evidente nos ciclos de 11 anos das atividades solares.

Além de sua proposta principal, os dados coletados pelo experimento serão integrados com outros sistemas de detecção de raios cósmicos para observação de chuveiros globais.

Palavras chave: Raios Cósmicos; Nuvens, *MaPMT*; *Front-End*; Antártica; Criosfera 1.

Abstract

The project CRE@AT (Cosmic Rays Experiment at Antarctic) is dedicated to measure cosmic rays in Antarctic surface, in which the master student worked: (i) in development of the front-end electronics and help in design and construction of the CREAT1 experiment, currently located in the advanced module of Brazilian scientific research Criosfera1, in Antarctic; (ii) the development of the CREAT2 experiment, in which it was necessary to upgrade the firmware, software and hardware; and (iii) development of the front-end electronics, at a firmware level, as well as acquisition and simulation software for the CREAT3 experiment, which aims to measure the angular distribution of cosmic rays.

The choice of installing the experiment on the Antarctic continent was due to the unique features of the place. Reduced atmosphere, different magnetic field, low humidity, absence of heavy elements and minimal interference in your ecosystem, make this continent a great natural laboratory.

Through the results obtained by the experiment, along with measurements of cloud cover obtained by satellites, we will study the existence of possible correlation between the flux of galactic cosmic rays and clouds formation. Because it's a survey that demand different sampling and statistics, requires several years of acquisitions to observe this phenomenon, in addition its relation is more evident in 11 years cycles of solar activity.

In addition to the main proposal, the collected data by the experiment will be integrated with others detection systems of cosmic rays to observe global showers.

keywords: Cosmic Rays; Clouds, *MaPMT*; *Front-End*; Antarctic; Criosfera 1.

Sumário

Lista de Figuras

Lista de Tabelas

Abreviaturas

RCG	*Raios Cósmicos Galácticos*
CRE@AT	*Cosmic Ray Experiment at Antarctica*
CBPF	*Centro Brasileiro de Pesquisas Físicas*
UERJ	*Universidade do Estado do Rio de Janeiro*
INCT	*Instituto Nacional de Ciência e tecnologia*
CAE	*Chuveiro Atmosférico Extenso*
UNICAMP	*Universidade Estadual de Campinas*
CLOUD	*Cosmics Leaving OUtdoor Droplets*
UV	*Ultra Violeta*
EACF	*Estação Antártica Comandante Ferraz*
CNPq	*Conselho Nacional de Desenvolvimento Científico e Tecnológico*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
SD	*Secure Digital*
WLS	*WaveLength Shifter*
MaPMT	*Multianode Photomultiplier Tube*
PMT	*Photomultiplier Tube*
CCMT	*Circuito de Controle Manual de Threshold*
CCDT	*Circuito de Controle Digital de Threshold*
PCB	*Printed Circuit Board*
PCI	*Placa de Circuito Impresso*
HV	*High Voltage*
PC	*Personal Computer*
DC	*Direct Current*
AC	*Aternating Current*
UART	*Universal Asynchronous Receiver-Transmitter*
USART	*Universal Synchronous Asynchronous Receiver-Transmitter*
ECL	*Emmiter-Coupled Logic*
CI	*Circuito Integrado*
EPROM	*Erasable Programmable Read-Only Memory*
TTL	*Transistor-Transistor Logic*
SPI	*Serial Peripheral Interface*
I^2C	*Inter-Integrated Circuit*

CS	*Chip Select*
ASCII	*American Standard Code for Information Interchange*
USB	*Universal Serial Bus*
VME	*VERSAModule Eurocard*
FPGA	*Field Programmable Gate Array*
CERN	*Organisation Européenne pour la Recherche Nucléaire*
ENV	*Equivalent Noise Value*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuits*
RTC	*Real Time Clock*
PWM	*Pulse Width Modulation*
QDC	*Charge to Amplitude Conversion*
NIM	*Nuclear Instrumentation Module*
LED	*Light Emitting Diode*
ASIC	*Application Specific Integrated Circuits*
MAROC	*Multi Anode ReadOut Chip*
ADC	*Analog to Digital Converter*
DAC	*Digital to Analog Converter*
RAM	*Randon Access Memory*

Capítulo 1

Introdução

Estudos indicam que ocorrem variações na formação de nuvens devido à incidência de radiação cósmica, tendo, assim, implicações no clima do planeta[1][2][3]. Considerando a direção do campo magnético, o tamanho reduzido da atmosfera e a ausência de elementos pesados presentes no Polo Sul, um estudo sobre a correlação da radiação cósmica e o clima no continente antártico se faz importante para o seu maior entendimento. Essas características únicas do continente, causam aumento significativo no fluxo dos raios cósmicos galácticos (RCG) que chegam em sua superfície, e o mínimo de interferência humana em seu ecossistema o torna um excelente laboratório natural, o que nos motivou no desenvolvimento do projeto CRE@AT (*Cosmic Ray Experiment at Antarctica*), dedicado ao estudo e monitoramento de raios cósmicos no continente antártico.

Os raios cósmicos são partículas energéticas provenientes do espaço que fornecem informações importantes acerca do sistema solar e do Universo. Essas partículas interagem na alta atmosfera gerando outras partículas através de processos físicos bem conhecidos. A maior parte das partículas geradas nesse processo interagem com os elementos da atmosfera e são absorvidas, entretanto algumas partículas, como os múons, conseguem chegar até a superfície e serem detectadas por instrumentos de detecção, agindo, assim, como indicativos da presença de raios cósmicos.

Em 12 de janeiro de 2012 foi inaugurado o módulo avançado de pesquisa científica brasileiro Criosfera 1, que viabilizou pesquisas de diversas áreas da ciência e nos permitiu instalar o experimento CRE@AT. Devido às condições peculiares do módulo, onde a infraestrutura de energia e transmissão de dados é limitada e o frio é intenso (-60º C no inverno), é necessário que haja uma completa autonomia da instrumentação. Assim, todos

os itens do experimento tiveram que ser cuidadosamente desenhados, produzidos e testados no laboratório do CBPF.

O projeto propôs o desenvolvimento de três experimentos de raios cósmicos, em que o primeiro, chamado de CREAT1 e considerado um piloto para a viabilização do projeto, foi produzido e enviado para o continente antártico em 2014. No segundo, chamado de CREAT2, foi realizado um *upgrade* em sua geometria, o que garantiu uma maior área de incidência, além de diversas melhorias a nível de *software* e hardware. O terceiro experimento, o CREAT3, ainda em fase de desenvolvimento, tem o objetivo de obter o modelo de produção dos raios cósmicos no continente antártico.

Devido a falta de recursos financeiros do Instituto Nacional de Ciência e Tecnologia (INCT) da Criosfera, muitas missões tiveram que ser adiadas. Infelizmente, o experimento CREAT2, embora finalizado, não pôde ser enviado para estação Criosfera 1, logo, os dados e resultados descritos nesta dissertação referentes ao seu detector só puderam ser coletados no Brasil.

1.1 Motivação

A todo momento recebemos a presença de raios cósmicos na superfície provenientes do sol e de explosões de estrelas. O primeiro físico a perceber esse intenso fluxo de radiação proveniente do espaço foi o austríaco Victor Franz Hess, recebido o prêmio Nobel de 1936, como fruto de sua descoberta. Antigamente, pensava-se que a influência na leitura de um contador Geiger vinha de isótopos radioativos oriundos da crosta terrestre. Hess, então, realizou um experimento no qual monitorou um contador Geiger em pontos de maior altitude, percebendo que, a medida que a altitude aumentava, o contador mantinha sua taxa de contagem. Entretanto, quando Hess levou o contador Geiger em um balão a 5 mil metros da superfície, notou que o contador aumentava a contagem, ao contrário do que se esperava. Sendo assim, a hipótese de influência dos isótopos do interior da crosta terrestre foi descartada e deduziu-se que essa radiação vinha da alta atmosfera, começando assim o estudo sobre raios cósmicos [4].

Os raios cósmicos são partículas elementares estáveis, prótons em sua maioria, de alta energia, que colidem constantemente com moléculas da sua atmosfera, principalmente nitrogênio e oxigênio [4]. A cadeia de eventos que se inicia após a interação de um raio cósmico com uma molécula da alta atmosfera terrestre é denominada de CAE (Chu-

veiro Atmosférico Extenso), mostrado na Figura 1.1. Os produtos da primeira interação movimentam-se aproximadamente na mesma direção do primário, dando origem a uma cascata de outras interações e podendo gerar um número (os denominados secundários) superior a 10^6 partículas. Um CAE é composto basicamente por 90% de elétrons, pósitrons e fótons, 9% de partículas alfa e 1% de partículas hadrônicas (prótons energéticos e píons carregados) [4][5][6].

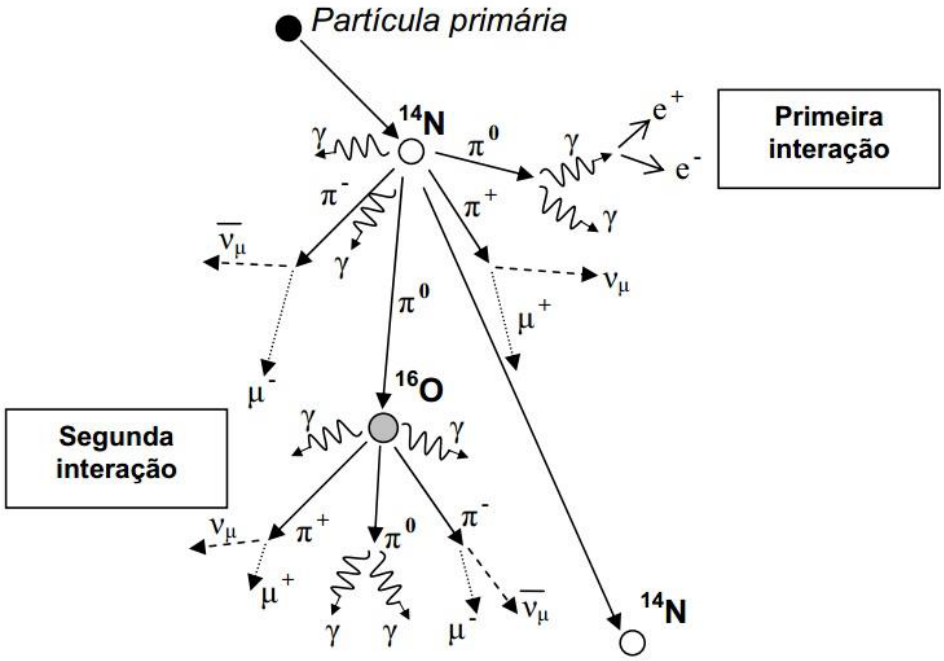


Figura 1.1: Chuveiro Atmosférico Extenso (CAE) [4].

A primeira colisão produz, em sua maioria, partículas denominadas de mésons, pions[1] e káons. Os mésons, em especial, decaem antes de interagirem e dão origem aos múons. A quantidade média de partículas produzidas nessa interação depende da energia do primário e do parâmetro de impacto da colisão. A densidade de partículas de um CAE ao nível do mar pode alcançar dezenas de milhares de partículas e a componente muônica representa até 15% do total de partículas carregadas [4][8].

Os múons são partículas elementares carregadas, com massa duzentas vezes maior

[1]No Brasil, um dos mais conhecidos físicos e pesquisadores no ramo de raios cósmicos foi o brasileiro César Lattes. O professor César Lattes, que teve papel importante na fundação do CBPF e CNPq, também criador do Departamento de Raios Cósmicos e Cronologia do Instituto de Física "Gleb Wataghin"da UNICAMP, em 1946, descobriu, na Bolívia, a existência do méson π, partícula com massa maior que elétron e inferior ao próton [7].

que a massa do elétron, são instáveis e têm vida média de aproximadamente dois microssegundos. Os múons também decaem em outras partículas, tendo como canal de decaimento a produção de elétrons (e^-), pósitrons (e^+), neutrino do elétron (ν_e), neutrino do múon (ν_μ), antineutrino do elétron ($\bar{\nu}_e$) e antineutrino do múon ($\bar{\nu}_\mu$) [4].

A maioria das partículas produzidas nestes chuveiros é absorvida pela atmosfera alta, exceto os múons, que, devido a pequena energia perdida no processo de ionização e longo tempo de vida, atravessam a atmosfera sem sofrer grandes deflexões. Na Figura 1.2 é mostrado o gráfico da concentração destas partículas nas diferentes altitudes da atmosfera. Como são pouco absorvidos pela atmosfera, os múons consistem nas partículas carregadas da radiação cósmica mais abundantes ao nível do mar [4].

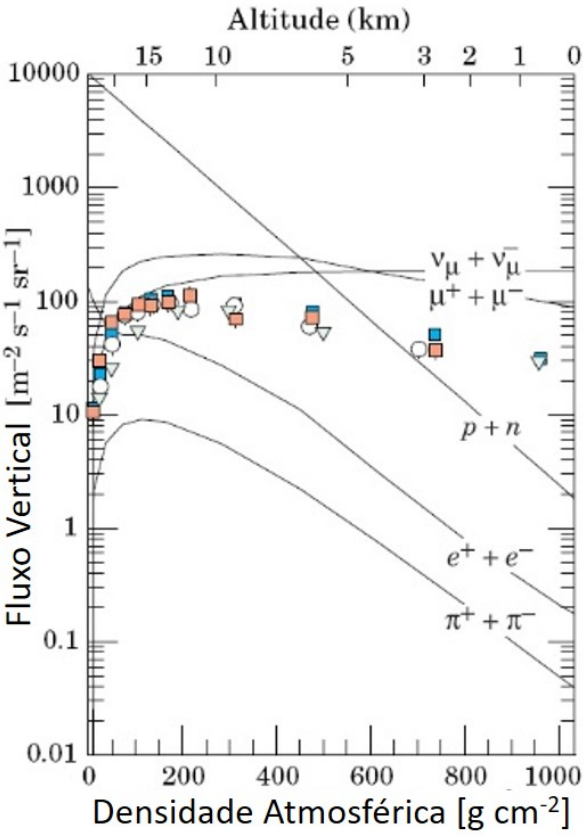


Figura 1.2: Gráfico da concentração de partículas em diferentes altitudes [Adaptado][4].

O gráfico da Figura 1.2, em escala logarítmica, demonstra como se comportam diversas partículas para diferentes altitudes. Na atmosfera alta, encontramos grandes concentrações de prótons e nêutrons provenientes do espaço diminuindo à medida que chegam à superfície terrestre. Já elétrons e píons que são formados a partir das colisões de raios cósmicos primários com átomos da alta atmosfera, são gradualmente absorvidos, ou seja, interagem com a matéria enquanto diminui a altitude. Entretanto, os múons e neutrinos chegam até a superfície com fluxo relativamente alto, comparados com as demais partículas [4].

Os múons chegam à superfície da Terra com velocidades próximas a da luz, com um valor médio $v = 2,992 \times 10^8 (m/s)$. O tempo que os múons levam para percorrer os 15 km de atmosfera é dado por $T = 50,13 \times 10^{-6} (s)$ [6].

Um dos fatores que podem alterar significativamente o fluxo de RCG presentes na superfície são as atividades solares. Atividades solares são consideradas variações magnéticas que ocorrem no Sol ou a partir dele, tendo sua intensidade calculada através da contagem do número de manchas solares. As manchas solares são regiões onde ocorre uma redução de temperatura e pressão das massas gasosas na fotosfera do Sol, onde são emitidas partículas ionizantes (basicamente elétrons) também chamadas de ventos solares. Essas partículas associadas aos ventos solares ficam presas na magnetosfera, criando, assim, uma diferença de potencial que bloqueia os raios cósmicos, diminuindo seu fluxo. A anticorrelação entre os RCGs e os ciclos de manchas solares pode ser facilmente observadas através das medidões realizadas no decorrer dos anos, ilustrada na Figura 1.3.

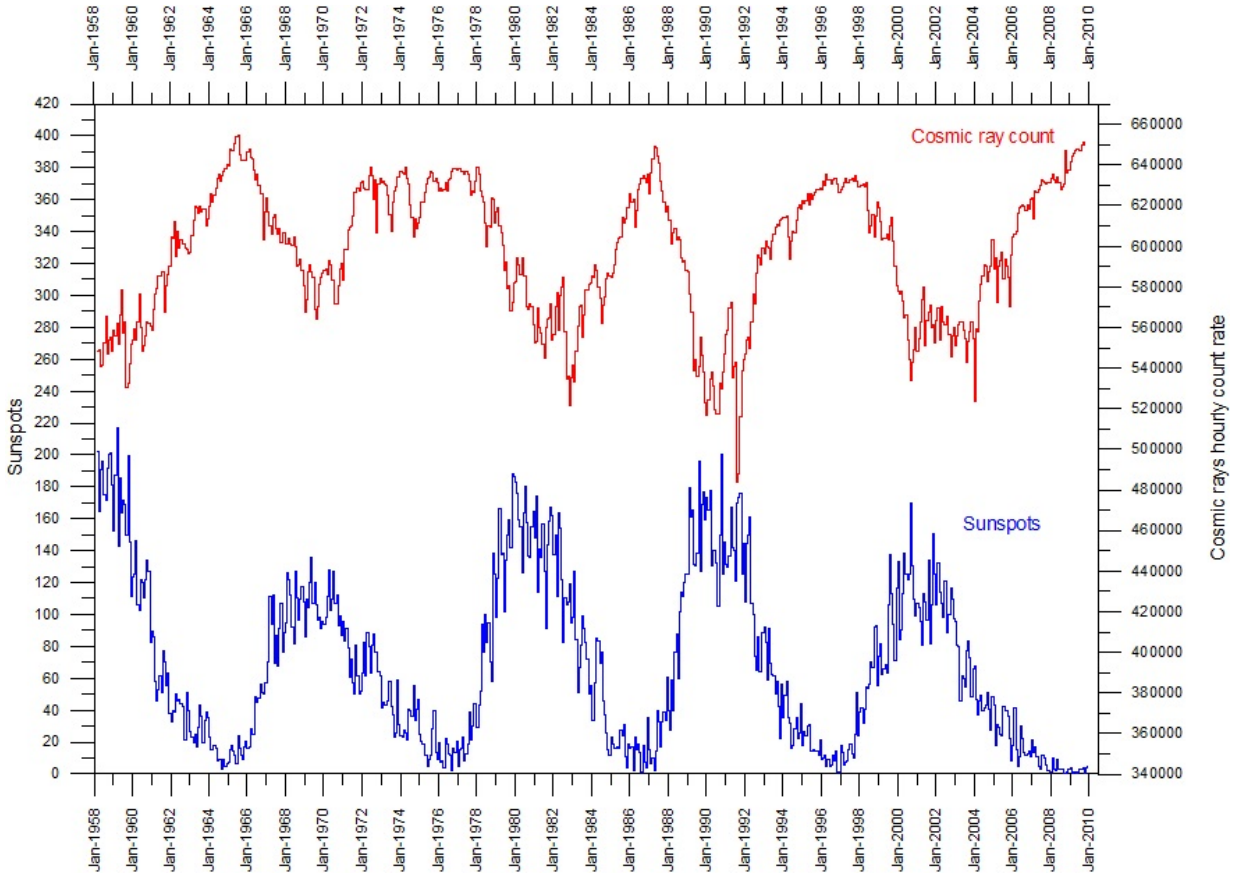


Figura 1.3: Anticorrelação entre os ciclos de manchas solares (em azul) e presença dos RCGs (em vermelho) na superfície terrestre[9].

As séries temporais mostram claramente uma queda na contagem de raios cósmicos nos períodos de máxima atividade solar e um aumento dessa contagem nos períodos de mínima atividade solar.

Os efeitos diretos das variações da luminosidade do sol nos ciclos de atividades solares resultam em mudanças muito pequenas na temperatura média terrestre (cerca de $0,1°C$). No entanto, é observada uma diminuição significativa na temperatura terrestre nos períodos de mínima atividade solar, sendo explicado apenas por algum efeito secundário ocasionado por essas atividades solares. Atualmente, o principal candidato para tais resultados climáticos é a influência dos raios cósmicos na formação de nuvens pelo efeito de ionização, similar ao processo que ocorre nas câmaras de nuvens.

A câmara de nuvens de Wilson, inventada por Charles T.R. Wilson, em 1911 [10], é um dos primeiros e mais simples detectores criados para o estudo de física de partículas. O

método inventado permitia a visualização do trajeto das partículas, promovendo diversas descobertas como o pósitron, méson π e inúmeras teorias sobre a interação de partículas elementares.

A câmara utilizava um equipamento que comprimia e expandia o ar presente nela. Na rápida expansão do ar, sua temperatura caia segundo a fórmula para a expansão adiabática:

$$TV^{\gamma^{-1}} = CONSTANTE \tag{1.1}$$

Onde T é a temperatura do ar, V é o volume ocupado por ele, e $\gamma \equiv C_P/C_V$, ou seja, a razão entre os calores específicos molares a pressão e volume constantes, lembrando que γ é sempre maior que 1, considerando um gás ideal.

Segundo a Equação 1.1, o rápido aumento do volume causa uma súbita queda na temperatura, criando um estágio de supersaturação do vapor d'água. A alta concentração desse vapor, devido à baixa temperatura, se condensa na passagem de partículas carregadas, formando gotículas de água e consequentemente nuvens. Esse processo é similar ao rastro de nuvens que ocorre na passagem de um avião no céu.

Isso ocorre porque partículas ionizantes arrancam elétrons das moléculas de ar produzindo íons que servem como núcleos de condensação. A alta concentração de vapor d'água se condensa em torno desses íons, produzindo traços ao longo do trajeto dessas partículas. A Figura 1.4 apresenta a câmara feita por Wilson e uma imagem do trajeto de partículas obtida através dela.

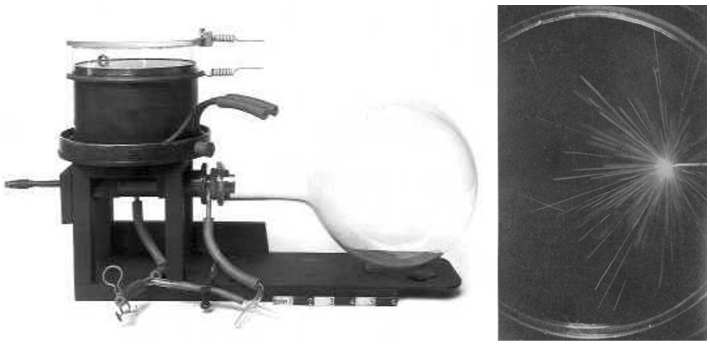


Figura 1.4: a)Primeira câmara de nuvens, inventada por Wilson [11] b)Imagem fotografada da trajetória de partículas [12].

No processo de condensação e formação de nuvens, são necessárias duas condições para que ocorra. Primeiro, deve haver uma saturação no ar que ocorre quando ele é resfriado abaixo do seu ponto de orvalho, mais comum quando o vapor d'água é adicionado. Segundo, é necessário que haja uma superfície sobre a qual o vapor d'água possa condensar. Quando a condensação ocorre no ar acima do solo, núcleos de condensação servem como superfície sobre a qual o vapor d'água condensa [13]. Estudos indicam que em condições onde o ar possui pouca concentração de poeira e outros aerossóis, a deposição de vapor d'água é extremamente improvável, exceto em condições supersaturadas, como presentes nas câmaras de nuvens. Na atmosfera real, a concentração de vapor ambiente é centenas de vezes menor, necessitando da presença de aerossóis para que a formação de nuvens ocorra. Os aerossóis atmosféricos são partículas sólidas ou líquidas presentes no ar, podendo ser naturais ou artificiais. Alguns aerossóis são originários da suspensão da poeira dos desertos e sal dos oceanos pelos ventos, outros são compostos pelo carbono emitido por fábricas e automóveis, ou pelos incêndios florestais. A influência desses aerossóis no clima pode ocorrer de maneira direta ou indireta. A forma direta ocorre quando os aerossóis mais escuros, como o carbono ou poeira, absorvem a luz solar aumentando a temperatura terrestre através do seu efeito estufa, ou quando os aerossóis de cor mais clara, como originários dos oceanos ou sulfatos, refletem os raios solares, diminuindo assim a temperatura terrestre. De maneira indireta, temos a formação dos aerossóis a partir de vapores percursores, como o ácido sulfúrico, pelo processo de nucleação.

Em 1997, os pesquisadores Heinrich Svensmark e Eigil Friis-Christensen, do instituto de pesquisas espaciais da Dinamarca, foram um dos primeiros pesquisadores a propor que os RCGs e as atividades solares poderiam ser fatores de influência do clima. Em teoria, os RCGs ao passarem pela atmosfera, podem ionizar alguns compostos voláteis. Quando o agrupamento de partículas atingem certo tamanho crítico, existe um gasto de energia ao invés de ganho de energia associado a evaporação. A inclusão de íons nesses grupos pode reduzir o tamanho crítico, formando novas partículas muito mais estáveis. Quanto maior a incidência dos RCGs, maior será a concentração desses íons, influenciando ainda mais na formação de nuvens, como ilustrado na Figura 1.5.

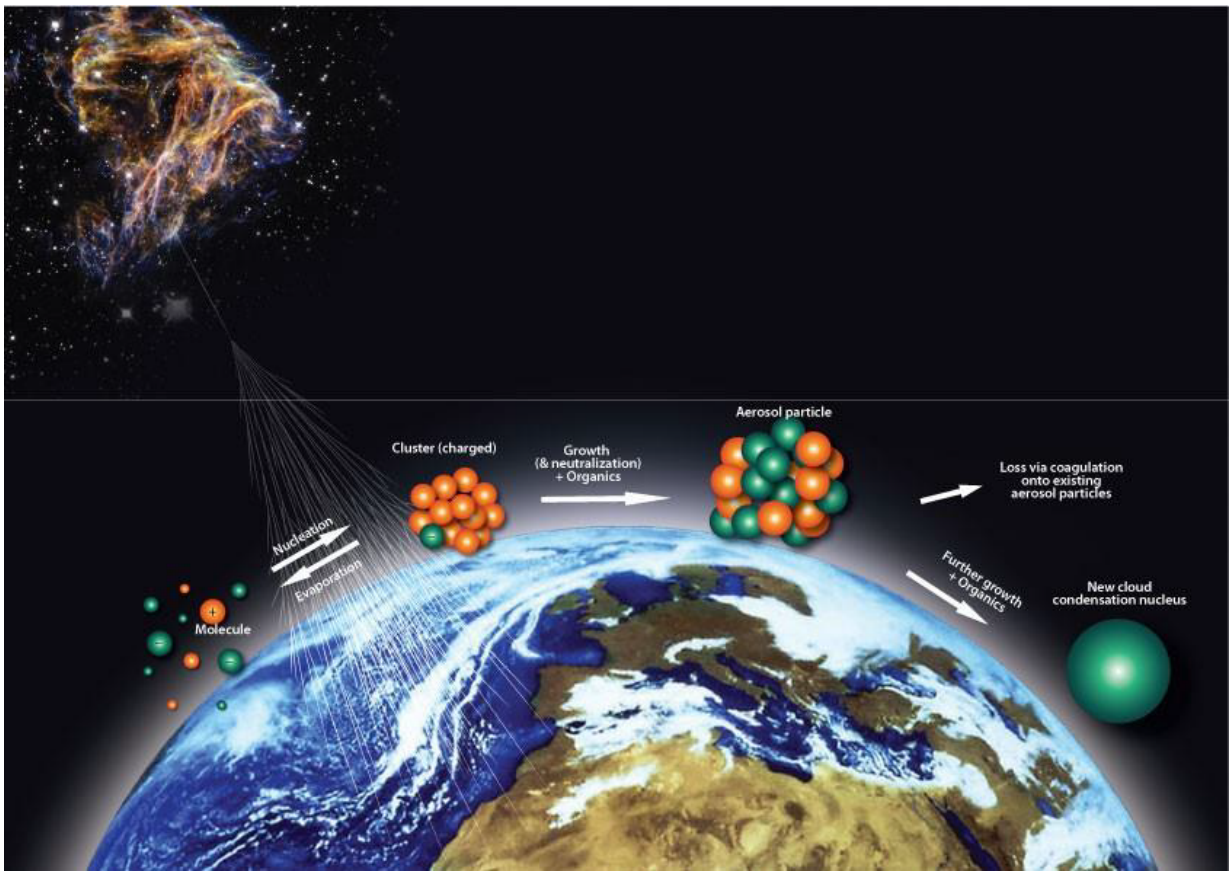


Figura 1.5: Processo de formação de nuvens na incidência dos RCGs [14].

A taxa de íons gerada nessa passagem depende principalmente da profundidade atmosférica, pois quanto maior a concentração moléculas na atmosfera, mais íons serão gerados.

Por ser uma reação probabilística, a profundidade atmosférica pode ser usada como uma medida da probabilidade de ocorrer uma reação no momento em que um raio cósmico particular atinja uma determinada altitude. A profundidade atmosférica é a massa de uma coluna de ar acima de uma superfície, com dimensões de *massa/área*.

Na Figura 1.6 podemos observar os dados coletados por satélites dessa relação entre cobertura de nuvens e raios cósmicos. Resultados esses, que foram anunciados em 1996 no encontro de ciências espaciais, em Birmingham, e publicado como " *Variation of cosmic ray flux and global cloud coverage - a missing link in solar-climate relationships*"(Svensmark and Friis-Christensen 1997). Até então, não havia confirmação teórica para tais resultados.

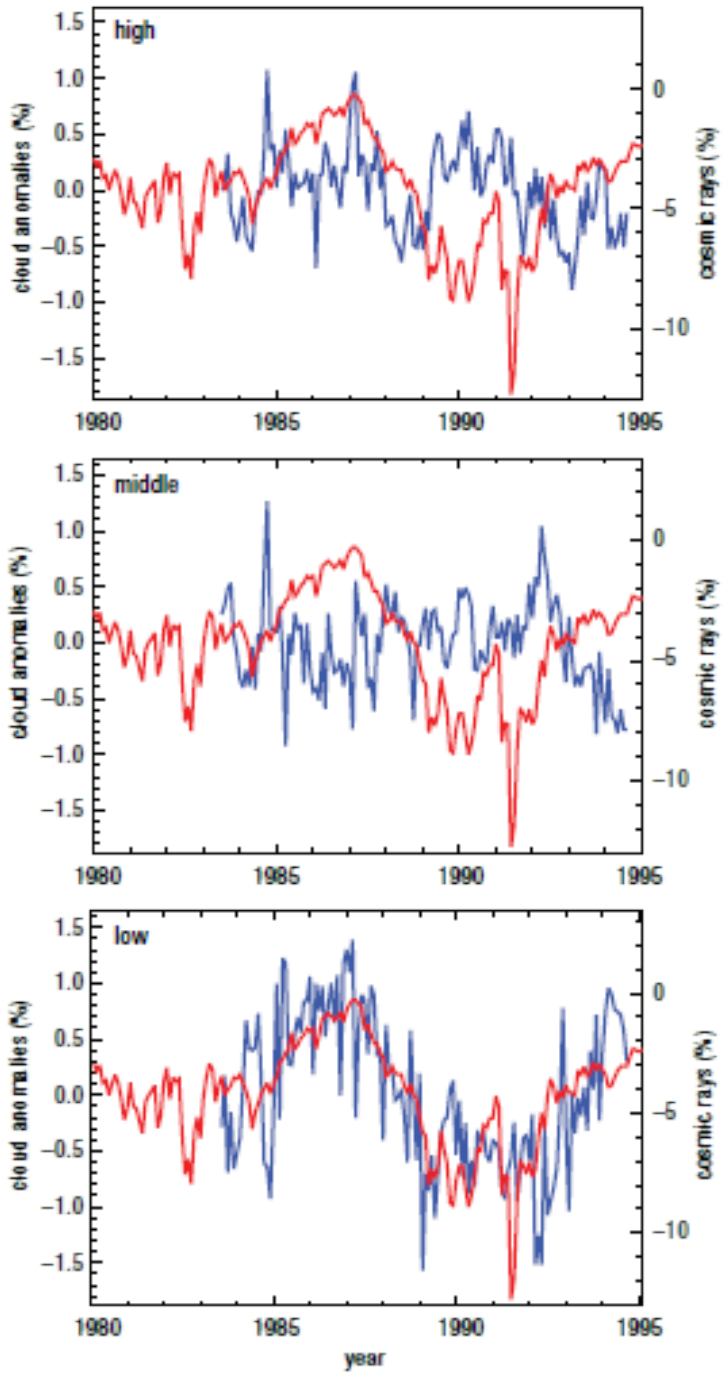


Figura 1.6: A linha azul mostra variações na nebulosidade global coletada pelo *International Satellite Cloud Climatology Project*. A linha vermelha é o registro de variações mensais nas contagens de raios cósmicos na estação de Huancayo, no Peru[15].

Devido a maior presença de aerossóis em regiões de baixa altitude, podemos ob-

servar maior correlação entre a cobertura de nuvens e fluxo de RCGs. Na Figura 1.7 são apresentados alguns dados mais recentes dessa relação.

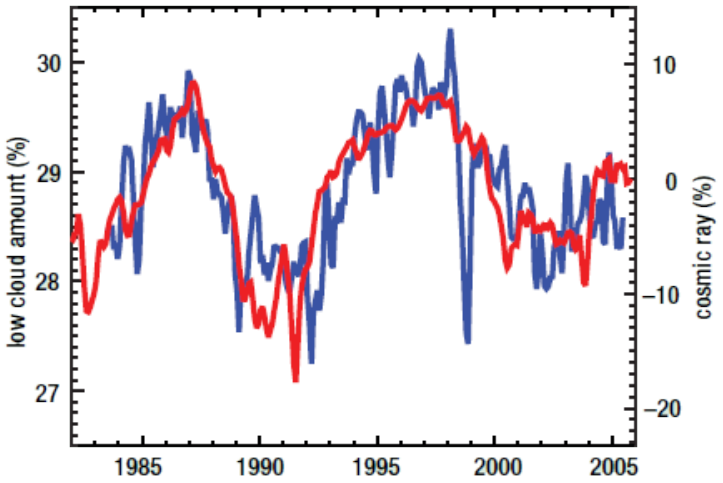


Figura 1.7: A linha azul mostra variações na nebulosidade global. A linha vermelha é o registro de variações mensais nas contagens de raios cósmicos[15].

O Experimento *CLOUD*, localizado no CERN, atualmente é um dos maiores experimentos que procuram entender essa influência dos raios cósmicos em aerossóis e nuvens, e sua correlação com o clima. O experimento consiste em uma câmara em que é possível ter completo controle das condições de temperatura, pressão, incidência de raios UV e combinações de gases atmosféricos para a emulação de uma atmosfera real. Além disso, o experimento tem a sua disposição um feixe do Proton Synchrotron para a simulação de raios cósmicos, que são 90% prótons de hidrogênio. Por não ser um processo completamente conhecido, os atuais modelos computacionais do clima não simulam tão bem os efeitos dos raios cósmicos e nuvens. O Experimento CLOUD começou efetivamente a coletar dados a partir de 2010 e funciona até hoje.

1.2 Continente Antártico

A Continente Antártico possui diversas peculiaridades que o difere dos demais continentes. Além de ser um dos maiores, seu centro está localizado no Polo Sul, onde se convergem as linhas longitudinais geográficas e onde os campos magnéticos da terra são ortogonais à superfície. Ele é considerado o continente mais frio, mais seco e com a maior

média de altitude do planeta. A temperatura média na costa durante o verão é de -10°C e no interior -40°C, podendo chegar à -80°C no inverno [16].

Para um maior controle das explorações no continente antártico, em 1 de dezembro de 1959, foi assinado o Tratado da Antártica, em que é permitida apenas a exploração científica do continente [17], garantindo o mínimo de impacto ambiental nesta localidade. O Brasil aderiu ao tratado em 1975 e, no início da década de 1980, foi inaugurada a Estação Antártica Comandante Ferraz (EACF) [18]. Localizada na ilha do Rei George, a 130 km da Península Antártica, a Estação Antártica Comandante Ferraz serve como refúgio e laboratório para diversos pesquisadores, oferecendo suporte em apoio da Marinha do Brasil.

Em 12 de janeiro de 2012, foi inaugurado o módulo avançado de pesquisa científica brasileiro Criosfera 1, desde então, localizado no interior do continente antártico, a 640 km do polo sul geográfico, latitude 84°S, como visto na Figura 1.8. Iniciativa do Ministério da Ciência, Tecnologia, Inovações e Comunicações, e do CNPq, o módulo é a primeira iniciativa de pesquisa brasileira a operar de forma contínua, autônoma e remota no interior do continente antártico [19].

Figura 1.8: Localização do módulo Criosfera I[20].

Sua estrutura conta com um espaço interno de $6,3 \times 2,5 \times 2,6[m^3]$ e um sistema híbrido eólico/solar autossuficiente para geração de energia elétrica, que permitem mantê-lo em funcionamento o ano inteiro, inclusive durante o inverno polar. Com objetivo de ser uma plataforma de pesquisa multiusuária, possui um grande potencial para estudos voltados à

biotecnologia, meteorologia, paleoclima, física, química da atmosfera, e astrofísica de altas energias, oferecendo condições infraestruturais à pesquisa com o mínimo de alteração do ecossistema local.

O módulo possui uma estação meteorológica que monitora temperatura do ar, pressão atmosférica, umidade relativa, intensidade e direção do vento e radiação solar. Um sistema ultrassônico, instalado no exterior do módulo, mede a dinâmica de deposição de neve na região em tempo real. Todos os dados meteorológicos, das concentrações de CO_2, da deposição de neve, entre outros, são enviados via satélite em tempo real para o Brasil. Estes dados, reunidos e interpretados à luz de modelos computacionais de transporte atmosférico, permitem aumentar nossa compreensão sobre a relação climática Antártica-América do Sul, o impacto da redução da camada de ozônio, da atividade vulcânica no hemisfério sul, da evolução dos processos globais de desertificação, do transporte atmosférico global de poluentes e de microrganismos, bem como aprofundar nosso conhecimento sobre a história climática contada pelos testemunhos de gelo[20].

1.3 Projeto CRE@AT

O Projeto CRE@AT (*Cosmic Rays Experiment at Antarctic*), tem como objetivo o estudo de raios cósmicos no continente antártico e sua relação com o clima, em especial, com a formação de nuvens. A escolha do continente foi devido às características únicas presente nele, que o torna uma espécie de laboratório.

A proposta inicial do experimento, foi a medição do fluxo de raios cósmicos que chega na superfície antártica, tendo como uma versão piloto, o experimento CREAT1, enviado em outubro de 2014.

O CREAT1 foi concebido como uma versão piloto para verificar a viabilidade de manter o experimento num ambiente hostil e trabalhando de maneira autônoma. Devido as dificuldades energéticas do local, principalmente no ano em que foi instalado, o experimento foi configurado para coletar dados apenas 30 minutos por dia com intervalos consecutivos de 10 minutos. A Figura 1.9 mostra o momento em que o módulo Criosfera 1 recebeu o experimento [20].

Figura 1.9: Face Sul do módulo Criosfera I[20].

Ao final de sua missão, um *upgrade* do detector foi considerado necessário. Diversos estudos sobre uma maior área de medição, quantidade de planos de coincidência e organização dos cintiladores plásticos, foram realizados até chegar na configuração atual. Sendo assim, foi então projetado e construído o detector do experimento CREAT2, de forma que fosse possível aproveitar parte da eletrônica do seu antecessor. O *firmware* do novo detector precisou ser totalmente refeito, assim como o projeto e desenvolvimento de um novo hardware de armazenamento de dados, contendo sensores de temperatura, pressão e campo magnético. Atualmente o detector do CREAT2 está localizado no CBPF, e o mesmo será enviado para o continente antártico.

O novo *setup* permite o envio de dados via satélite através do protocolo *TCP/IP* para um servidor localizado no *CBPF*, até então, os dados eram apenas armazenados em dois cartões SD, sendo um deles *backup*, necessitando que haja uma missão para coleta destes dados.

Finalizado o *upgrade*, foi então iniciado o projeto do detector do experimento CREAT3, cujo objetivo consiste na medição do modelo de produção dos raios cósmicos no continente antártico, capaz de realizar a trajetografia de partículas cósmicas.

Os três detectores são à base de tiras cintiladoras plásticas, fibras *WLS* (*wavelength shifter*) e *MaPMT* (Multianode Photomultiplier Tube), sendo o detector do CREAT1 e CREAT2 de 16 canais e o do CREAT3 de 64 canais. A escolha dessa tecnologia foi devido ao seu menor impacto ambiental e por ter sua performance menos suscetível à variações de temperatura e pressão, por não utilizar gás. A eletrônica de *Front-End* dos dois primeiros

detectores foi projetada pelo mestrando, além de grande parte dos *firmwares* e *softwares* descritos nesta dissertação.

Capítulo 2

Desenvolvimento do Projeto

2.1 Fenômenos de Cintilação

Emitir luz em consequência da absorção de energia, tanto de partículas carregadas quanto neutras, é uma propriedade presente em uma gama de materiais. Segundo o postulado de Bohr, ao introduzir o conceito de níveis de energia de seu modelo atômico, a absorção desta energia promove um dos elétrons do átomo para níveis orbitais mais energéticos porém instáveis. A desexcitação destes elétrons ocorre na forma de emissão de um fóton, como mostrado na Figura 2.1, e esse processo é chamado de excitação atômica.

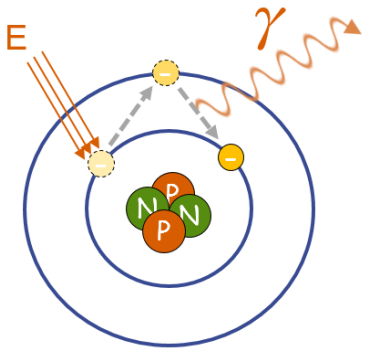


Figura 2.1: Representação da geração de fótons na desexcitação de elétrons.

Interação de Partículas Carregadas

A passagem de uma partícula negativa em colisão com um elétron orbital de um átomo faz com que ocorra uma repulsão desses elétrons. A transferência de energia pode produzir a ionização ou excitação do elétron orbital, dependendo de sua energia transmitida.

A ionização ocorre quando a colisão de uma partícula carregada com um elétron orbital faz com que esse elétron seja arrancado de sua órbita, gerando um íon livre e o átomo com carga positiva. A Figura 2.2 ilustra esse processo.

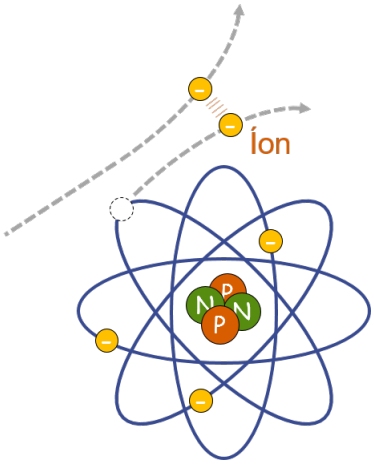


Figura 2.2: Processo de ionização gerando íons.

A passagem de uma partícula carregada de alta energia, como os múons, faz com que ela sofra deflexão e perda de energia por Bremsstrahlung, ao passar pelos átomos de um meio. Essa energia é transmitida para o fóton gerado no processo de desexcitação do elétron, como observado na Figura 2.3.

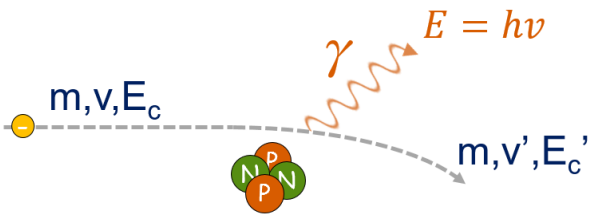


Figura 2.3: Perda de energia por Bremsstrahlung.

Interação de Fótons

A interação de fótons com os átomos ocorre de maneira diferente das partículas. Essa interações podem ocorrer de diferentes formas, como o efeito fotoelétrico, espalhamento Compton e produção de pares.

O efeito fotoelétrico foi explicado por Einstein em 1905, dando a ele o Prêmio Nobel por sua descoberta. O efeito ocorre quando um átomo é exposto a uma radiação eletromagnética de alta frequência, como um feixe de luz. Um fóton da radiação transfere sua energia total para um elétron orbital, arrancando-o com uma certa velocidade, ocorrendo o processo de ionização. A troca de energia é dada por $E_c = hf - E_{lig}$, sendo E_c a energia cinética, hf a energia da radiação incidente e E_{lig} a energia de ligação do elétron ao seu orbital, como ilustrado na Figura 2.4.

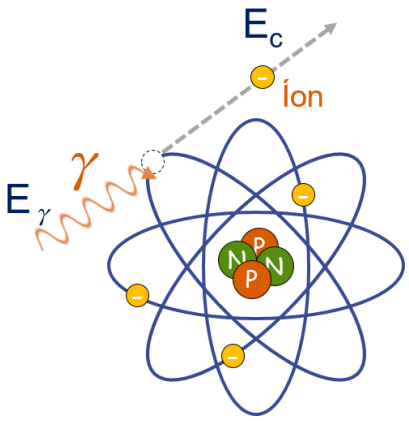


Figura 2.4: Efeito fotoelétrico gerando íon.

Quando a energia da radiação eletromagnética é maior, as chances de ocorrer o espalhamento Compton aumentam. O espalhamento Compton ocorre quando parte da energia da radiação incidente é transmitida para o elétron, na forma de energia cinética, e parte vai para o fóton espalhado, considerando também a energia de ligação do elétron ao seu orbital, vide Figura 2.5.

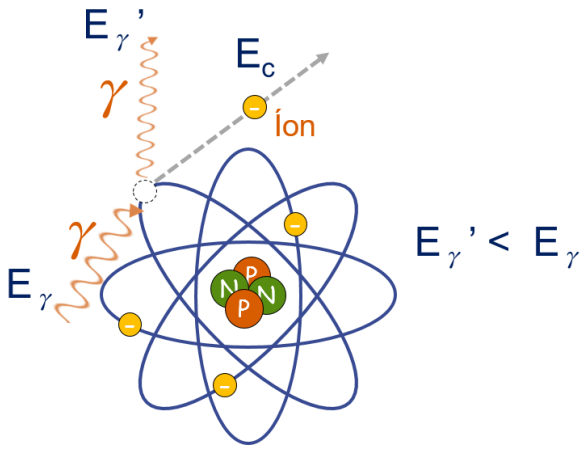


Figura 2.5: Espalhamento Compton gerando íon e um fóton espalhado.

O processo de produção de pares ocorre quando a radiação eletromagnética passa pelo núcleo de um átomo com número atômico elevado, dando origem a um par de elétron-pósitron e o desaparecimento do fóton. Para ocorrer esse fenômeno, é necessário que o fóton tenha uma energia igual ou superior a $1,022MeV$, sendo esta igual a massa do pósitron mais a massa do elétron multiplicada por c^2, segundo a equação $E = mc^2$. O processo de aniquilação do elétron e pósitron gerados acontece quando essas duas partículas são atraídas entre si, ou por um outro par oposto a elas, tendo como resultado a geração de dois fótons. A Figura 2.6 ilustra esse fenômeno.

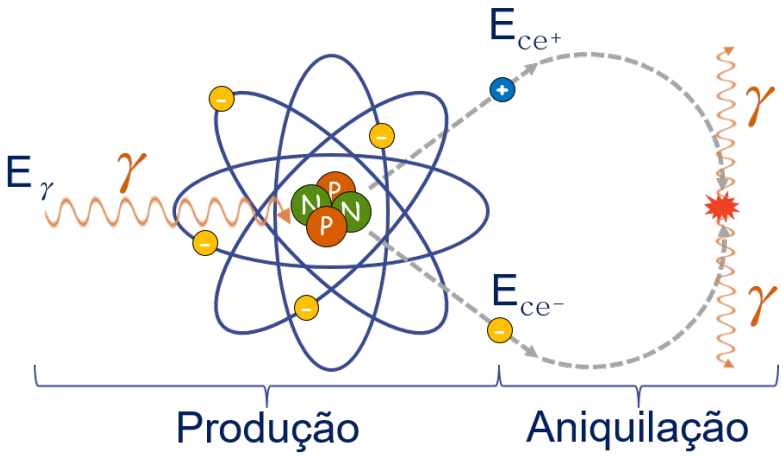


Figura 2.6: Produção de pares dando origem ao elétron-pósitron e aniquilação gerando dois fótons.

2.2 Cintiladores Plásticos e Fibras *Wavelength Shifter*

Materiais cintiladores podem ser compostos por diversos materiais, sendo eles orgânicos ou inorgânicos. Materiais orgânicos, como os utilizados nos detectores do projeto, têm a característica de terem seus elétrons excitados e desexcitados de maneira quase que instantânea, levando até 10ns nesse processo chamado de fluorescência. Materiais inorgânicos, como cristais, tem a característica de levar mais tempo nesse processo de excitação e desexcitação, podendo ser na ordem de milissegundos ou até segundo, em alguns casos, e esse processo é chamado de fosforescência.

Os cintiladores utilizados nos detectores deste projeto foram desenvolvidos pelo *Fermilab* (*Fermi National Accelerator Laboratory*) e são feitos de um material fluorescente, cujos átomos emitem fótons na faixa espectral da luz azul ao serem ionizados pelo pulso de campo elétrico de uma partícula ionizada, no nosso caso, raios cósmicos. Uma fibra *Wavelength Shifter* (WLS) modelo Y-11(175)MSJ da *KURARAY*, é acoplada ao cintilador, absorvendo os fótons de mais alta energia (luz azul) e emitindo múltiplos fótons de mais baixa energia (luz verde). Esse ganho em número de fótons verdes aumenta a eficiência do sistema de detecção, já que alguns fótons são perdidos no caminho até a fotomultiplicadora [21]. Diferente das fibras ópticas convencionais de transmissão de dados, essas fibras WLS absorvem fótons que incidem ortogonalmente em sua estrutura e guia os fótons emitidos até suas extremidades. A Figura 2.7 apresenta uma imagem do conjunto.

Cada cintilador é coberto por material opaco para bloquear a luz externa. O interior da tira e fibra *WLS* é revestido com um material reflexivo que faz com que os fótons incidentes reflitam até chegar nas células da fotomultiplicadora, gerando os pulsos de corrente. A observação deste fóton em dispositivos fotomultiplicadores consiste então na assinatura da passagem das partículas pelo detector. Os materiais cintiladores utilizados precisam conter as seguintes propriedades:

- Converter a energia cinética de partículas carregadas em luz com alta eficiência quântica;
- Ser transparente, de modo que a luz possa ser transmitida a um dispositivo capaz de captá-la;
- O processo de emissão deve ter curta duração.

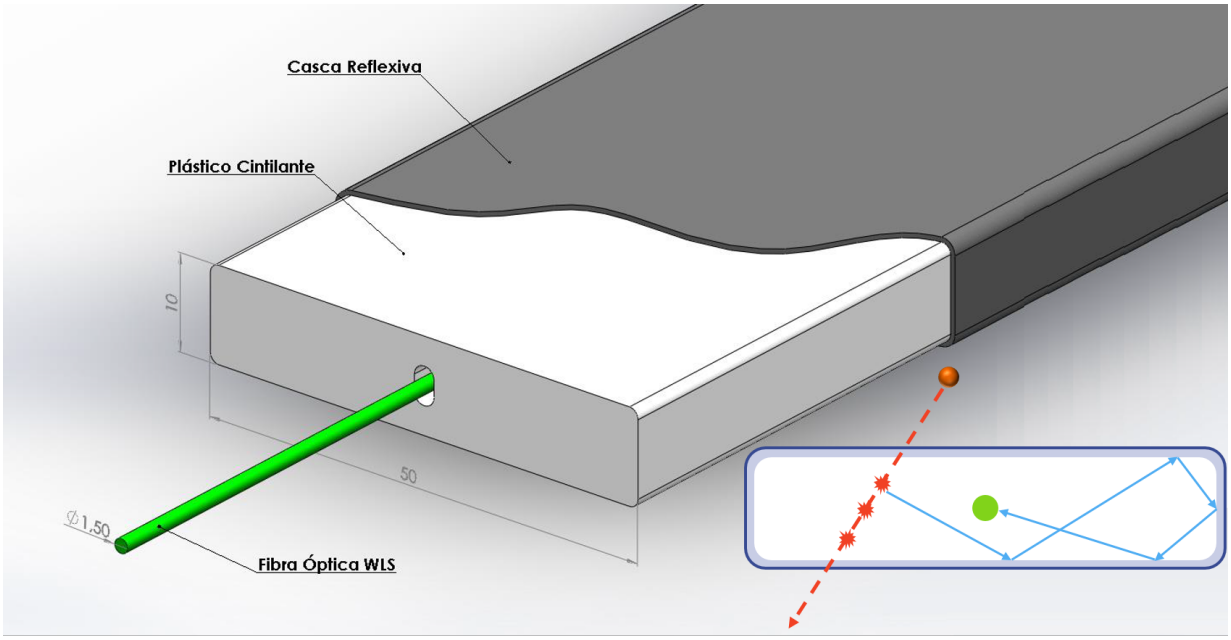


Figura 2.7: Representação da tira cintiladora plástica e da fibra WLS em milímetros[20]

2.3 Fotomultiplicadoras

As PMTs (*photomultiplier tubes*) têm a função de gerar uma corrente mensurável em decorrência da chegada de pequena quantidade de fótons. A estrutura básica de uma fotomultiplicadora tem como componentes principais: o fotocatodo, a óptica de focalização de elétrons, os dinodos e o anodo [23].

O processo de multiplicação se inicia através do efeito fotoelétrico, com a transferência da energia do fóton para um elétron no fotocatodo seguido da migração do elétron até a superfície do fotocatodo e o transporte do elétron até o primeiro dinodo. Após esse processo, os elétrons são conduzidos num processo de multiplicação até o anodo [23], como mostrado na Figura 2.8:

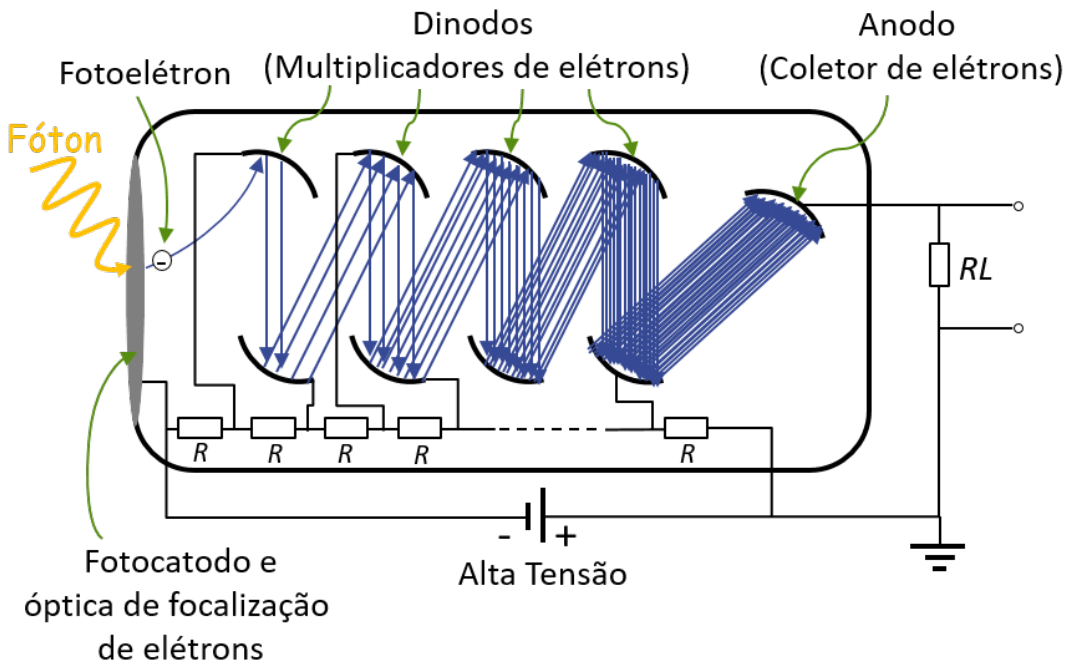


Figura 2.8: Estrutura básica de um tubo fotomultiplicador.

A óptica de focalização de elétrons é composta por eletrodos que geram um campo elétrico que guia os elétrons para a superfície do primeiro dinodo. Uma diferença de potencial é aplicada entre os dinodos subsequentes, de modo que os elétrons são acelerados gerando novos elétrons nos choques com as superfícies. O material da superfície dos dinodos é escolhido de tal forma que cada elétron incidente provoque a emissão de vários elétrons. Tipicamente, com uma diferença de potencial de 100V aplicada entre dinodos, cada elétron produza cerca de 30 novos elétrons, dos quais apenas uma pequena fração atinge o próximo dinodo, já que muitos não chegam a se desprender da superfície [23]. Dessa forma, a estrutura fotomultiplicadora composta pelos dinodos fornece um ganho em corrente elétrica expresso por $G = \alpha\delta^n$, onde α é a fração de fotoelétrons coletados e n é o número de dinodos. O valor de δ é próximo de 5 para os dinodos convencionais e depende da tensão entre dinodos. Ganhos da ordem de $10^6 \sim 10^7$ são obtidos com fotomultiplicadoras de 10 estágios [23]. Mesmo sem a presença de fótons entrando no fotocatodo, uma pequena corrente, denominada corrente de escuro, ocorre devido a geração aleatória de elétrons e lacunas que são atraídos pelo campo elétrico. Tal corrente pode ser confundida com sinais reais, necessitando de uma lógica que possa discrimina-la.

MaPMT (*Multianode Photomultiplier*)

A *MaPMT* utilizada neste trabalho, Figura 2.9, é uma fotomultiplicadora de 16 canais independentes, cujos pulsos de corrente gerados em cada um dos seus canais é enviado e tratado por um circuito externo contendo 16 amplificadores e discriminadores.

Figura 2.9: Fotografia da *MaPMT* de fabricação da Hamamatsu [24].

Segundo seu *datasheet* [25], a corrente gerada pelo acionamento de uma das células da *MaPMT* é especificado entre $-150\mu A$ e $-300\mu A$, tendo seu valor típico de aproximadamente $-200\mu A$, utilizando uma tensão de alimentação de -800V e uma lâmpada de tungstênio de baixa intensidade como fonte de luz. Este sinal ao ser injetado em um circuito com impedância de entrada de 50Ω, tem como resultado um pulso de tensão de aproximadamente $-10mV$ e largura de $1,4ns$ a 50% de amplitude, como podemos observar na Figura 2.10.

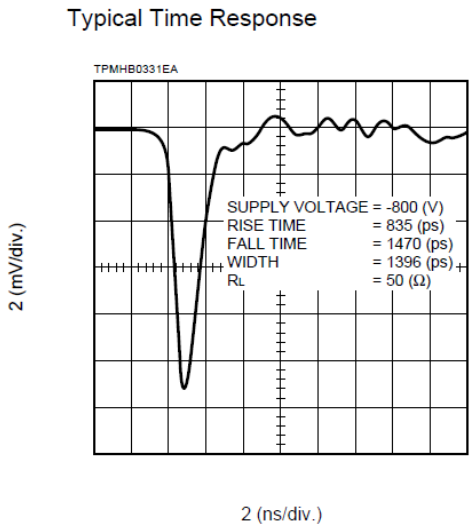


Figura 2.10: Sinal típico de um evento de poucos fótons em umas das células da *MaPMT* para uma impedância de 50Ω, segundo seu *datasheet*[25].

Com base nestes dados, foi proposto o projeto de uma eletrônica que fosse capaz de trabalhar com sinais de entrada de baixa amplitude e largura de pulso, e operasse em conjunto com tiras cintilantes em projetos envolvendo contagem de raios cósmicos. Eletrônica essa que foi utilizada nos detectores do CREAT1 e CREAT2.

2.4 Eletrônica de *Front-End*

A eletrônica de *front-end* desenvolvida neste trabalho possui 16 canais, impedância de entrada de 50Ω e é composta por quatro circuitos principais:

- Circuito amplificador;
- Circuito discriminador;
- Circuito de controle manual de *threshold* (CCMT);
- Circuito de controle digital de *threshold* (CCDT).

Uma fotografia da eletrônica é apresentada na Figura 2.11. O *layout* de sua *PCB* foi elaborado no *Software Altium Designer* pelo técnico Fernando Sousa, contendo 6 camadas dedicadas aos sinais elétricos, uma camada dedicada ao *ground* e outra às fontes de

alimentação, $+5V$ e $-5V$. O projeto completo desta eletrônica é apresentado no Apêndice A.

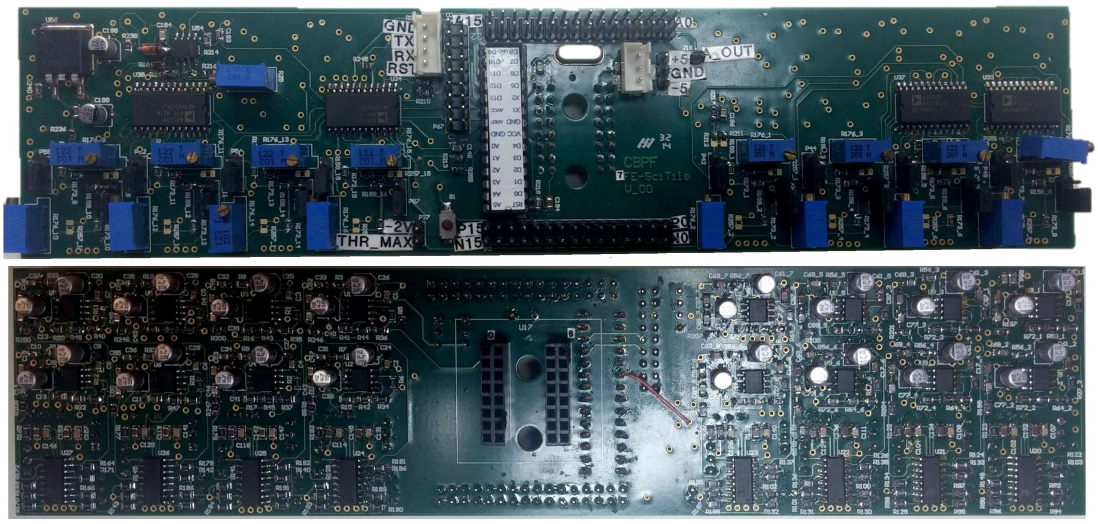


Figura 2.11: Fotografia da face superior e inferior da eletrônica de *front-end*.

Na Figura 2.12, é apresentado o posicionamento de cada componente de sua *PCB*, assim como seus sinais de entrada e saída.

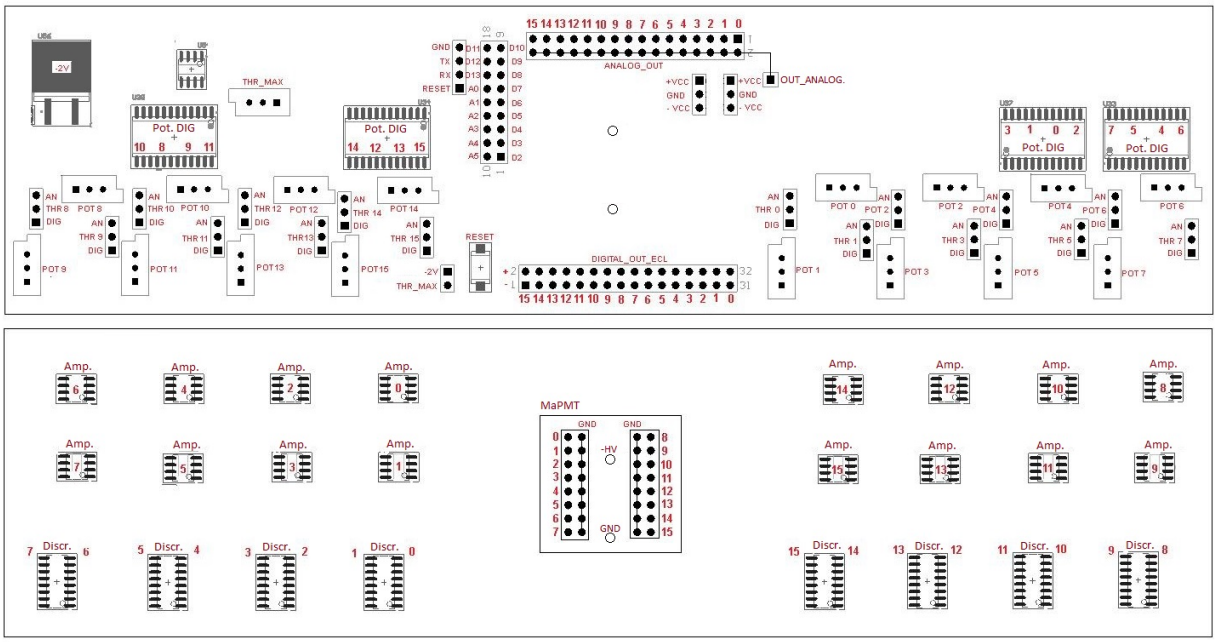


Figura 2.12: Diagrama de componentes e sinais de saída e entrada da eletrônica de *front-end* desenvolvida.

Na Figura 2.13, podemos observar o diagrama dessa eletrônica de *front-end* de 16 canais, indicando a comunicação entre os circuitos, periférico e instrumentos que são usados no *setup* experimental.

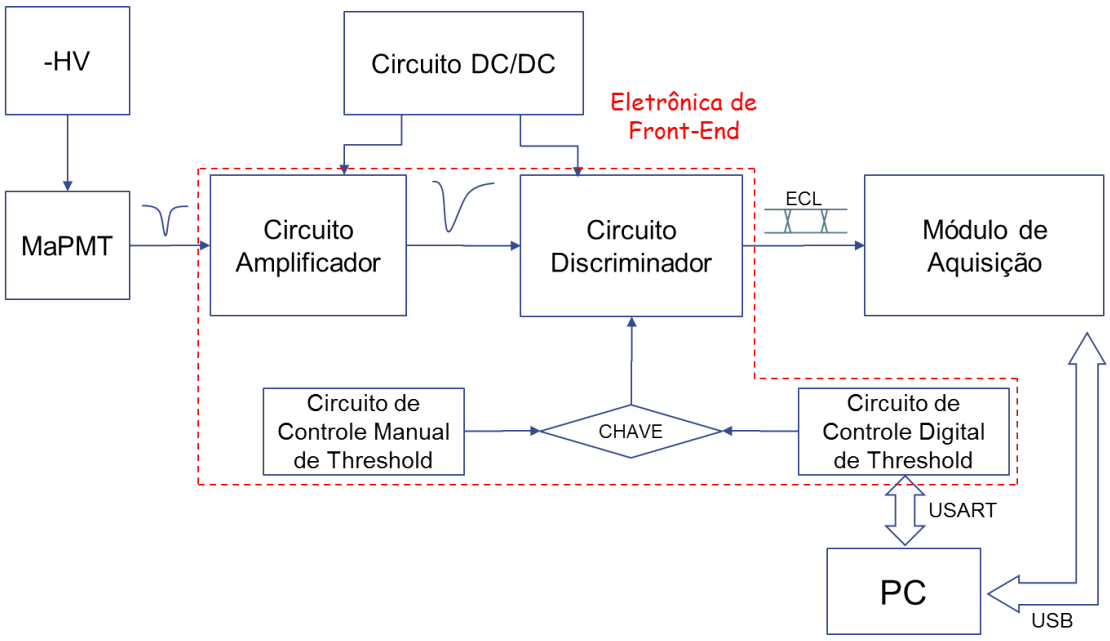


Figura 2.13: Diagrama de um *setup* experimental.

2.4.1 Circuito Amplificador

O sinal proveniente da *MaPMT* não possui amplitude e largura que possam ser discriminadas, por isso um circuito amplificador não inversor foi projetado. O amplificador, utilizando o circuito integrado *AD8001AR* [26], foi projetado com um ganho de aproximadamente 200, acrescentado em sua saída um capacitor que eliminasse o sinal *DC* e um filtro passa-baixa, com a função de alongar o sinal tipicamente de $2ns$, vindo da *MaPMT*, para aproximadamente $20ns$, facilitando assim o processo de discriminação de sinal, próximo estágio de sua eletrônica.

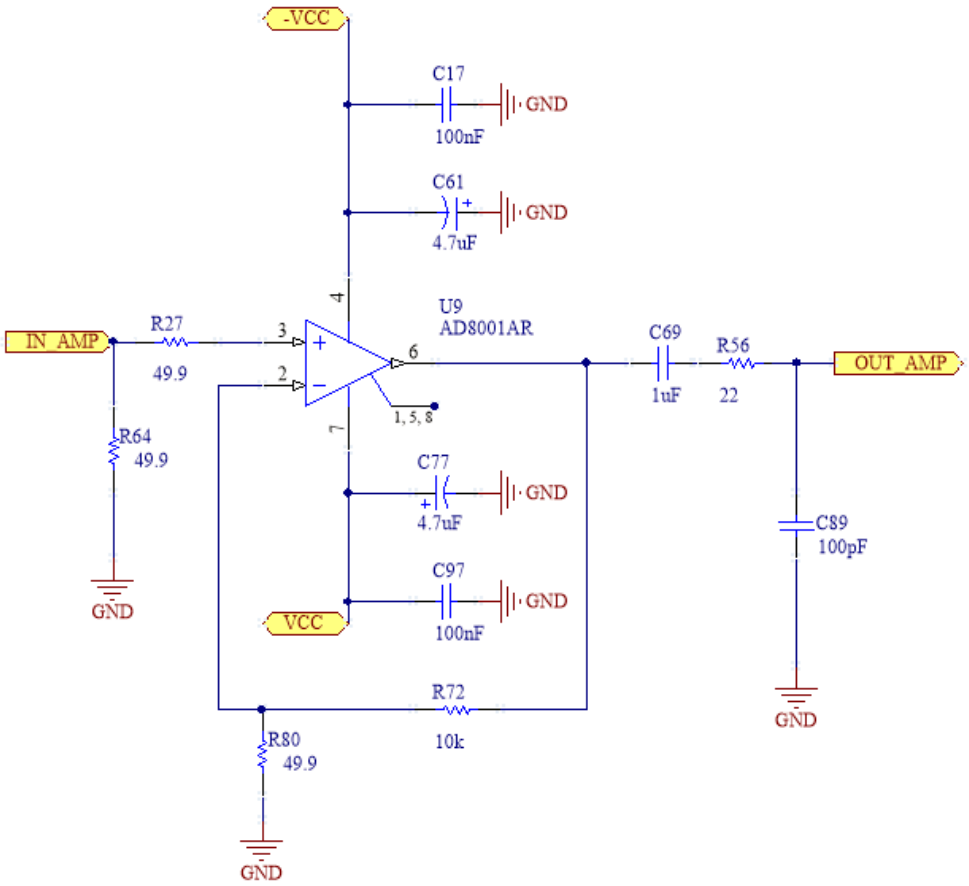


Figura 2.14: Desenho esquemático do circuito amplificador e filtro RC - realizado no *Software Altium Designer*.

Devido ao filtro passa-baixa, com a finalidade de realizar o *shaper* do sinal, a amplitude de saída do amplificador diminui, resultando num ganho total de aproximadamente 20.

2.4.2 Circuito Discriminador

Todo circuito eletrônico possui um ruído intrínseco proveniente da fonte de alimentação, temperatura ou até mesmo interferências externas, que dependendo de sua amplitude, pode ser confundido com os sinais elétricos do sistema. Um circuito discriminador pode ser usado para separar o sinal de interesse do ruído. Como a amplitude dos sinais de interesse é maior que a do ruído intrínseco, a discriminação dele é realizada através da comparação entre o sinal de saída do amplificador, sendo este somado ao ruído, e uma tensão de

referência (*threshold*). Quando a amplitude deste sinal é maior que a tensão de referência, ambos em módulo, ele é transformado em um pulso diferencial na lógica *ECL* (*emmiter-coupled logic*). O circuito discriminador possui um comparador modelo *AD96687BR* [27], de alta velocidade e baixa tensão, capaz de operar até $800MHz$, possibilitando leitura de sinais com larguras de pulso maiores que $625ps$.

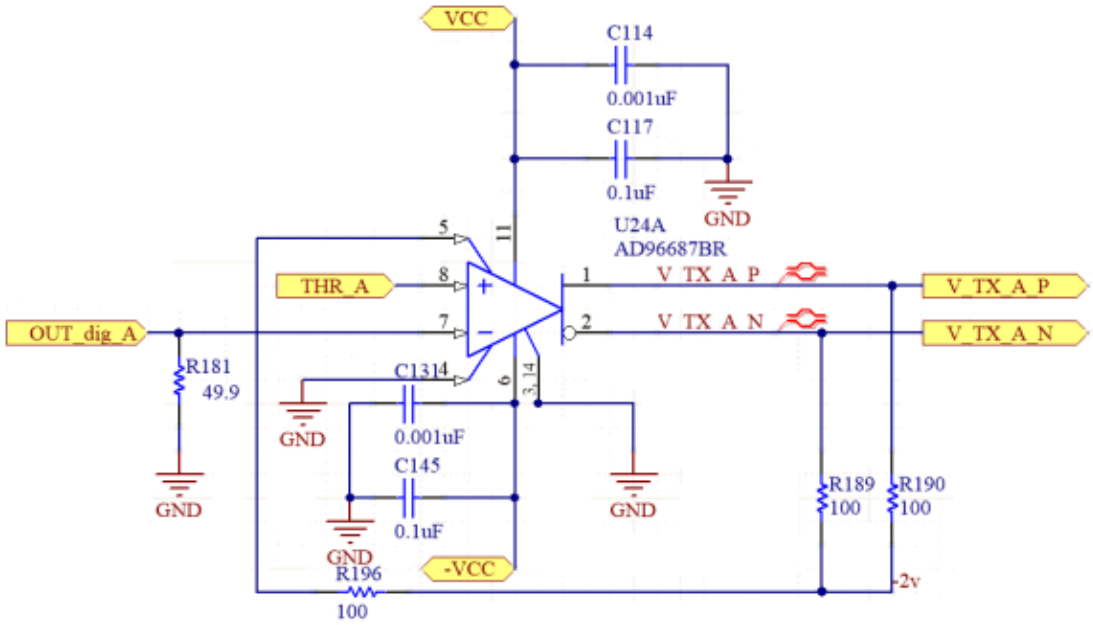


Figura 2.15: Desenho esquemático do circuito discriminador.

No comparador, o sinal na entrada inversora é subtraído do sinal de referência (*threshold*) na entrada não-inversora. Quando o resultado dessa subtração é positivo, um pulso *ECL* positivo é gerado. O comparador com saída diferencial *ECL* faz uso de duas linhas físicas para a transmissão do sinal de saída. Depois de enviado para o sistema de aquisição, esses sinais são subtraídos. A lógica *ECL* tem como características principais sua baixa amplitude de sinal, permitindo uma maior velocidade na transmissão de dados e menor suscetibilidade a ruídos decorrentes das linhas de transmissão e capacitâncias parasitas [28].

O ruído está presente em ambos os sinais diferenciais, positivo e negativo. Sendo assim, pelo ruído possuir a mesma fase, ao ocorrer a subtração do sinal diferencial positivo e negativo, ele é eliminado, como mostrado na Figura 2.16.

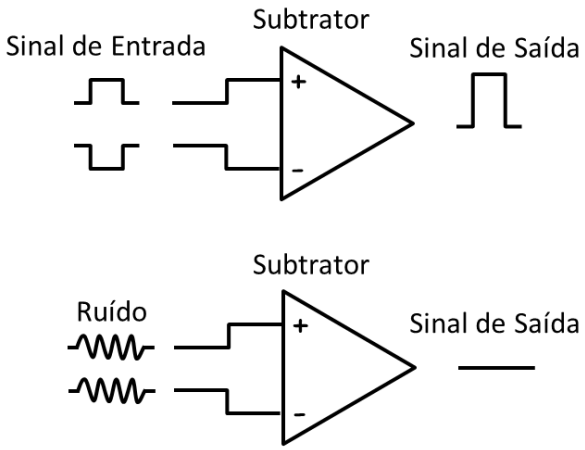


Figura 2.16: Comportamento do sinal diferencial e ruído ao passar por um amplificador subtrator (diferencial).

O valor de referência dos comparadores pode ser configurado pelo circuito de controle manual de *threshold* (CCMT), sem a necessidade de uma comunicação externa, ou pelo circuito de controle digital de *threshold* (CCDT), devendo ser configurado digitalmente através de uma comunicação serial. Sua escolha se dá através de um conector *header*, agindo como uma chave.

2.4.3 Circuito de Controle Manual de *Threshold* (CCMT)

O CCMT tem a função de fornecer a tensão de referência para o discriminador. Seu funcionamento se dá a partir de um divisor resistivo de tensão, que tem sua resistência ajustada através de um *trimpot* (resistor variável).

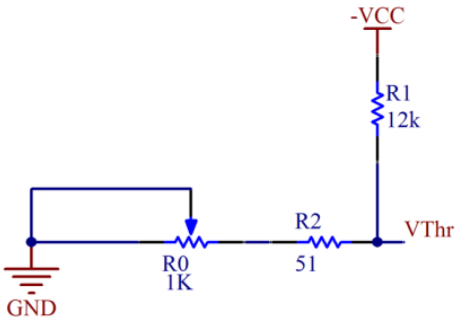


Figura 2.17: Desenho esquemático do CCMT utilizado no projeto.

Tendo $-V_{cc} = -5,00V$, $R1 = 12K\Omega$ e $R2 = 51\Omega$, a tensão V_{Thr} gerada por esse divisor, segue a Equação 2.1:

$$V_{Thr} = \frac{-5V \times (51\Omega + R0)}{51\Omega + R0 + 12k\Omega} \tag{2.1}$$

A resistência máxima de $1K\Omega$ do trimpot R0, foi calculada de modo a obter V_{Thr} operando numa faixa de -21 mV a -403 mV, segundo a Equação 2.1.

2.4.4 Circuito de Controle Digital de *Threshold* (CCDT)

O CCDT, assim como CCMT, possui a função de configurar uma tensão de referência para o circuito discriminador.

Seu circuito é composto por quatro *CIs AD5204* [29], cada um contendo quatro potenciômetros digitais de $10K\Omega$ com 256 posições, um microcontrolador *ATMEGA328* [30] e um circuito amplificador inversor com controle de ganho, que fornece a tensão máxima para o controle do *threshold* de cada discriminador, podendo variar de $0mV$ até $-417mV$.

O valor do *threshold* é alterado enviando um comando para seu microcontrolador através do protocolo serial *TTL RS-232*. Ao receber o comando referente à escrita de um novo valor de *threshold*, o valor é armazenado na memória *EPROM* do microcontrolador, e um registrador de 11 *bits* é enviado via protocolo *SPI* (*Serial Peripheral Interface*) para os potenciômetros digitais. O registrador possui os três primeiros *bits* reservados para o endereço do canal e os oito últimos à sua impedância correspondente. A habilitação do CI potenciômetro digital que receberá o registro de controle é feita através dos pinos $\overline{CS}$ (*chip select*), em nível baixo.

Ao final do bloco de controle do potenciômetro digital é acrescentado um divisor de tensão com sua tensão de *threshold* máxima (Vthr_Máx), podendo ser regulada de $0mV$ até $-417mV$. A resistência configurada no potenciômetro digital corresponderá a uma tensão de *threshold* associada. Na Figura 2.18, podemos observar o diagrama de controle dos potenciômetros *AD5204*.

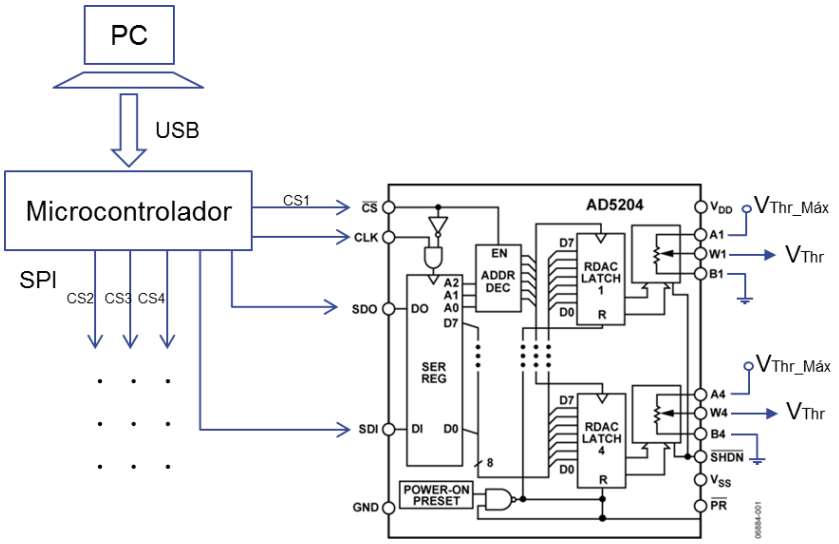


Figura 2.18: Diagrama de controle dos potenciômetros digitais

O *firmware* do microcontrolador foi elaborado de forma que o valor inicial da posição do potenciômetro digital sempre seja o último valor configurado pela interface, valor esse escrito e lido em sua memória *EPROM*. Isso permite que o uso do computador seja um instrumento opcional, caso o valor de *threshold* tenha a necessidade de ser alterado.

O potenciômetro digital utilizado possui uma resolução de 8 *bits* e resistência máxima de $10K\Omega$, possuindo uma resolução de 39Ω.

Comandos de Controle:

A comunicação com o microcontrolador se dá através do protocolo *TTL RS-232*, utilizando $+5V$ para nível alto e $0V$ para baixo. As palavras de controle são compostas por caracteres alfanuméricos enviados no padrão *ASCII*, com o intuito de ser de mais alto nível para facilitar sua utilização pelo usuário. Um conversor *USB-Serial* pode ser utilizado para sua comunicação com um PC.

Para se alterar um valor de *threshold* é enviado um caractere alfabético minúsculo, representando o canal que deseja ser alterado, e três caracteres numéricos na base decimal, representando a posição referente à resistência do potenciômetro digital (000 a 255). Cada canal tem seu caractere representativo seguindo a sequência crescente alfabética, ou seja, a letra *a* representa o canal 0, *b* o canal 1, assim sucessivamente.

Para a leitura da posição do potenciômetro digital correspondente ao valor de *threshold* é enviado o caractere alfabético, em maiúsculo, que representa o canal desejado, ou seja, a letra A representa o canal 0, B o canal 1, assim sucessivamente.

Calibração do Potenciômetro Digital

Para que o valor do *threshold* em bits corresponda ao seu valor real em tensão, uma calibração foi realizada. A calibração consistiu na elaboração de um programa em *C/C++* que enviasse ao potenciômetro digital, através do microcontrolador, os 256 valores de *threshold* possíveis. Essa tensão correspondente era medida com um multímetro Modelo *Fluke 8846A* [31] e seus valores armazenados no computador. A comunicação com o multímetro foi feita através do mesmo software descrito anteriormente, utilizando comunicação serial (*RS-232*). Código em *C/C++* que coleta os dados do multímetro para diferentes valores de *threshold*, pode ser observado no Apêndice B.

O valor da resistência do potenciômetro digital é obtido através da expressão fornecida pelo fabricante[29], apresentado na Equação 2.2:

$$R_{WB}(Dx) = \frac{Dx}{256} \times R_{BA} + R_W \tag{2.2}$$

Sendo R_{WB} a resistência associada ao valor Dx (valor decimal correspondente ao número de bits a ser enviado), R_{BA} o fundo de escala de $10K\Omega$, R_W a resistência interna do pino W, sendo esta de 45Ω, e 256 o número de posições do potenciômetro digital (*AD5204*).

Para fins práticos, consideramos R_W igual a zero (devido à impedância infinita do comparador). Logo, teremos o cálculo da tensão de *threshold* dado pela Equação 2.3:

$$VThr = \frac{R_{WB}(Dx)}{R_{BA}} \times VThr\_max \tag{2.3}$$

Sendo assim, através das Equações 2.2 e 2.3 teremos a Equação 2.4:

$$VThr = \frac{Dx}{256} \times VThr\_max \tag{2.4}$$

No gráfico da Figura 2.19, podemos observar que não existe uma linearidade perfeita

nos valores. Para resolver essa diferença, fizemos um ajuste para polinômio de 3º grau, Equação 2.5, e inserimos essa função de calibração no software de análise.

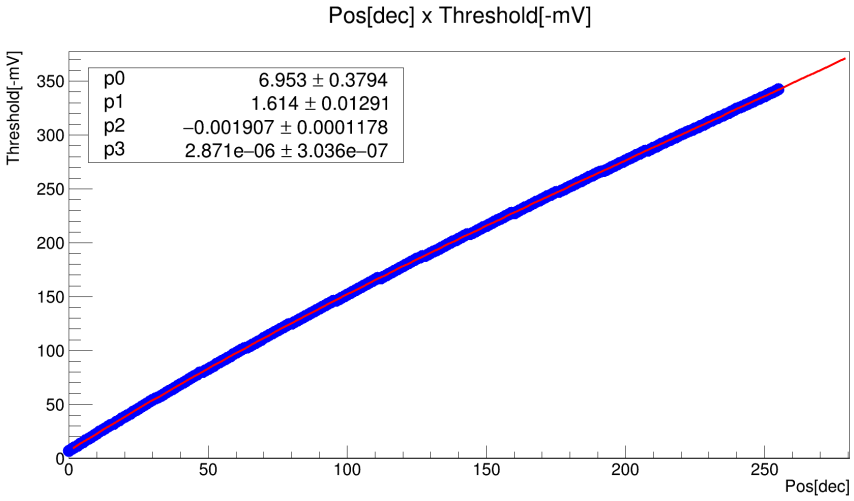


Figura 2.19: Gráfico de calibração do potenciômetro digital (AD5204).

Com base nos resultados, foi possível obter a seguinte função do valor de tensão real do *Threshold*:

$$V_{Thr}(D) = (3 \times 10^{-6})D^3 - 0,0019D^2 + 1,61D + 6,95 \tag{2.5}$$

Onde D é o valor do registrador correspondente à posição do potenciômetro digital. O software de análise pode ser observado no Apêndice C

2.5 Caracterização da Eletrônica

Para realizarmos a caracterização da eletrônica de *front-end*, identificando o ganho, *offset* e ruído equivalente de cada canal da eletrônica, foi injetado na entrada do amplificador um sinal similar ao da *MaPMT*.

Nos testes, foram utilizados os seguintes equipamentos:

- Gerador de função Tektronix AWG7082C (8GS/s) [32];
- FONTE DC - ICEL PS6100 [33];

- Osciloscópio - Tektronix DPO70404C Digital Phosphor Oscilloscope (25GS/s - 4GHz) [34];
- CAEN V1495 - General Purpose VME Board [35];
- CAEN V1718 - Controller (VME) [36].

O método consiste em injetar um pulso de tensão com diferentes amplitudes, -5, -7 e -10 mV, sendo o último compatível ao sinal proveniente da $MaPMT$ na incidência de pouca luz. Ao injetarmos sinais de diferentes amplitudes, é possível verificar o comportamento do circuito para a faixa dinâmica característica de uma $MaPMT$, para baixa incidência de luz, e a linearidade dos parâmetros a serem medidos. Na Figura 2.20 é possível observar um sinal de pulso de tensão gerado pelo gerador de função (roxo) e seu sinal amplificado (azul).

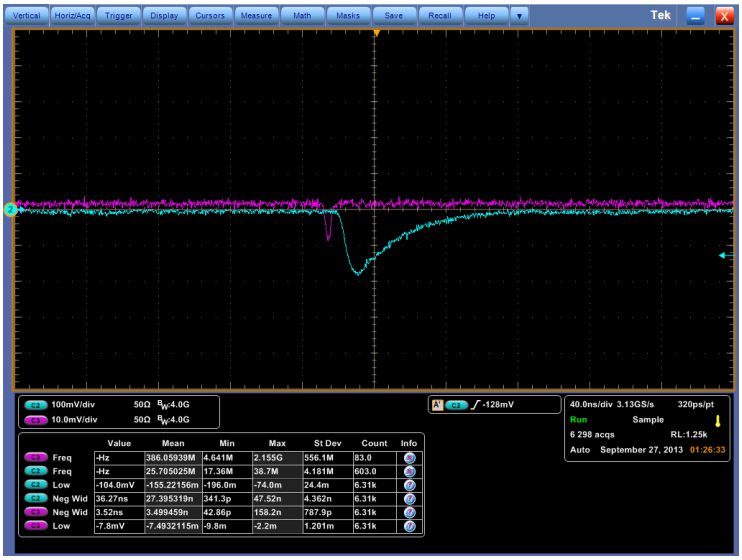


Figura 2.20: *Shaper* do sinal injetado, utilizando a escala de $10mV/div$, e do sinal amplificado pelo circuito amplificador, utilizando a escala de $100mV/div$.

Os pulso foram gerados numa frequência de $10KHz$ e largura de $\sim 2ns$, amplificados e discriminados pelo circuito eletrônico por diferentes valores de *threshold*. A saída discriminada na lógica *ECL* é enviada para o *FPGA Altera Cyclone I* do módulo V1495, onde são feitas as contagens dos pulsos que posteriormente serão lidas e armazenadas por um software de aquisição de dados. A comunicação entre o *PC* e o Módulo *V1495* se dá através do módulo *V1718* pelo barramento *VME*. Na Figura 2.21 podemos ver o diagrama do teste realizado:

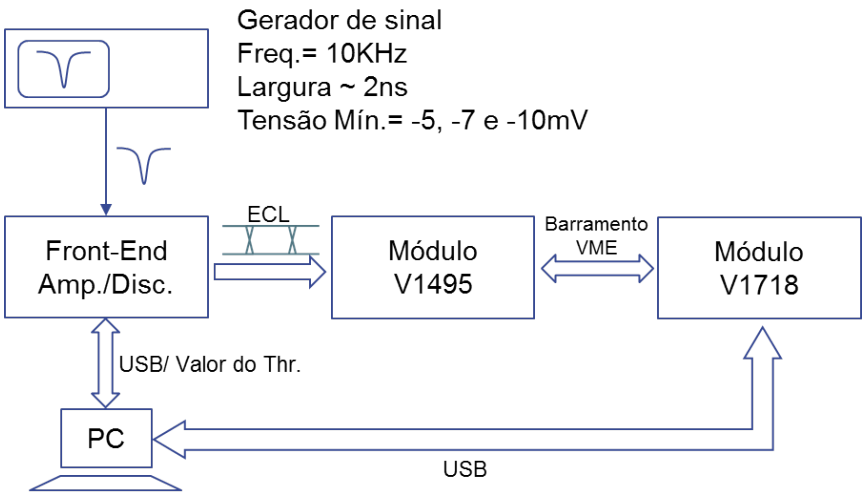


Figura 2.21: Diagrama do *setup* de caracterização.

Os valores das contagens e seu respectivo *threshold* foram armazenados através de um software elaborado em *C/C++* e um *script* Linux, que o executa diversas vezes para diferentes valores de *threshold*. Os códigos utilizados podem ser observado no Apêndice E.

A Figura 2.22 mostra as regiões de diferentes valores de *threshold* para um sinal analógico típico de detecção.

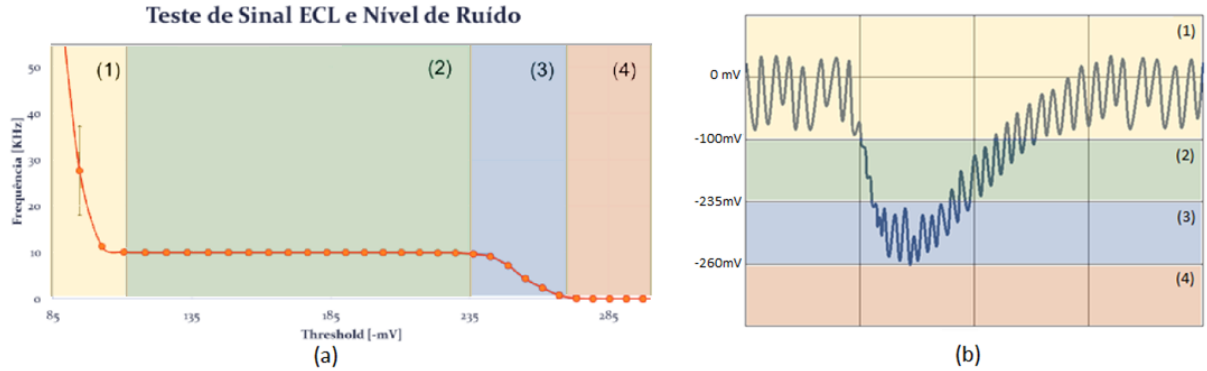


Figura 2.22: (a) Gráfico de frequência versus *threshold*, incluindo seu erro padrão. (b) Imagem demonstrativa de um pulso amplificado, mostrando as regiões de diferentes valores de *threshold*.

Para melhor estimar os valores coletados e fazer uma análise quantitativa dos sinais, um software de análise de dados foi desenvolvido em linguagem C/C++, utilizando a ferramenta Root [37], desenvolvida pelo *CERN*. O código referente às análises, pode ser visto no Apêndice D. Foi realizado um ajuste pela função de Curva S, baseada na integral

da função de distribuição gaussiana de probabilidade, dada por [38]:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.6}$$

Onde μ é a média, σ o desvio padrão e x a grandeza em questão.

Integrando teremos:

$$D(x) = \frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \tag{2.7}$$

Os gráficos de *threshold* vs contagem para diferentes valores de amplitude injetada, referente ao canal 1 da eletrônica de front-end, podem ser observados nas Figuras 2.23, 2.24 e 2.25, sendo sua função de ajuste dada por:

$$F_{Ajust.S} = p0 \times \frac{1}{2}\left[1 - erf\left(\frac{x-p1}{p2\sqrt{2}}\right)\right] \tag{2.8}$$

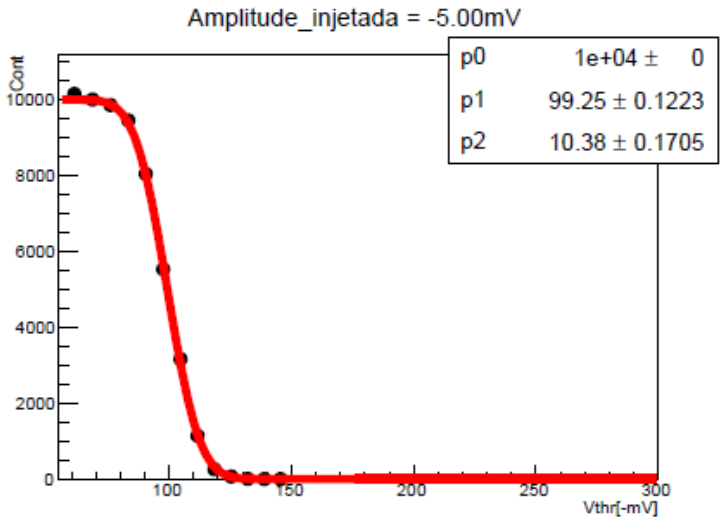


Figura 2.23: Gráfico de contagem x *threshold* para pulso de entrada de $-5mV$.

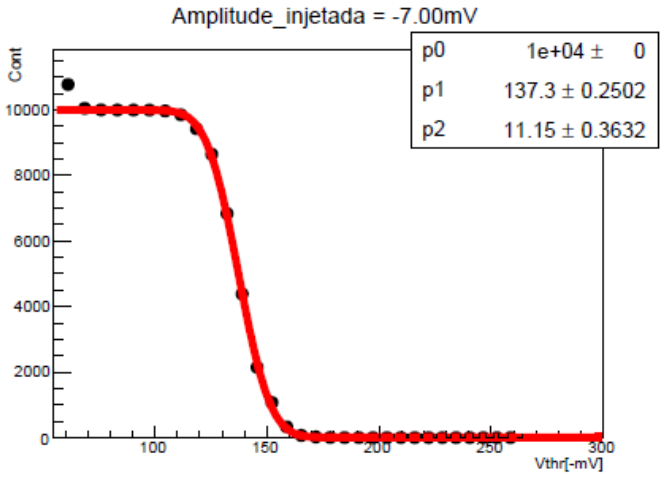


Figura 2.24: Gráfico de contagem x *threshold* para pulso de entrada de $-7mV$.

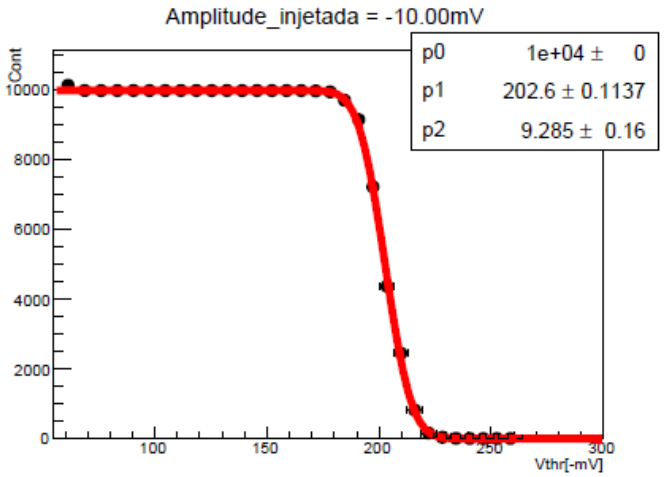


Figura 2.25: Gráfico de contagem x *threshold* para pulso de entrada de $-10mV$.

Os parâmetros dos gráficos para diferentes amplitudes injetadas são dados por:

- Parâmetro p0: informa o valor máximo de contagem, fixado em 10000 (frequência da fonte);
- Parâmetro p1: o valor médio da amplitude do sinal;
- Parâmetro p2: o desvio padrão (σ) do ruído.

Sabemos que três desvios padrões cobrem cerca de $99,7\%$ do conjunto de probabilidade, multiplicando cada desvio padrão por 3, teremos uma amplitude de ruído de até $31,14 \pm 0,51mV$ para um sinal de entrada de $-5mV$, $33,45 \pm 1,09mV$ para um sinal de entrada de $-7mV$ e $27,86 \pm 0,48mV$ para um sinal de entrada de $-10mV$, sendo estes tanto negativo quanto positivo. Dessa forma, teremos a amplitude máxima do ruído de $62,28 \pm 1,02mV$, $66,90 \pm 2,18mV$ e $55,71 \pm 0,96mV$ pico a pico, respectivamente, para este canal.

Na Figura 2.26 podemos observar o gráfico referente à média de todos os canais, apresentando o valor médio da amplitude do sinal de saída versus a amplitude injetada. Os valores se aproximam a uma função do 1º Grau, demonstrando que a amplitude do sinal de saída é linearmente proporcional à amplitude do sinal de entrada, onde o coeficiente angular da função representa o ganho e a constante seria o *offset* do circuito.

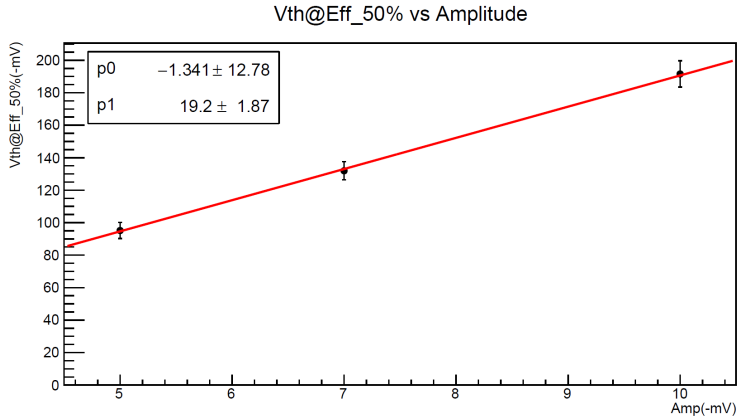


Figura 2.26: Gráfico de valor médio da amplitude do sinal versus Amplitude Injetada.

Na Figura 2.27 foi realizado um gráfico referente à média de todos os canais, contendo o valor médio da amplitude do sinal de saída versus o desvio padrão (σ) do ruído, também chamado de ruído equivalente (ENV). Assumimos que o ruído não sofre influência

da amplitude do sinal de entrada, logo, aplicamos seu ajuste segundo à função polinomial de grau zero.

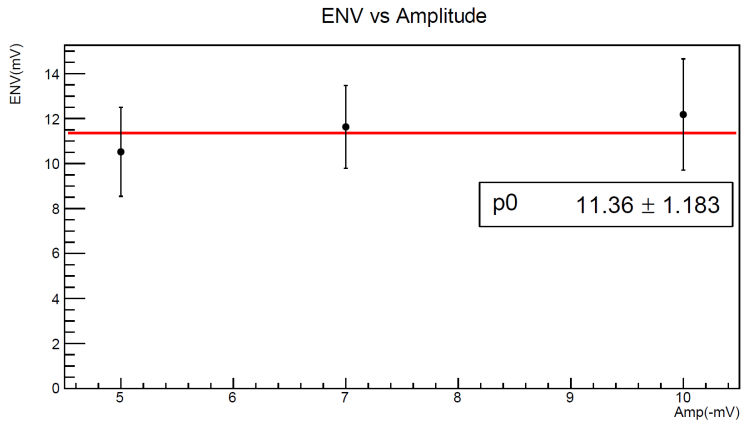


Figura 2.27: Gráfico de desvio padrão (σ) do ruído versus valor médio da amplitude do sinal.

Na Tabela 2.1 são apresentados os resultados com seus respectivos desvios padrões.

Tabela 2.1: Tabela de caracterização da eletrônica de *front-end*.

Canal	ENV (σ)	*Offset* $[mV]$	Ganho Médio
0	$10,92 \pm 0,74$	$7,42 \pm 0,33$	$21,38 \pm 0,04$
1	$10,27 \pm 0,94$	$4,73 \pm 0,27$	$20,71 \pm 0,03$
2	$10,47 \pm 1,22$	$0,33 \pm 0,23$	$19,49 \pm 0,03$
3	$9,90 \pm 1,19$	$-0,86 \pm 0,22$	$19,15 \pm 0,03$
4	$10,35 \pm 0,54$	$1,47 \pm 0,23$	$19,37 \pm 0,03$
5	$9,48 \pm 0,40$	$2,34 \pm 0,24$	$19,52 \pm 0,03$
6	$10,83 \pm 1,29$	$1,35 \pm 0,26$	$19,49 \pm 0,03$
7	$13,91 \pm 1,79$	$1,62 \pm 1,47$	$19,26 \pm 0,16$
8	$17,01 \pm 1,60$	$4,09 \pm 0,93$	$18,03 \pm 0,16$
9	$13,59 \pm 1,61$	$6,83 \pm 1,45$	$18,88 \pm 0,21$
10	$10,74 \pm 1,52$	$6,44 \pm 0,45$	$18,03 \pm 0,06$
11	$11,49 \pm 1,75$	$7,73 \pm 0,38$	$19,69 \pm 0,05$
12	$9,71 \pm 1,02$	$-1,53 \pm 0,21$	$19,17 \pm 0,03$
13	$10,61 \pm 0,96$	$-2,71 \pm 0,22$	$19,29 \pm 0,03$

14	$11,26 \pm 1,44$	$0,12 \pm 0,29$	$18,92 \pm 0,04$
15	$12,52 \pm 1,28$	$-2,66 \pm 0,69$	$18,93 \pm 0,12$
Média	$11,44 \pm 1,96$	$2,29 \pm 3,53$	$19,33 \pm 0,83$

Podemos observar que o canal 8 possui um ruído acima dos demais, sendo este de aproximadamente $102mV$ pico a pico (6σ) na saída do seu amplificador. Portanto, sinais de entrada de até $-5mV$ são confundidos com seu ruído intrínseco, não podendo ser discriminados.

2.6 Sistema Mecânico

Devido à necessidade de uma estrutura que fosse completamente vedada da luz, foi projetada e construída sob encomenda uma caixa de polietileno preto de alta densidade para ser usada como suporte e proteção dos detectores dos experimentos CREAT1 e CREAT2. A caixa foi desenvolvida de forma que seu encaixe com a tampa não criasse qualquer vão que permitisse a entrada de luz. Uma espécie de labirinto foi projetado em sua parte frontal, de modo que dificultasse a entrada de luz que viesse através de seus conectores elétricos. Uma imagem tridimensional com corte longitudinal pode ser observada na Figura 2.28.

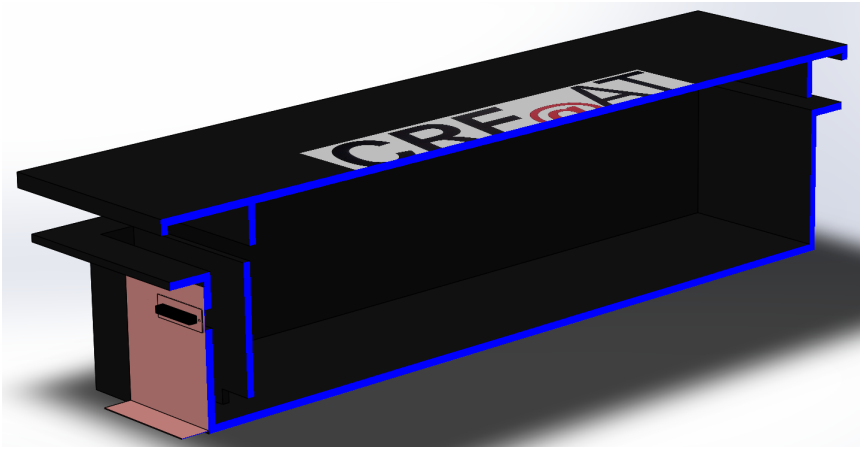


Figura 2.28: Desenho em três dimensões da caixa do CREAT 1 e 2

A Figura 2.29 e 2.30 apresentam o projeto em duas dimensões dessa estrutura.

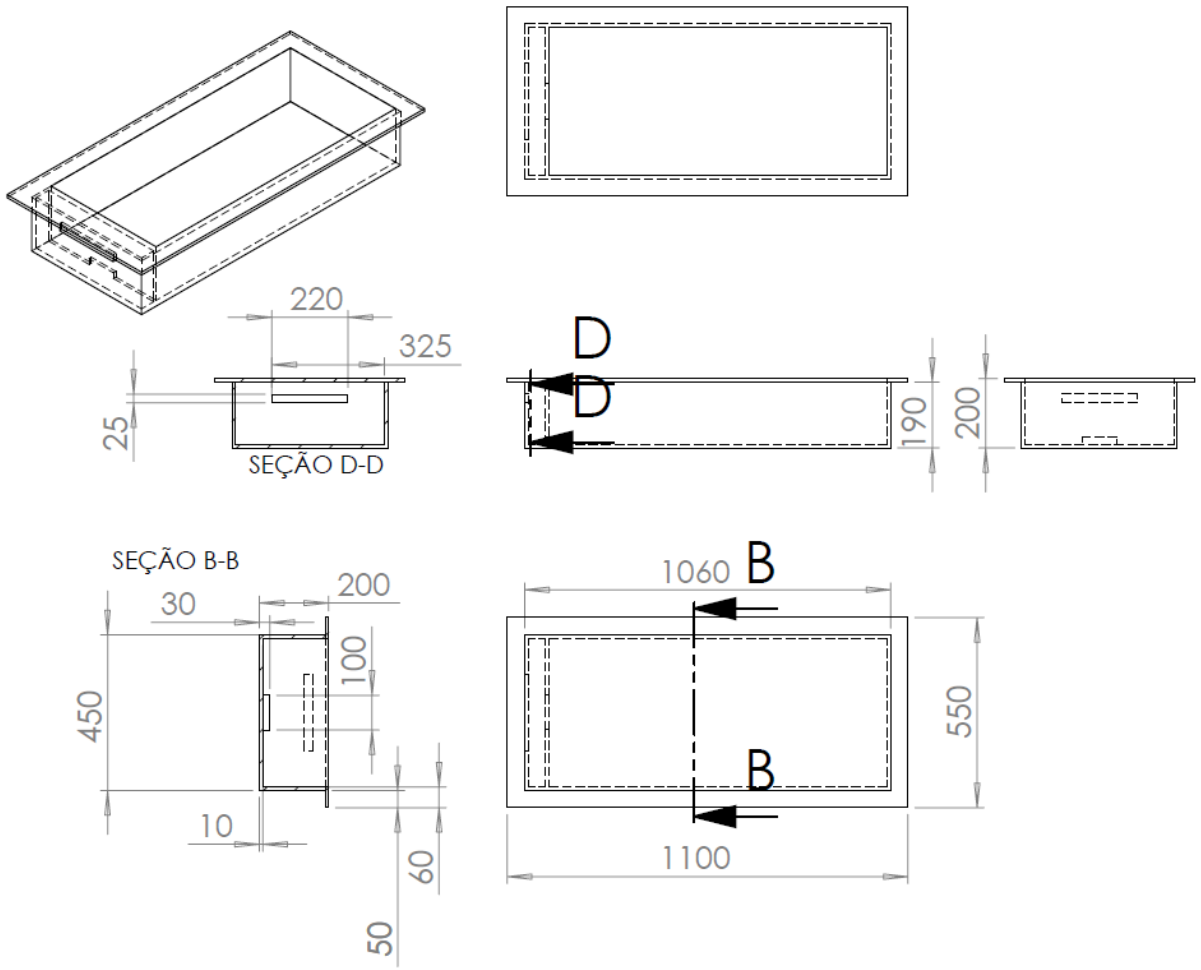


Figura 2.29: Desenho cotado em duas dimensões da caixa do CREAT 1 e 2

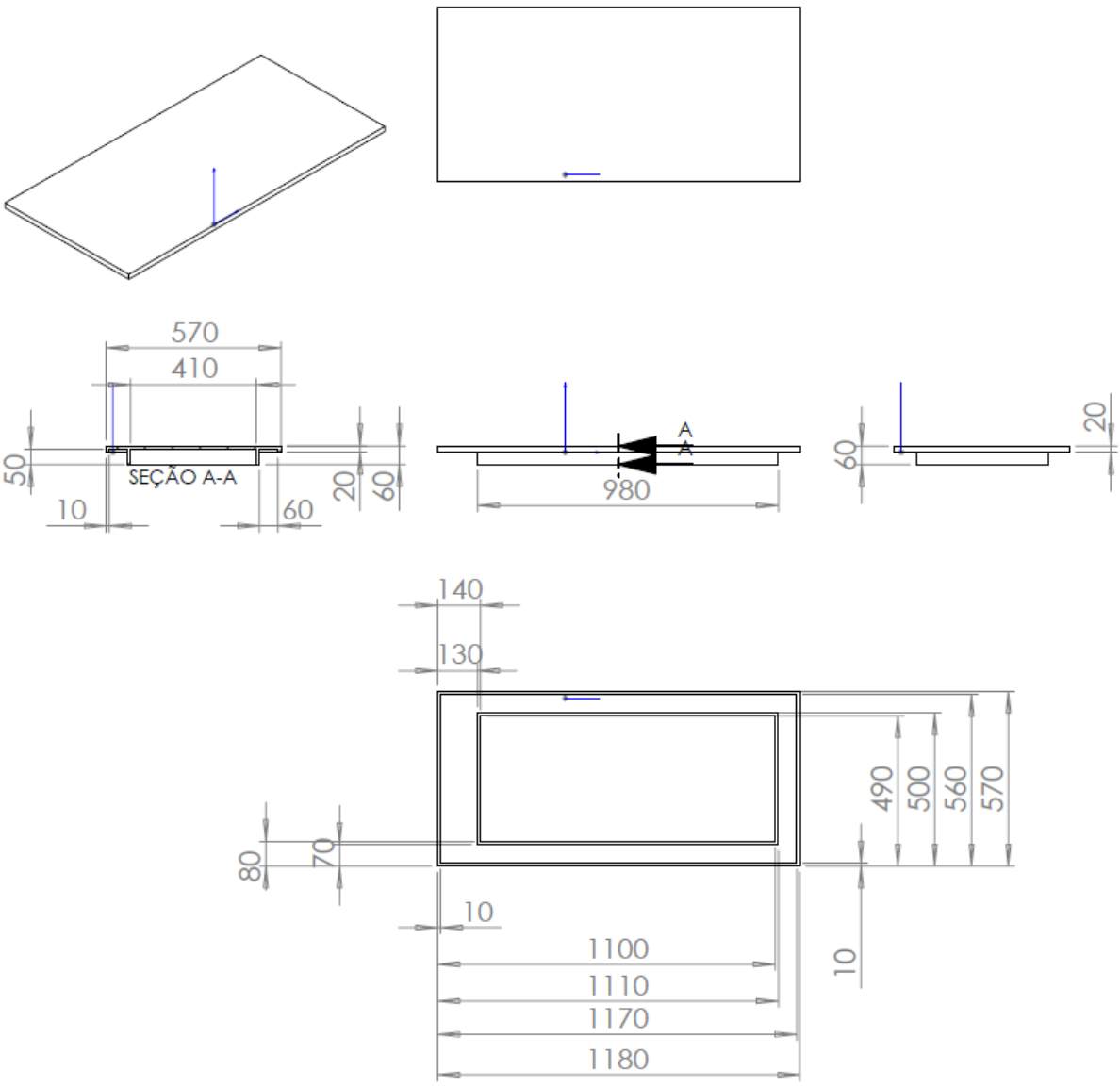


Figura 2.30: Desenho cotado em duas dimensões da tampa do CREAT 1 e 2

Cada tira cintiladora possui uma fibra WLS que necessita estar perfeitamente alinhada às células da MaPMT. O desalinhamento pode acarretar em perda de eficiência ou *crosstalk* óptico, comprometendo o funcionamento do detector. Sendo assim, uma peça de suporte foi projetada para garantir este acoplamento.

A peça é composta por uma máscara que alinha as 16 fibras com as células da fotomultiplicadora e dois suportes que mantém todo o conjunto unido. Uma imagem tridimensional dessa estrutura pode ser observada na Figura 2.31.

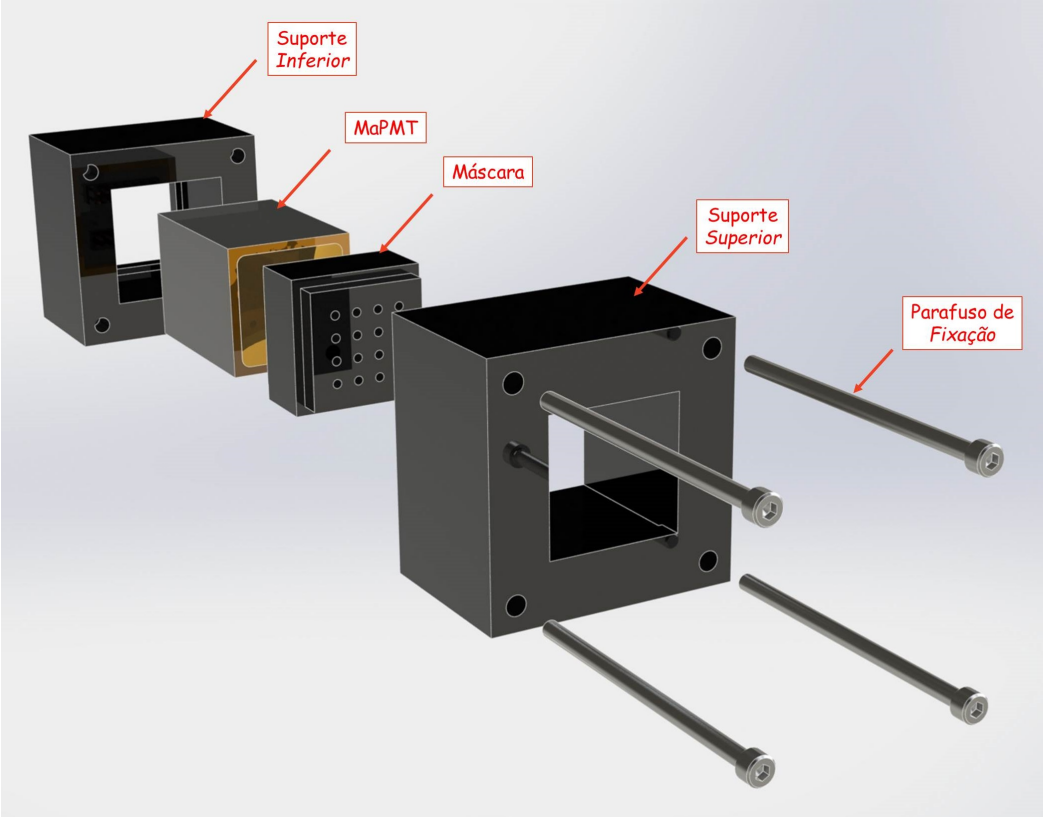


Figura 2.31: Desenho em três dimensões da peça de suporte da *MaPMT* e fibras *WLS*.

No detector do CREAT1, foi utilizada uma máscara usinada de latão, cujo coeficiente de expansão linear é $17,9 \times 10^{-6} K^{-1}$ [39], e no do CREAT2 foi utilizado o mesmo material da caixa, o polietileno de alta densidade com coeficiente de expansão linear $1,3 \times 10^{-4} K^{-1}$ [40]. Já o suporte das tiras cintiladoras, assim como a peça de suporte do conjunto *MaPMT*+fibra *WLS*, foi desenvolvido em alumínio, garantindo o posicionamento e fixação de todos os elementos do detector. A figura 2.32 apresenta uma imagem detalhada dessa estrutura no detector do CREAT1.

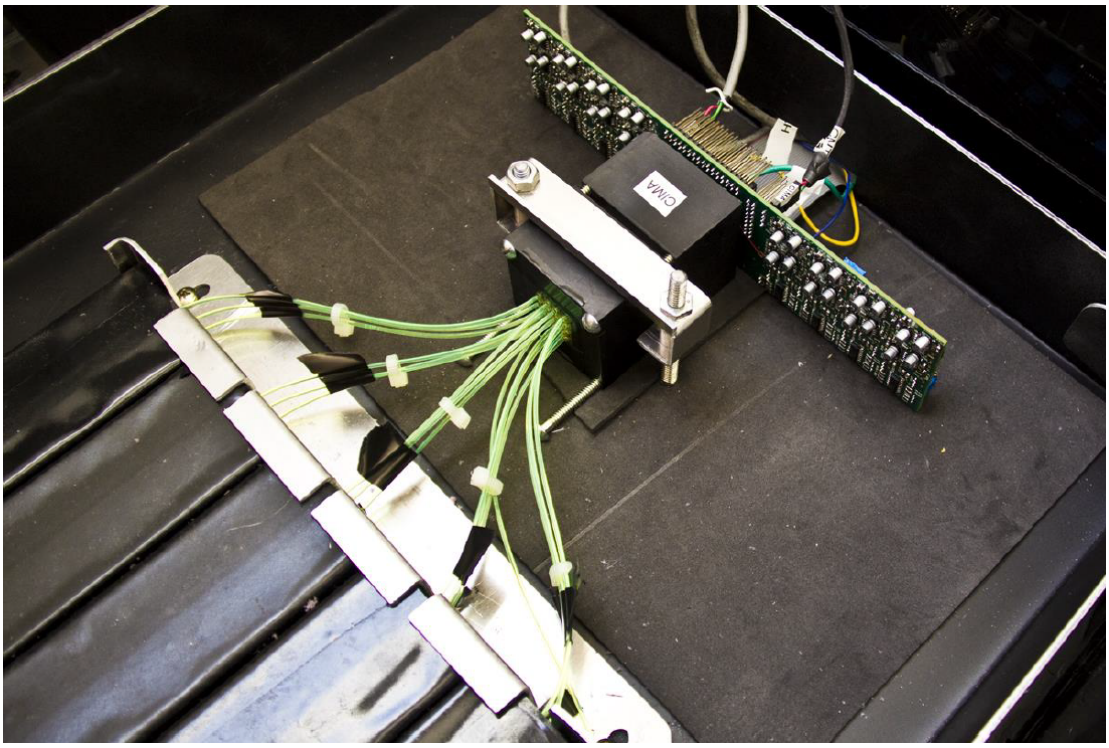


Figura 2.32: Estrutura de alinhamento entre as fibras WLS e a fotomultiplicadora [20].

2.7 Simulação Monte Carlo

Devido a geometria e organização das tiras cintiladoras dos detectores, parte das partículas que passam pelos planos de detecção não podem ser detectadas, pois alguns ângulos de incidência das mesmas não permite. Para considerar o efeito geométrico nos cálculos do fluxo de raios cósmicos, uma simulação estatística se faz necessária.

O método Monte Carlo, consiste em algoritmos computacionais baseados em amostragens aleatórias repetidas para se obter resultados numéricos. Nas simulações, são gerados diversos traços com diferentes ângulos de incidência de θ e ϕ em coordenadas esféricas, segundo o modelo de produção muônica teórico. Depois, é verificado a probabilidade de uma partícula passar pelo plano de detecção, sem que possa ser detectada devido a geometria do detector. O modelo de produção teórico utilizado nas simulações é de $cos(\theta)^2$ [41].

A constante de probabilidade calculada deve ser considerada no cálculo do fluxo de raios cósmicos, juntamente com a eficiência de cada plano de detecção.

2.8 CREAT 1 - Versão Piloto

A versão piloto do detector foi enviada para o módulo Criosfera 1 em outubro de 2014, tendo seu *firmware* e *softwares* de aquisição desenvolvidos pelo, até então, estudante de engenharia Ulisses Carneiro. O projeto detalhado desse detector está documentado em seu trabalho de conclusão de curso de Engenharia Eletrônica do CEFET/RJ, realizado em 2015 [20]. O detector contava com uma MaPMT de 16 canais e 15 tiras cintiladoras organizadas como uma matriz 5x3, como mostrada na Figura 2.33. A vantagem da configuração 1x1x1, em que três tiras cintiladoras ($49,5cm \times 5cm$) ficam alinhadas uma em cima da outra, está na disponibilidade de cinco medidas independentes de fluxo de raios cósmicos, melhorando, assim, o erro sistemático do experimento. Cada conjunto de três cintiladores foi denominado como detector A, B, C, D e E.

Figura 2.33: Experimento CREAT1 dentro do módulo Criosfera 1.

A lógica de coincidência utilizada nesta configuração emite um pulso lógico alto na ocorrência de um evento em qualquer combinação de coincidência entre os três planos de cintiladores, como mostrado na Figura 2.34.

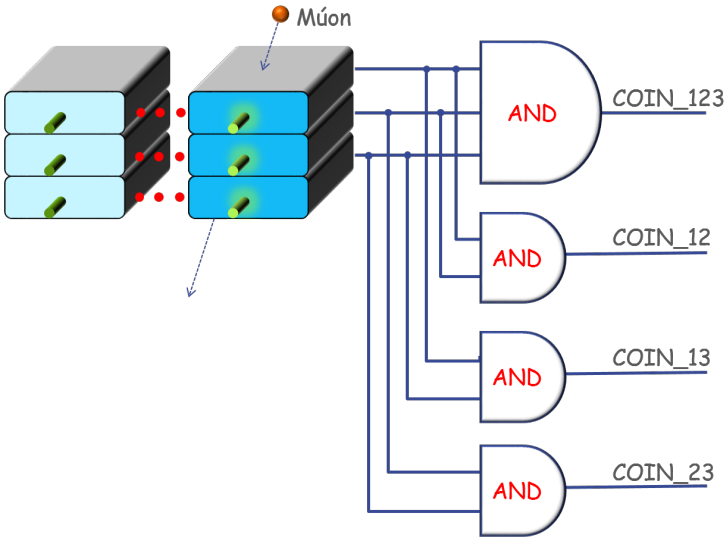


Figura 2.34: Lógica de coincidência da configuração 1x1x1.

Os pulsos gerados são armazenados em registros de contagem em um certo tempo de aquisição pré-definido. Essas contagens nos fornecem todas as informações para os cálculos de eficiência e fluxo.

A eficiência de cada canal do detector se refere à quantidade de eventos detectados dentre todos os eventos que de fato ocorreram. O cálculo referente à essa medida é realizado dividindo a contagem da coincidência tripla do conjunto pela contagem da coincidência dupla entre os outros dois canais restantes, como exemplificado na Equação 2.9:

$$Eff_2 = \frac{CONT_{1\&2\&3}}{CONT_{1\&3}} \tag{2.9}$$

Onde Eff_2 se refere a eficiência da tira do meio, $CONT_{1\&2\&3}$, a contagem de eventos que passaram pelas três tiras, e $CONT_{1\&3}$ a contagem de eventos que passaram simultaneamente pelas tiras superior e inferior.

Nesta configuração, não há possibilidade de uma partícula passar pelas tiras superior e inferior sem passar pela tira do meio, fazendo com que o cálculo da medida de eficiência dessa tira seja bem preciso. Entretanto, sua geometria permite que haja a possibilidade de uma partícula passar apenas pela tira superior e mediana ou inferior e mediana, dependendo do seu ângulo de incidência e posicionamento, o que torna a medida de eficiência das tiras superiores e inferiores menos precisa.

De acordo com a simulação Monte Carlo, o fator de correção geométrica da eficiência

de cada plano de detecção é de 0,978 para o primeiro plano, 1 para o segundo plano, e 0,957 para o terceiro plano.

O cálculo do fluxo é obtido a partir da contagem da coincidência dos planos 1 e 2, dividido pela eficiência de cada um dos dois planos (já corrigidos), tempo de aquisição, geometria e área de incidência, como mostrado na Equação 2.10:

$$Fluxo = \frac{CONT_{1\&2}}{Eff_1 \times Eff_2 \times T_{AQ} \times GEOM \times AREA} \tag{2.10}$$

Devido à largura e espessura de cada tira, dependendo do seu ponto de incidência, sua geometria não permite a detecção de partículas com ângulos maiores que 78,7°, necessitando de uma correção geométrica baseada na simulação de Monte Carlo. Através da simulação, calculamos que 95,8% dos raios cósmicos que passam conseguem ser detectados.

Embora haja um conjuntos de cinco medidas independentes de detecção, esta configuração possui algumas desvantagens que nos levou à sua mudança de configuração e *upgrade* para o detector do CREAT2, são elas:

- Área de incidência reduzida, o que nos dá uma menor estatística;
- Cálculo de eficiência impreciso, devido à organização das tiras;
- Apenas dois planos de coincidência para cálculo do fluxo e um auxiliar para o cálculo de eficiência;

2.8.1 Resultados

Nos períodos em que o experimento CREAT1 ficou tomando dados, o sistema de geração eólica de energia do módulo Criosfera I não funcionou como o esperado, inviabilizando o funcionamento ininterrupto do experimento. Assim, foi preciso economizar energia das baterias do módulo, que eram carregadas apenas pelos painéis solares, de forma que se mantivesse funcionando o ano inteiro. Para isso, durante o primeiro ano, optou-se em coletar apenas três medidas de fluxo por dia, cada uma com dez minutos de tempo de aquisição e alta tensão de -750V. Já nos demais anos, optamos que o experimento funcionasse de forma ininterrupta até que as baterias acabassem, o que nos deixou sem dados durante alguns meses, principalmente no período de inverno, onde não havia luz para os

painéis solares recarregarem as baterias. Vale ressaltar que, a partir de 2016, a alta tensão do módulo foi alterada para -700V, com intuito de verificar o comportamento do detector para tal valor. Vale ressaltar que o valor da alta tensão de alimentação das fotomultiplicadoras está diretamente ligado ao seu ganho, comprometendo a eficiência do detector.

Os resultados da análise da taxa de contagem, fluxo e eficiência de cada plano de detecção estão apresentados nas figuras a seguir. A nomenclatura utilizada nos gráficos para o detector A e válida para os demais, é:

- hA12: Histograma referente à taxa de contagem da coincidência entre os planos 1 e 2 do detector A;
- hA13: Histograma referente à taxa de contagem da coincidência entre os planos 1 e 3 do detector A;
- hA23: Histograma referente à taxa de contagem da coincidência entre os planos 2 e 3 do detector A;
- hAn1: Histograma referente à taxa de contagem do plano 1 do detector A;
- hAn2: Histograma referente à taxa de contagem do plano 2 do detector A;
- hAn3: Histograma referente à taxa de contagem do plano 3 do detector A;
- hA123: Histograma referente à taxa de contagem da coincidência tripla dos planos do detector A;
- hfluxA: Histograma referente ao fluxo do detector A;
- heffA1: Histograma referente à eficiência do plano 1 do detector A;
- heffA2: Histograma referente à eficiência do plano 2 do detector A;
- heffA3: Histograma referente à eficiência do plano 3 do detector A;

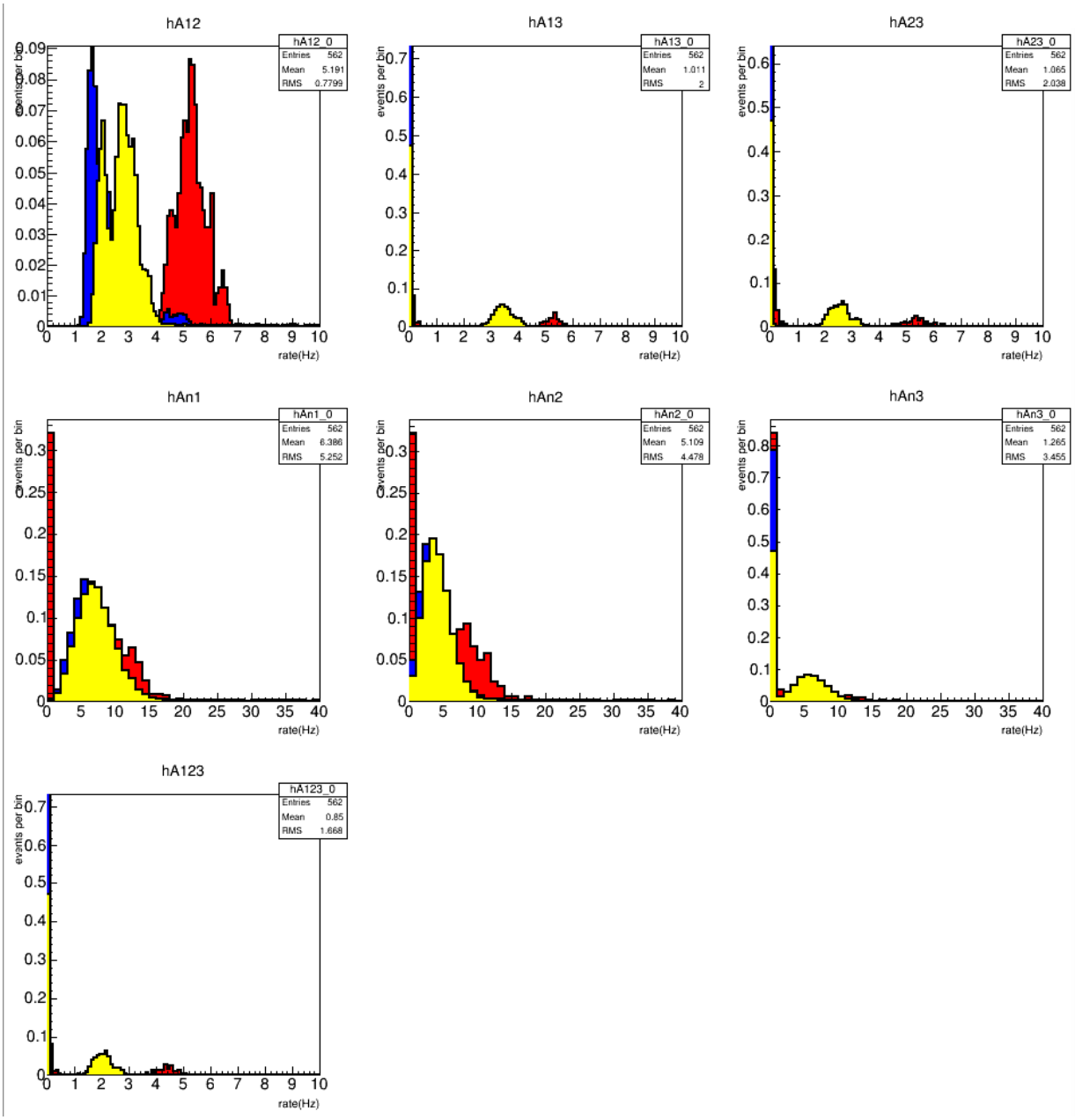


Figura 2.35: Histogramas referentes à taxa de contagem do conjunto A de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

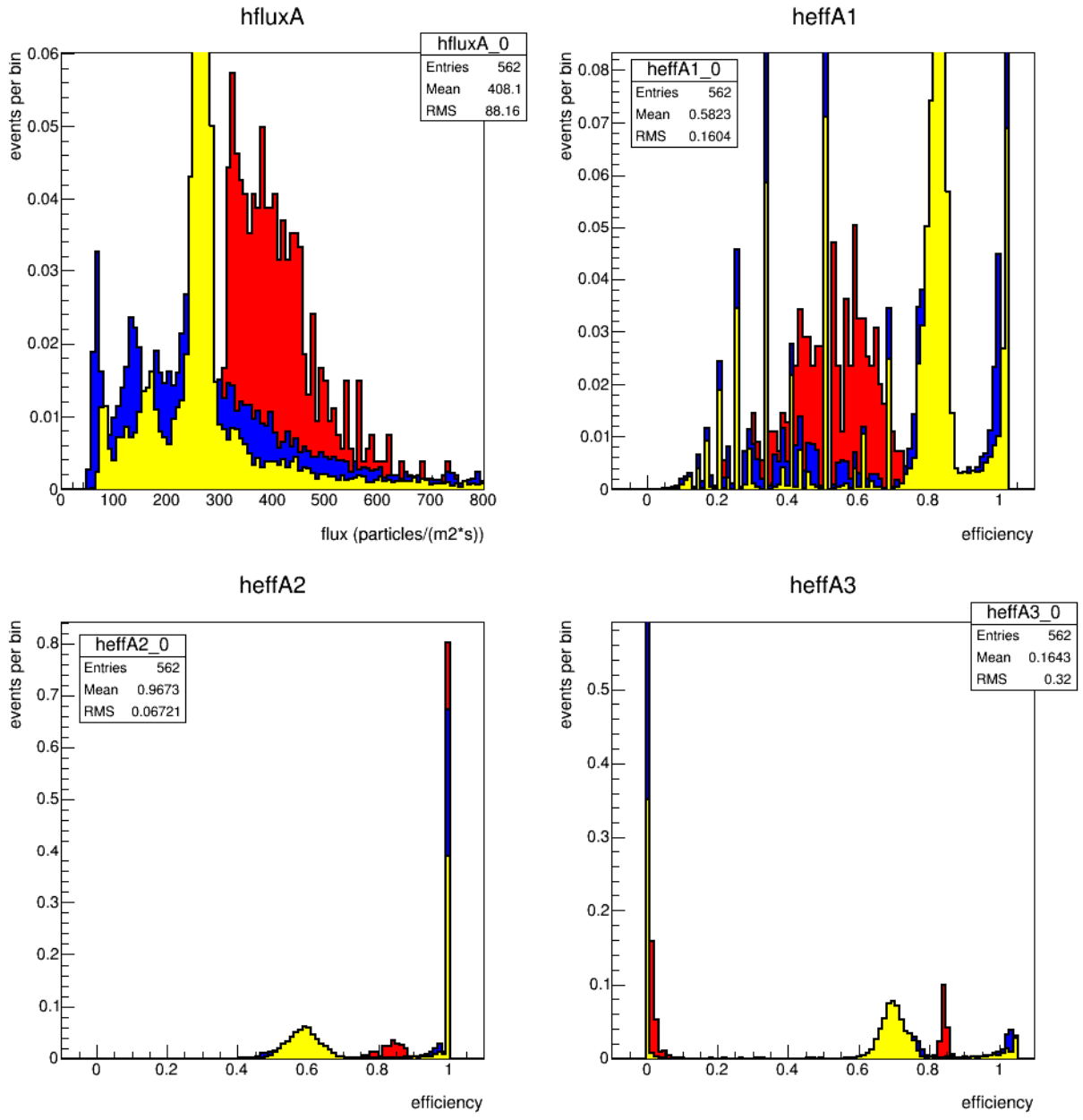


Figura 2.36: Histogramas referentes ao fluxo e eficiências do conjunto A de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

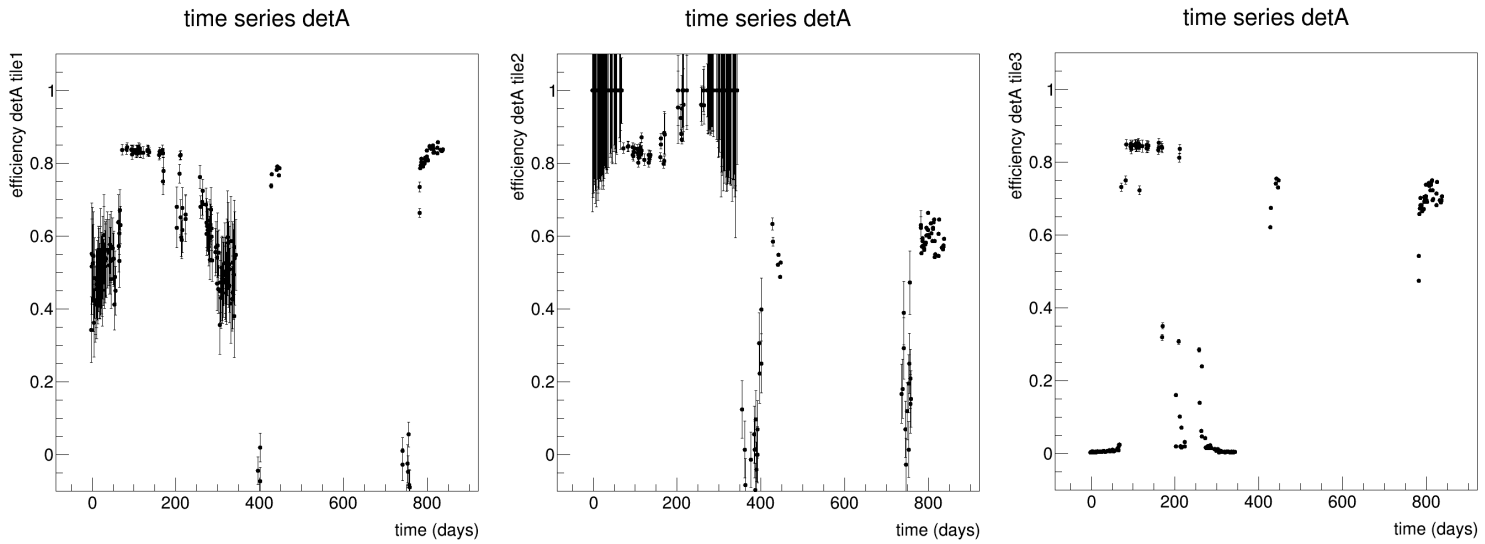


Figura 2.37: Série temporal da eficiência do conjunto A de detecção para as missões de 2015, 2016 e 2017.

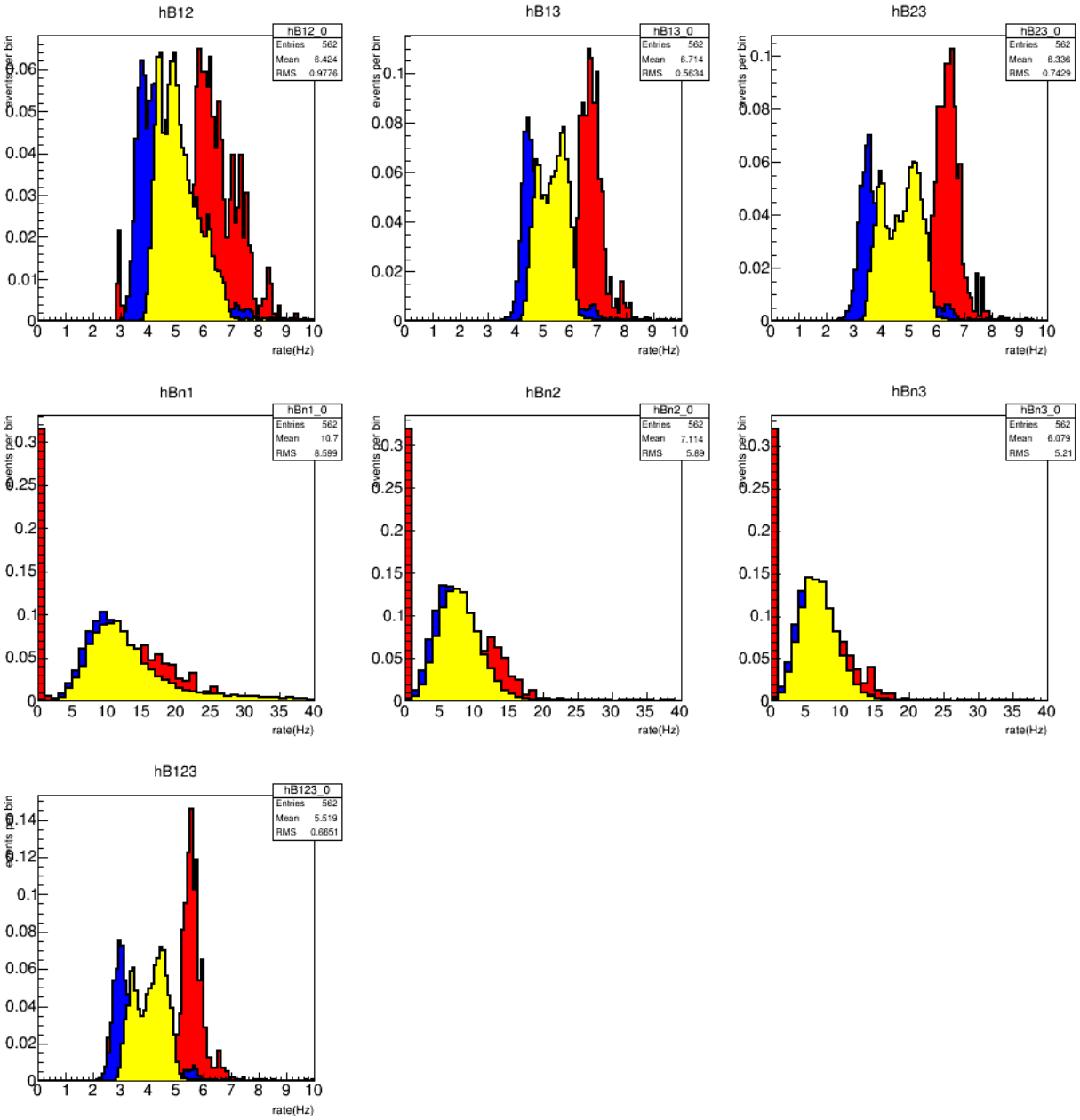


Figura 2.38: Histogramas referentes à taxa de contagem do conjunto B de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

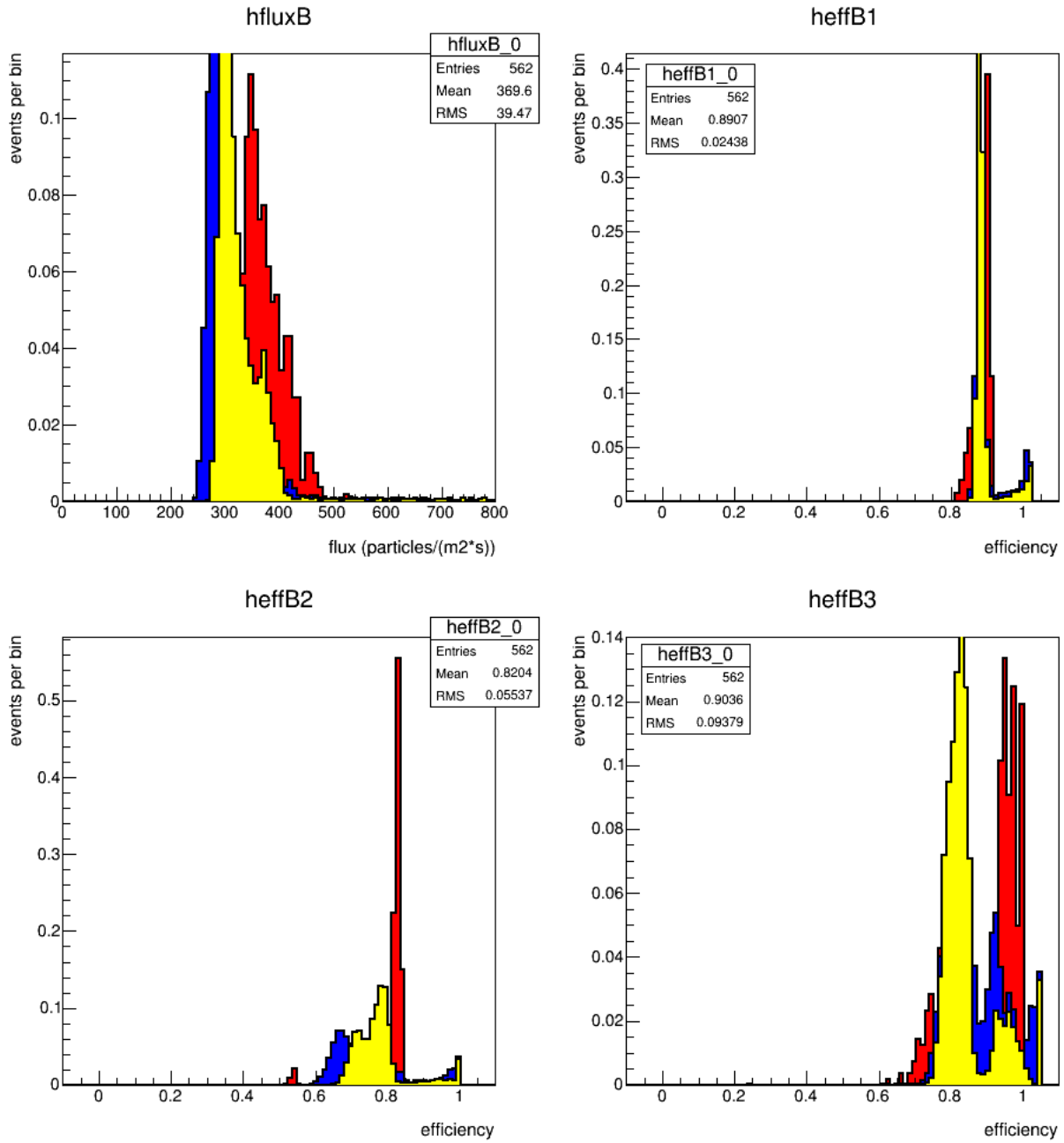


Figura 2.39: Histogramas referentes ao fluxo e eficiências do conjunto B de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

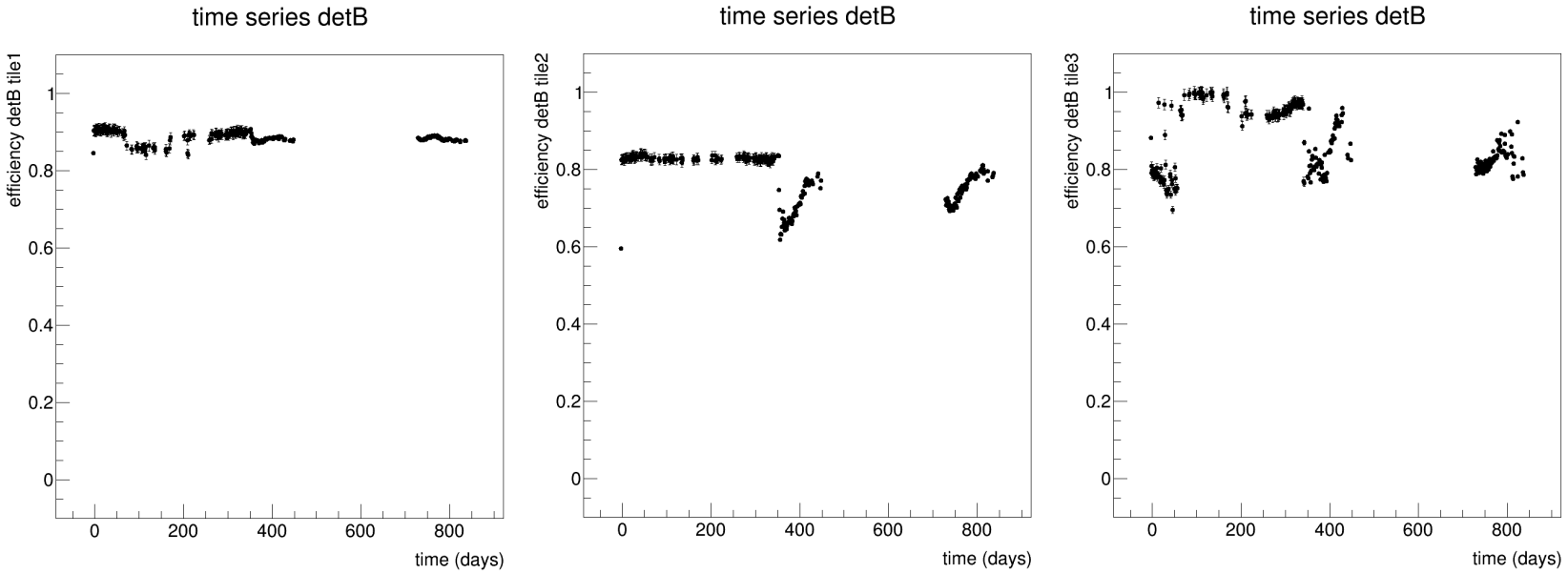


Figura 2.40: Série temporal da eficiência do conjunto B de detecção para as missões de 2015, 2016 e 2017.

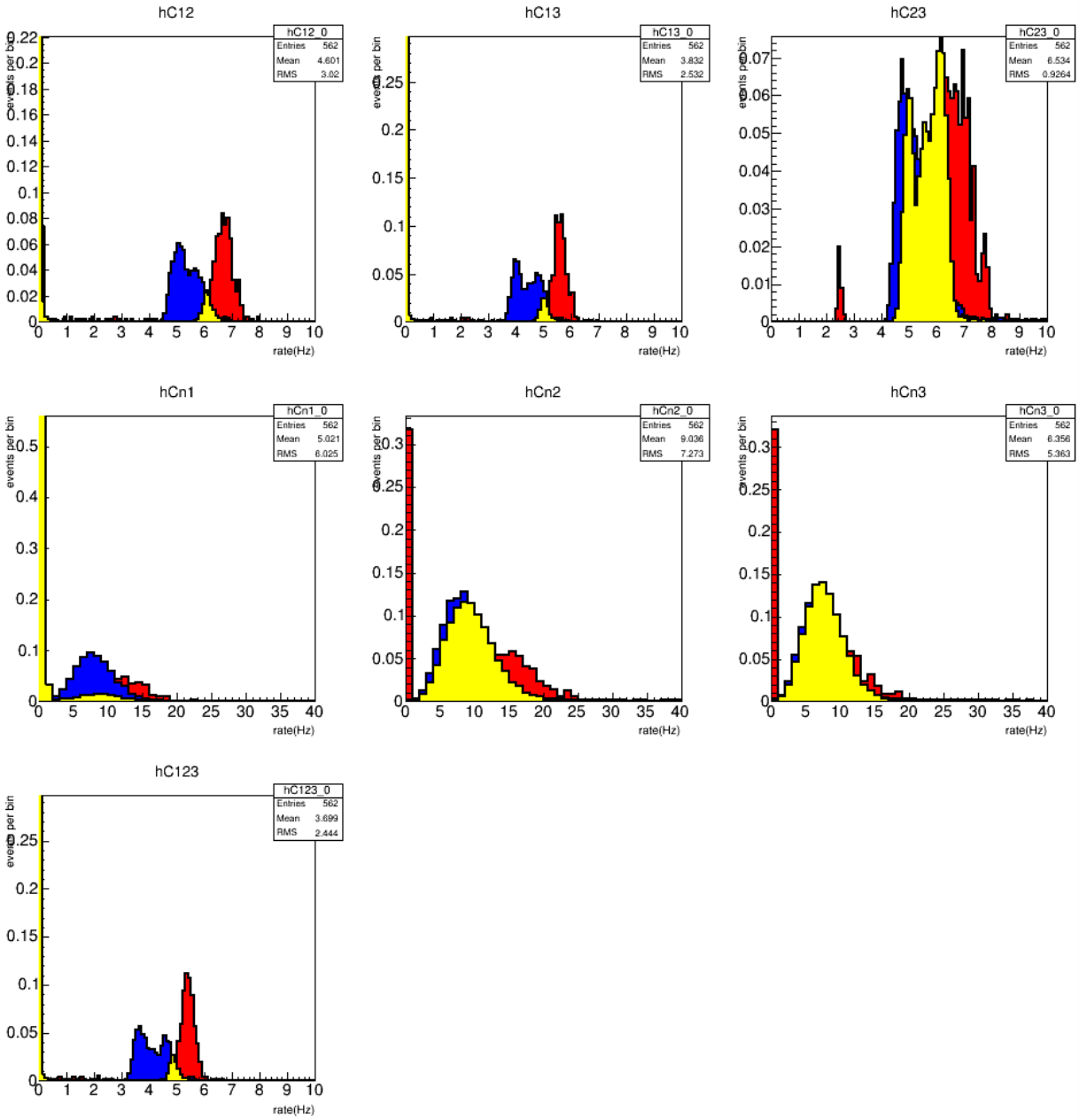


Figura 2.41: Histogramas referentes à taxa de contagem do conjunto C de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

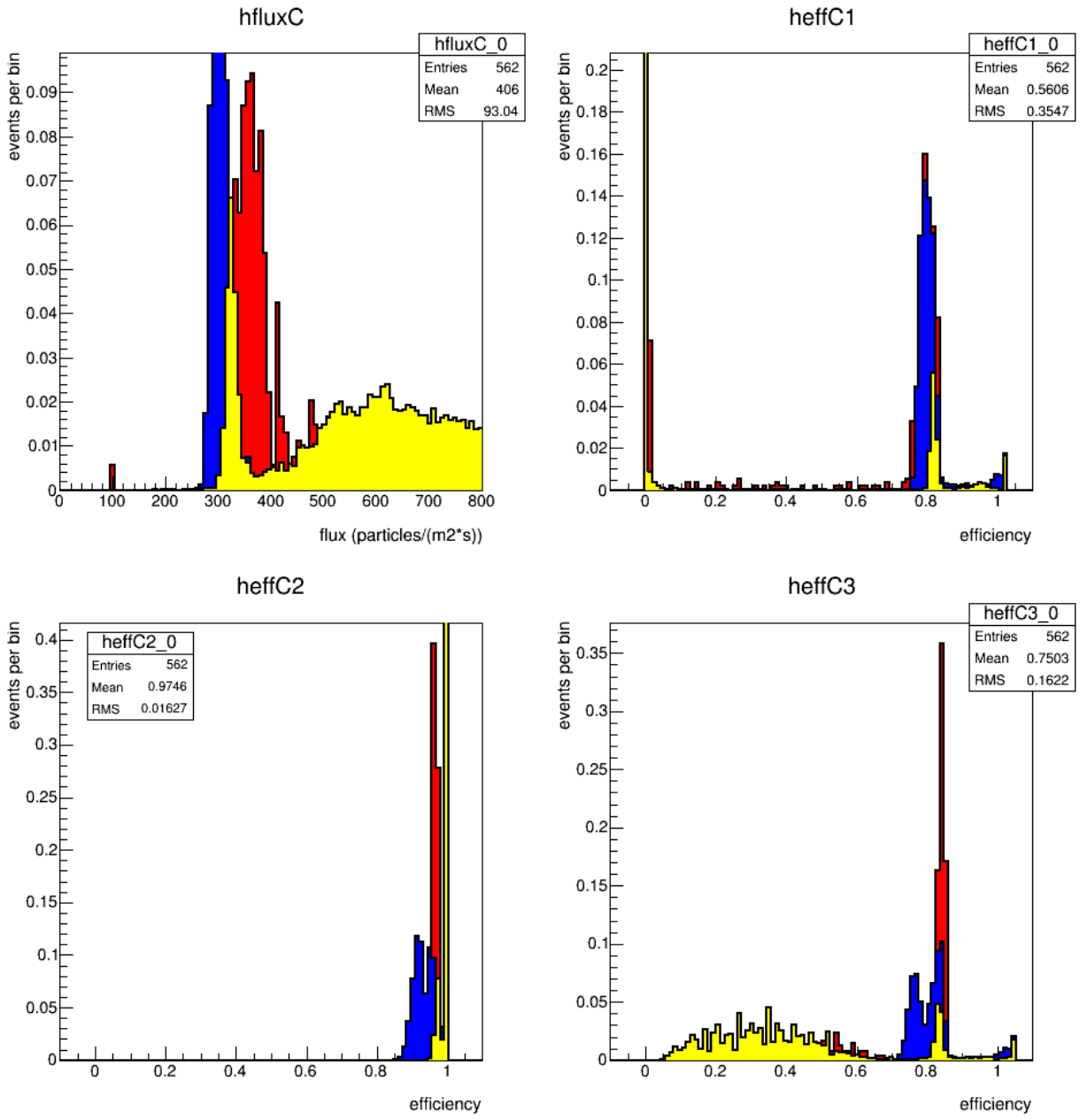


Figura 2.42: Histogramas referentes ao fluxo e eficiências do conjunto C de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

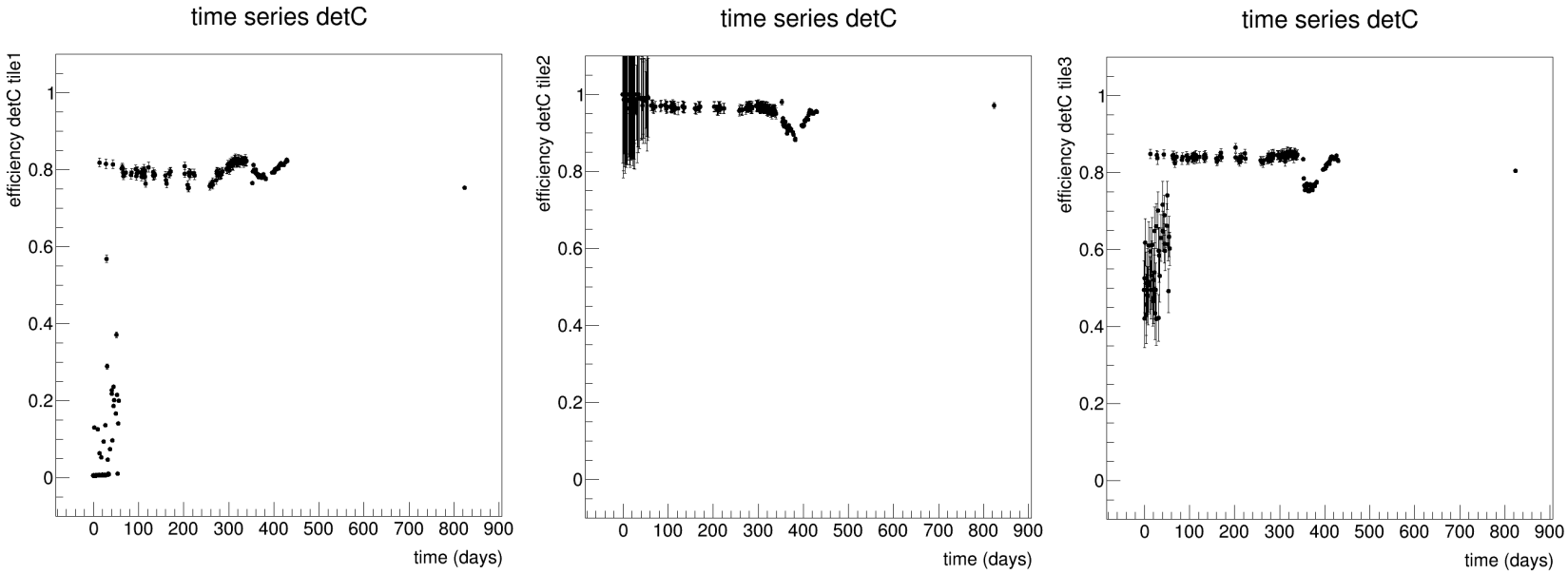


Figura 2.43: Série temporal da eficiência do conjunto C de detecção para as missões de 2015, 2016 e 2017.

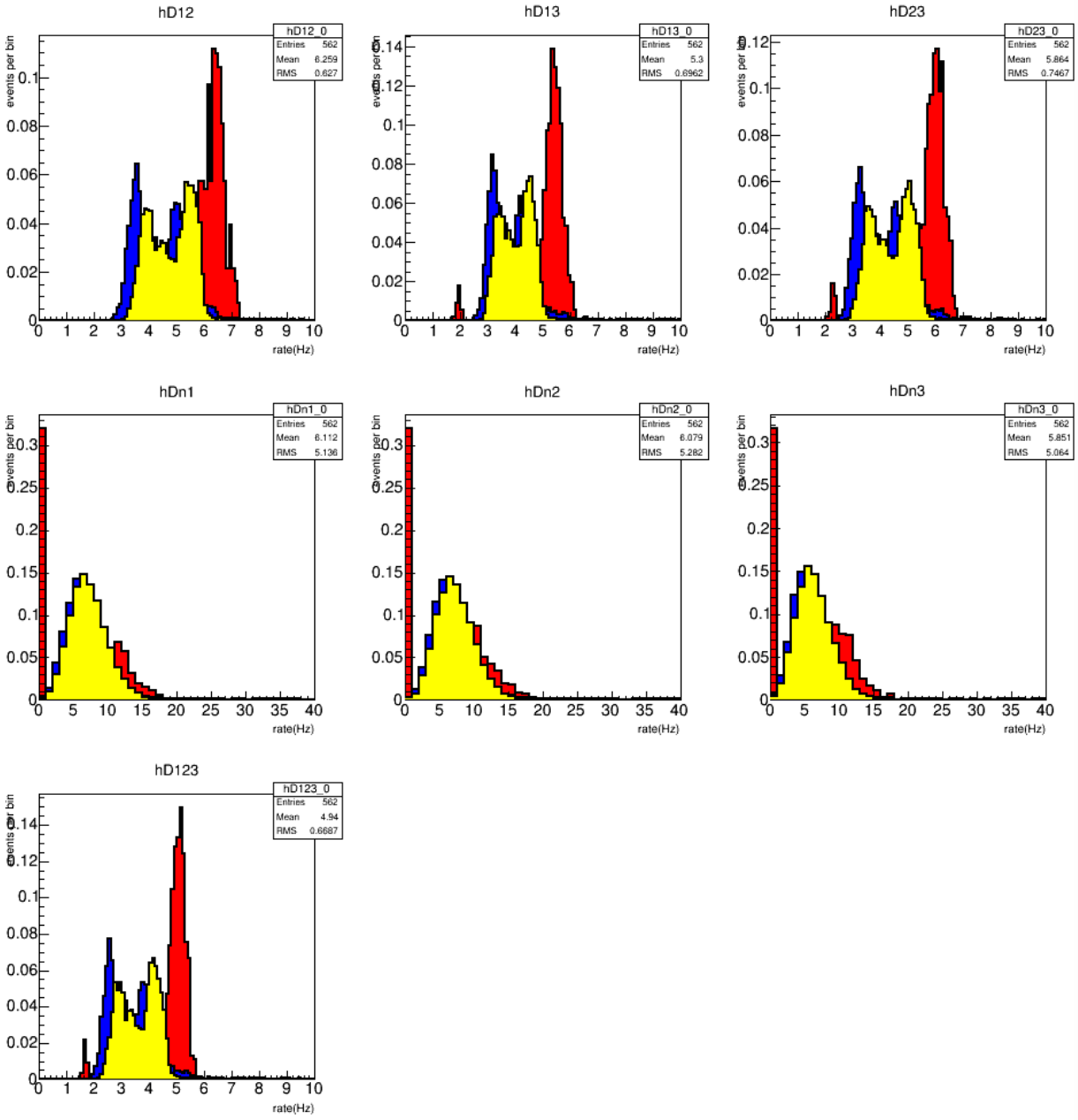


Figura 2.44: Histogramas referentes à taxa de contagem do conjunto D de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

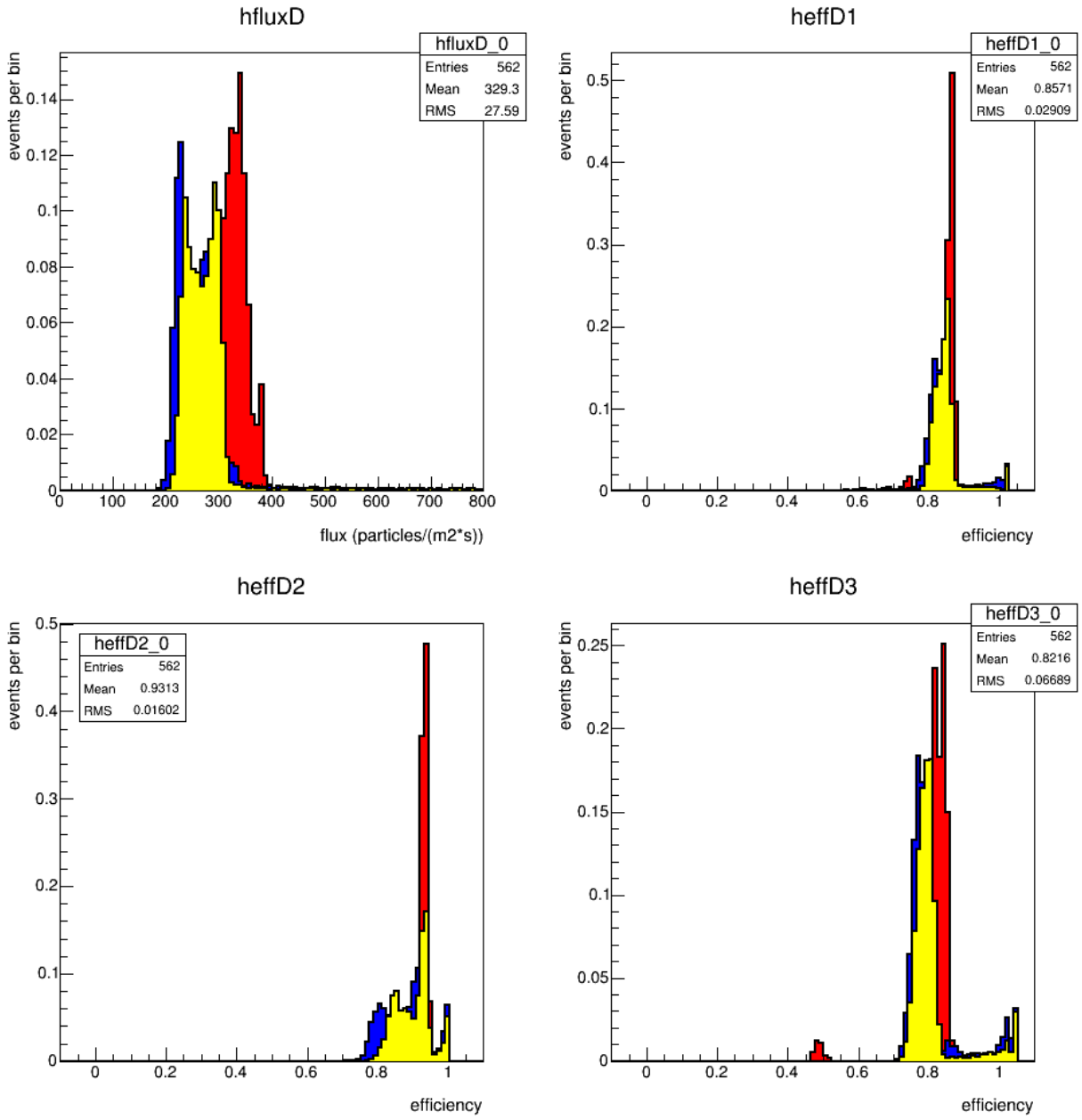


Figura 2.45: Histogramas referentes ao fluxo e eficiências do conjunto D de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

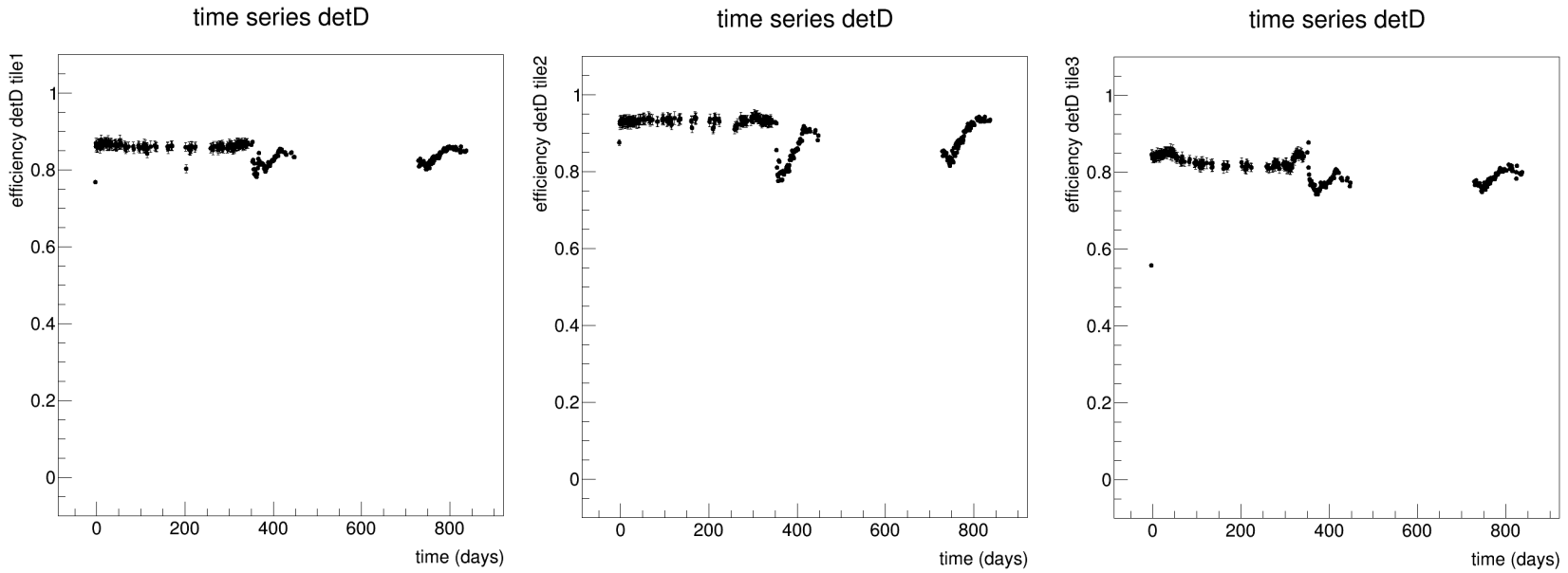


Figura 2.46: Série temporal da eficiência do conjunto D de detecção para as missões de 2015, 2016 e 2017.

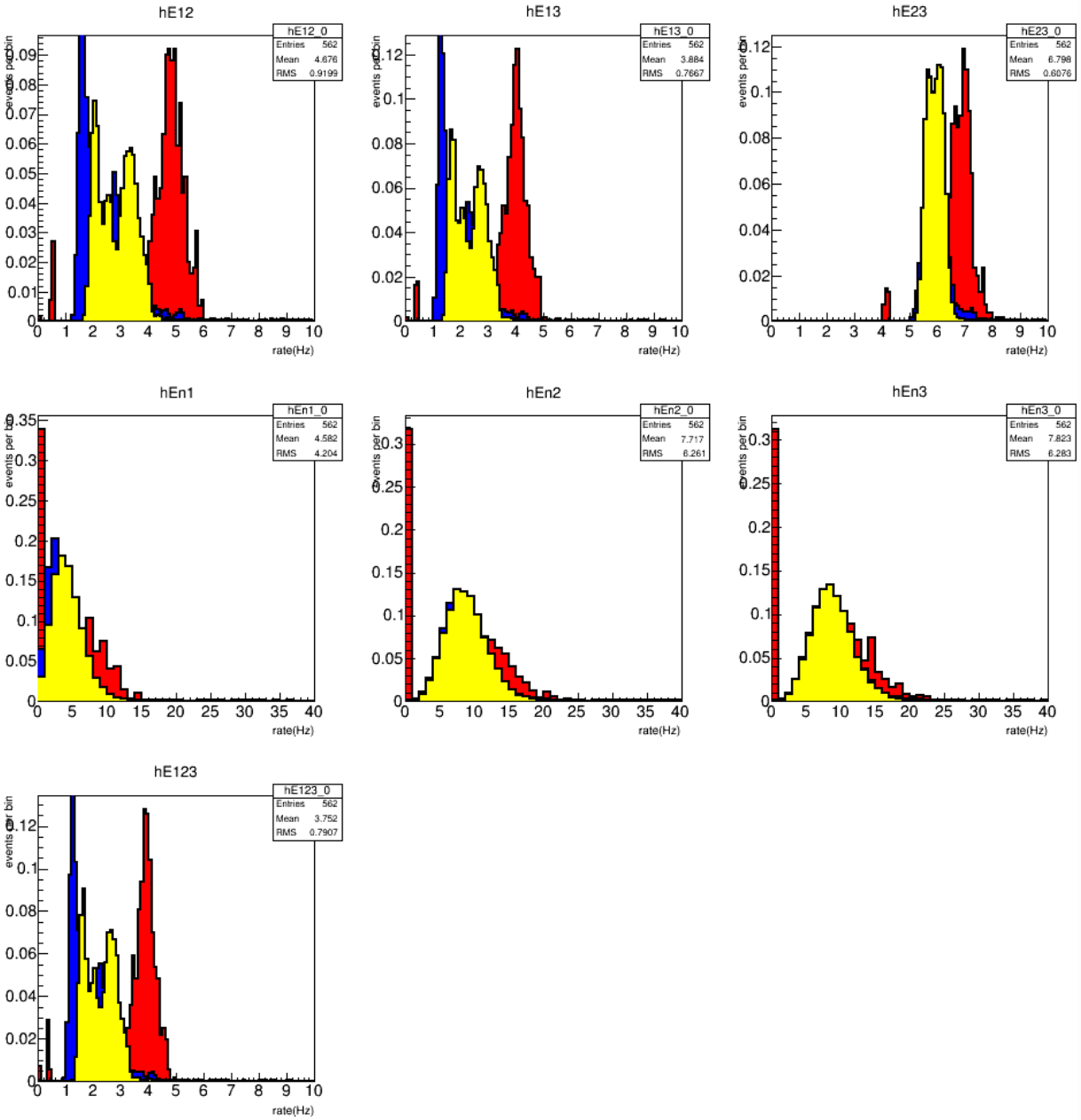


Figura 2.47: Histogramas referentes à taxa de contagem do conjunto E de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

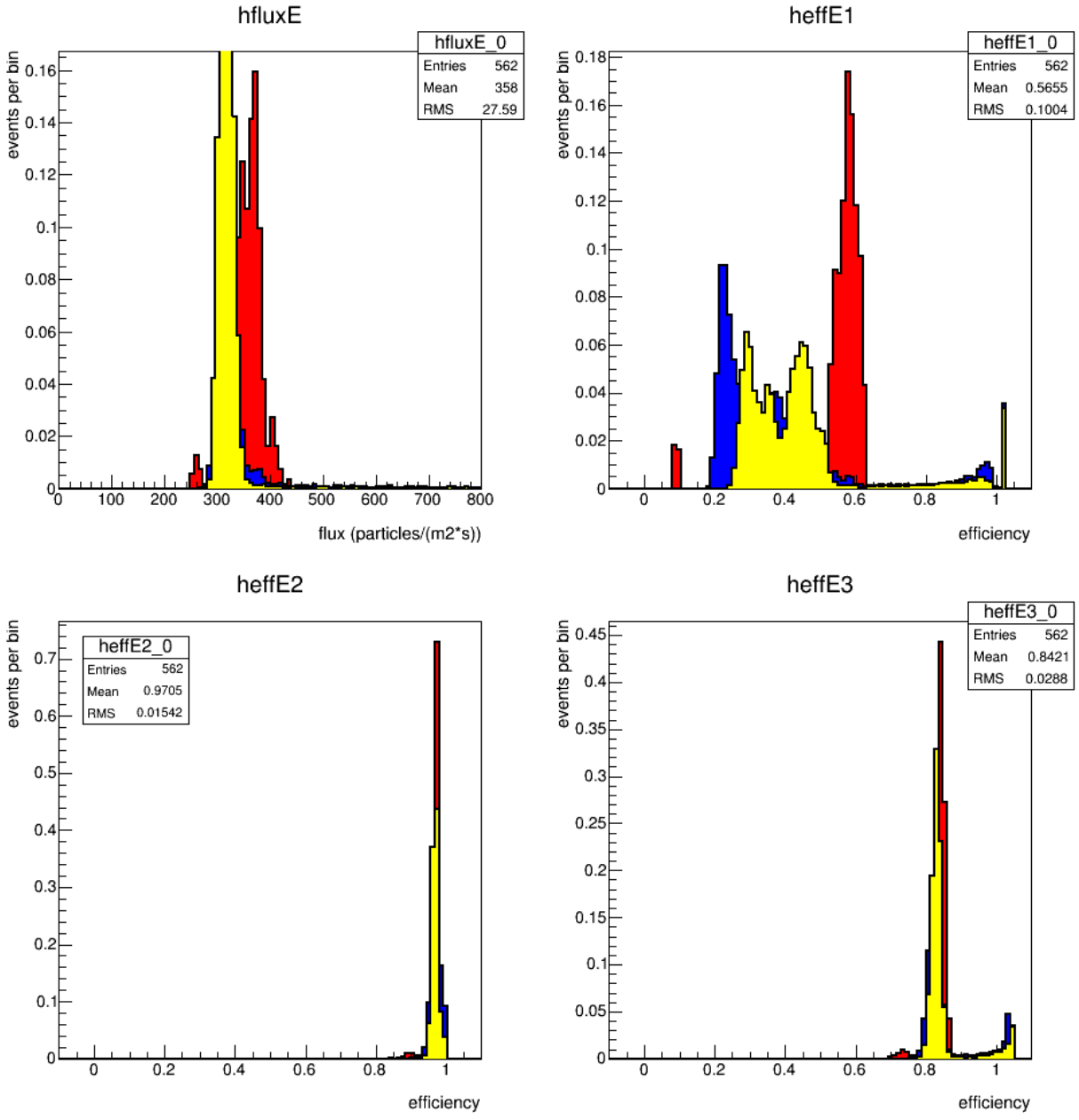


Figura 2.48: Histogramas referentes ao fluxo e eficiências do conjunto E de detecção, onde vermelho, azul e amarelo, são os dados referentes a 2015, 2016 e 2017, respectivamente.

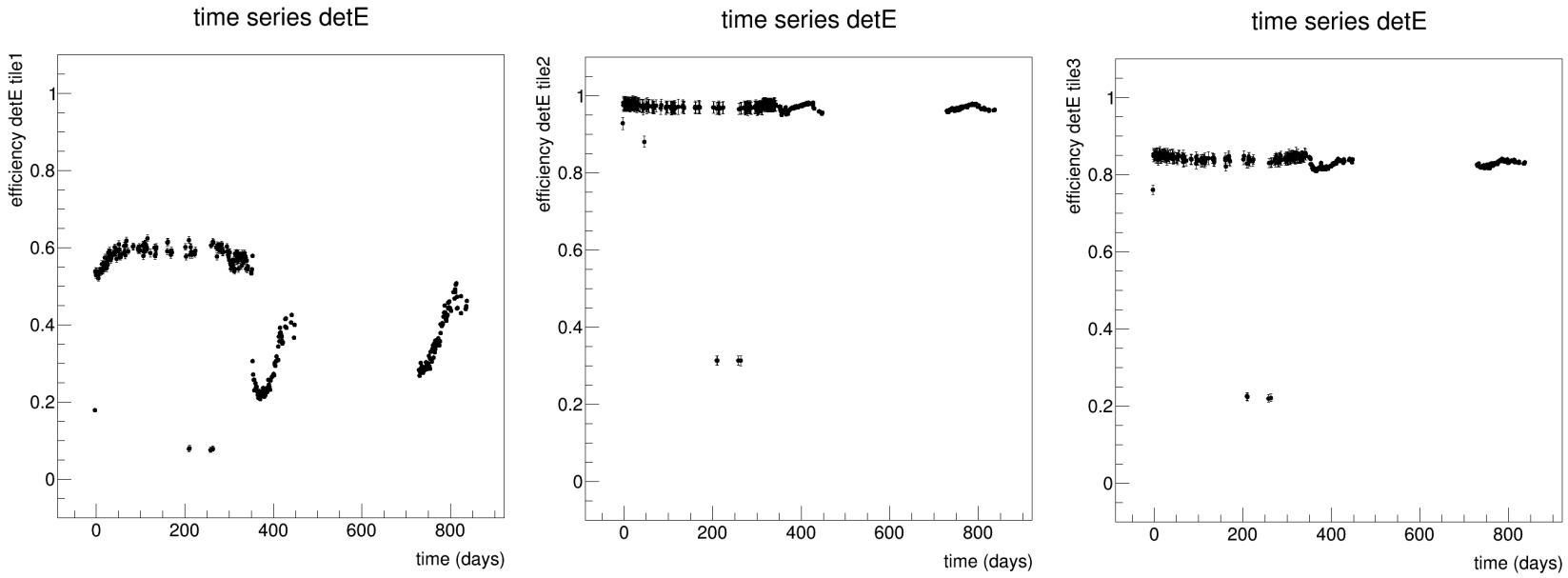


Figura 2.49: Série temporal da eficiência do conjunto E de detecção para as missões de 2015, 2016 e 2017.

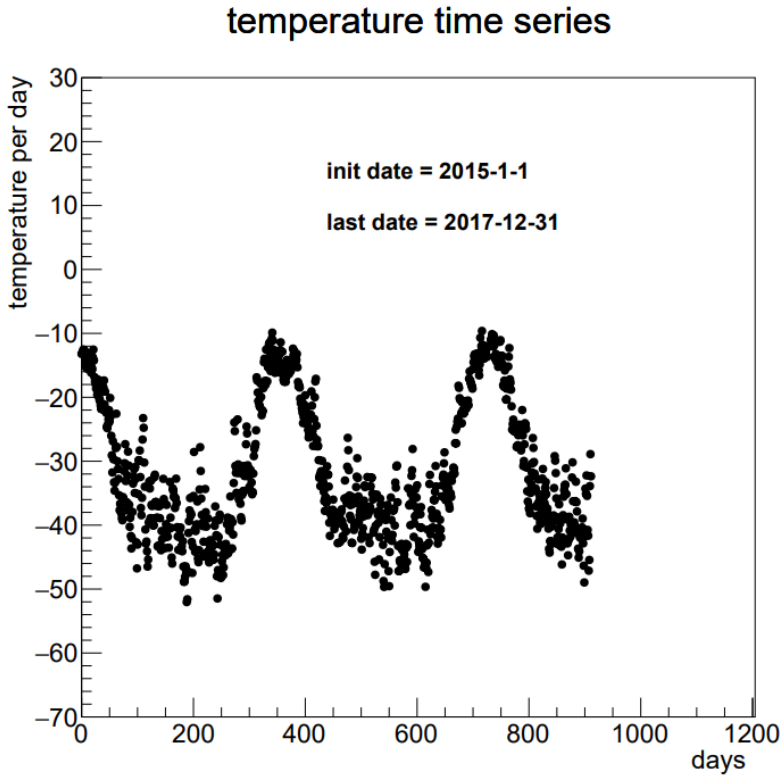


Figura 2.50: Gráfico de dias versus temperatura medido pelo sensor do Módulo Criosfera 1.

Ao analisarmos os dados de eficiência de cada detector, podemos observar uma variação da mesma em algumas épocas do ano, onde ocorrem grandes mudanças de temperatura. Tudo indica que esse fenômeno ocorra devido a dilatação do material, afetando, assim, a eficiência do detector. Quando a máscara de suporte das fibras dilata ou contrai, o posicionamento das mesmas varia, desalinhando-as com relação à MaPMT, consequentemente, diminuindo a eficiência óptica do detector. Esse fenômeno é agravado ainda mais no caso do detector do CREAT1, pois o *holder* de suporte foi fabricado com polietileno de alta densidade e sua mascara com latão, fazendo com que a diferença no coeficiente de expansão linear de cada material, influencie ainda mais no alinhamento do conjunto. Vale ressaltar a necessidade de um estudo mais aprofundado sobre essa relação entre a temperatura e a eficiência do detector. Entretanto, o cálculo do fluxo de RCG é corrigido com a eficiência, fazendo com que esta variação não afete significativamente os resultados. Tal conclusão foi baseada no fato em que os resultados de 2015, em que a eficiência se manteve constante, o fluxo correspondente se comportou similar aos demais anos.

Analisando os dados referentes aos detectores A e C, observamos que algumas de suas tiras tinham uma eficiência muito baixa, não permitindo que a correção de seu fluxo seja realizada corretamente, portanto, seus resultados foram descartados.

Na Figura 2.51, podemos observar o fluxo de raios cósmicos medido pelo CREAT1 na Antártica.

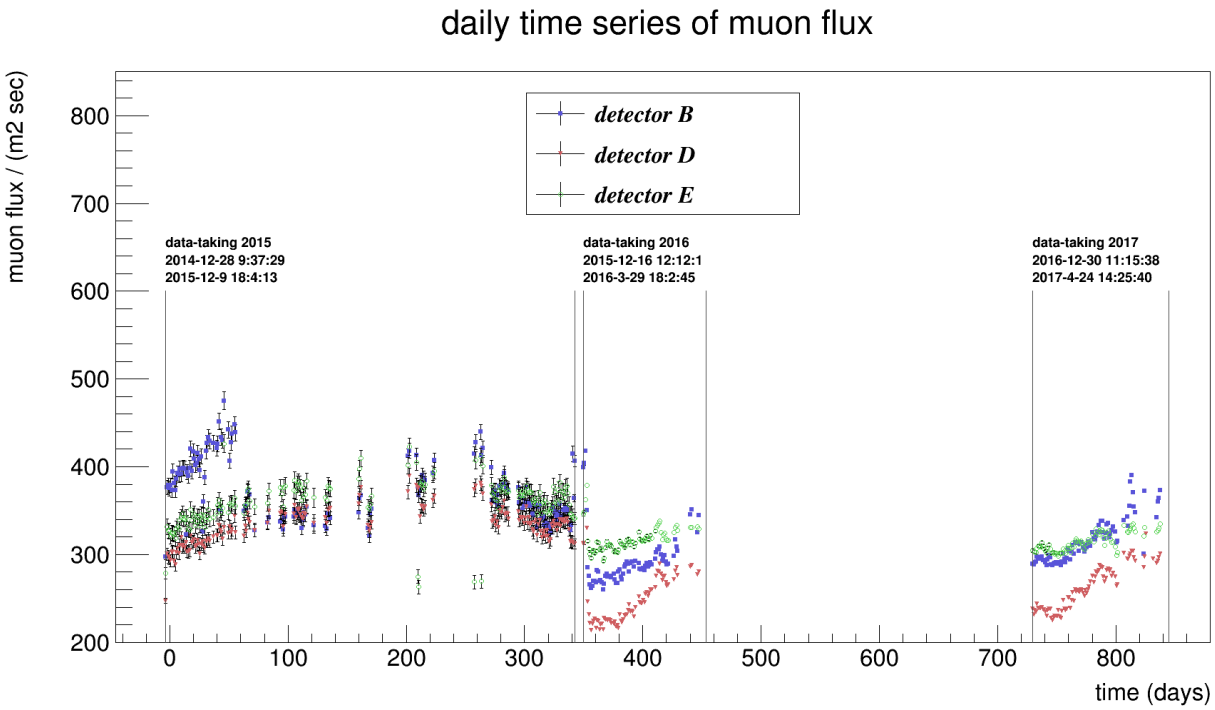


Figura 2.51: Gráfico de dias versus o fluxo de raios cósmicos na Antártica para o período de 2015, 2016 e 2017.

Nos resultados observamos que as incertezas nas medidas no ano de 2015 é maior do que nos demais anos, isso ocorre devido aos poucos dados coletados diariamente neste período. Entretanto, a diferença observada entre o fluxo de raios cósmicos médio em 2015 e nos demais anos, ocorre devido à diferença de alta tensão utilizada nas missões, de -750V em 2015 e -700V em 2016.

Por ser uma versão piloto, muitas análises com relação ao valor de *threshold*, alta tensão ideal, e influência da eficiência com a temperatura, não foram realizadas de forma adequada. Muitas dessas pendências foram devido à urgência na finalização e envio do detector para o continente antártico. Entretanto, embora suas medidas absolutas estejam comprometidas, como medidas relativas, seus resultados são válidos.

Um fenômeno de grande importância para a proposta do experimento são os chuveiros atmosféricos extensos, onde o fluxo de RCG é acima do normal. Existe a possibilidade desses fenômenos aumentarem significativamente a formação de nuvens, tornando sua correlação com os RCG mais evidente. Na Figura 2.52, podemos observar os dias do ano onde esse fenômeno supostamente ocorreu para comparação futura com os dados de formação de nuvens coletados por satélites meteorológicos.

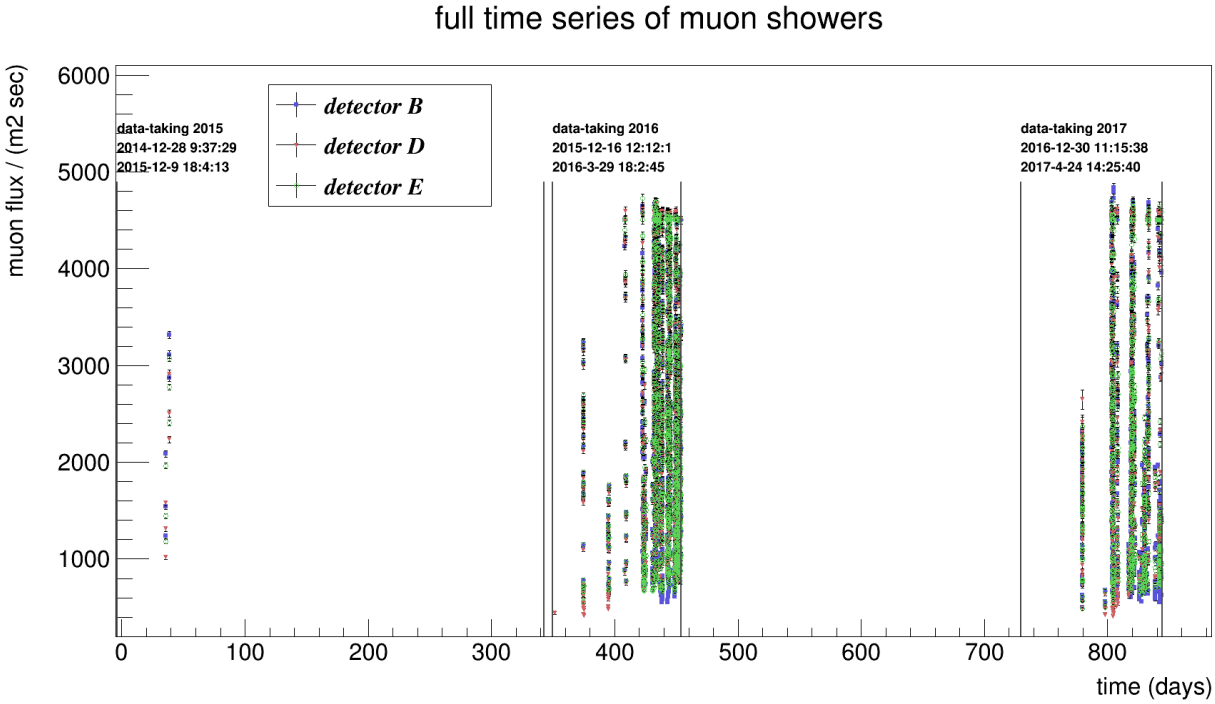


Figura 2.52: Gráfico de dias versus o fluxo de raios cósmicos na Antártica para o período de 2015, 2016 e 2017. Onde o fluxo de raios cósmicos é acima da média.

Através dos resultados, pôde-se perceber os pontos que deveriam ser melhorados, assim como os testes e análises necessários para validação do novo experimento, o CREAT2.

2.9 CREAT 2

Após bem sucedida missão realizada em 2014, foi iniciada a construção de um novo detector objetivando o *upgrade* do seu antecessor, porém utilizando a mesma tecnologia e *hardware*. A configuração atual é composta por dois conjuntos idênticos de sete tiras cintiladoras e fibras *WLS*, denominados detector A e detector B. Cada detector é organizado

da seguinte forma: no plano superior são colocadas duas tiras lado a lado de 60cm, no segundo plano (auxiliar) é colocada uma tira de 58cm alinhada e centralizada com os demais planos, no terceiro e quarto plano são colocadas duas tiras de 60cm organizadas como no plano superior, em uma configuração chamada de 2x1x2x2. A vantagem desta nova configuração está em sua maior área de incidência de partículas, três planos coincidentes para o cálculo do fluxo e um plano auxiliar para o cálculo mais preciso de eficiência individual de cada plano. A Figura 2.53 apresenta uma imagem em três dimensões do sistema, parte do conjunto de sete tiras do detector A está a mostra para facilitar sua visualização.

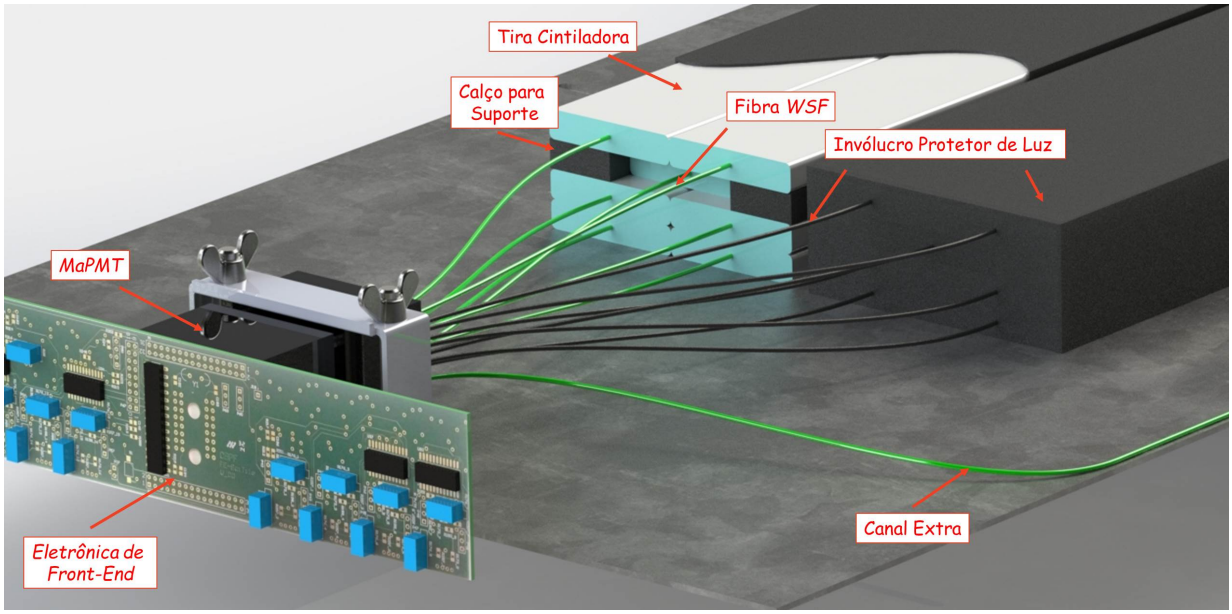


Figura 2.53: Desenho em três dimensões do interior do detector.

O sistema é organizado de forma que a probabilidade de uma partícula passar pelo plano auxiliar sem passar por um dos demais planos seja praticamente nula, pois, devido sua geometria, seria necessários ângulos de θ maiores que 45°. Dessa forma, o plano auxiliar é utilizado para calcular a eficiência dos demais planos, corrigindo, assim, o cálculo do fluxo de raios cósmicos.

De acordo com a simulação Monte Carlo, o fator de correção geométrica da eficiência de cada plano de detecção é de 0,999 para o primeiro plano, 1 para o terceiro plano, e 0,978 para o quarto plano. Através da simulação, foi calculado que 95,1% dos raios cósmicos que passam, conseguem ser detectados.

2.9.1 *Firmware* de Aquisição do *FPGA*

Para o tratamento do sinal digital vindo da *Front-End*, triagem e contagem dos eventos favoráveis, um módulo CAEN V1495 foi utilizado. O módulo possui um *FPGA ALTERA Cyclone* EP1C20F400C6, e segue os padrões VME (ANSI/IEEE 1014-1987).

O *firmware* desenvolvido possui quatro blocos principais e o seu diagrama é apresentado na Figura 2.54.

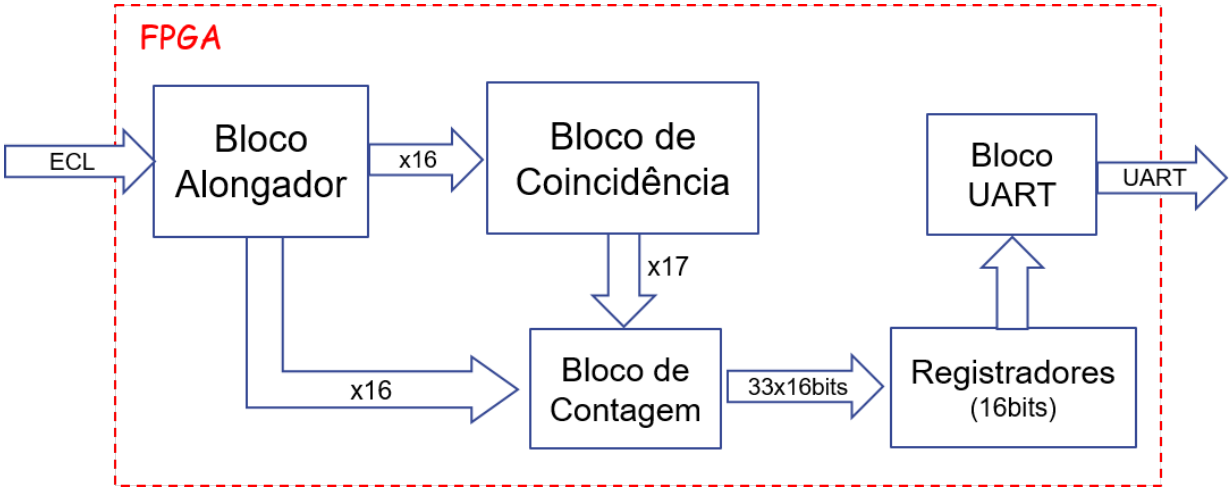


Figura 2.54: Diagrama de blocos do *firmware* do CREAT2.

Os detalhes de cada bloco seguem nas seções seguintes e seu código em VHDL está apresentado no Apêndice H.

Bloco de Alongamento de sinal digital

Os pulsos diferenciais *ECL* gerados pela eletrônica de *Front-End* são enviados para o *FPGA* do módulo V1495. Devido à pequena largura de pulso desses sinais, podendo chegar a 20ns, e a possíveis atrasos entre os canais, foi desenvolvido um bloco no *firmware* responsável pelo alongamento desses sinais, de forma que tais pulsos de entrada com largura variável possam ser alongados para larguras fixas, configuráveis e múltiplas do *clock* do sistema, podendo ir de $25ns$ até $1,6384ms$. Como o método de detecção é assíncrono, sinais com largura de pulso menores que o período do *clock* do sistema ($25ns$) podem ser detectados.

Tendo o funcionamento similar à um multivibrador monoestável, qualquer pulso

de entrada desse bloco, independente de sua largura, gera um pulso de saída mantido em nível alto até que um contador de pulsos de *clock* chegue à um valor predeterminado no seu registro correspondente. Os pulsos de entrada subsequentes que vierem antes que o sinal de saída volte ao seu nível baixo, são ignorados, como mostrado na Figura 2.55.

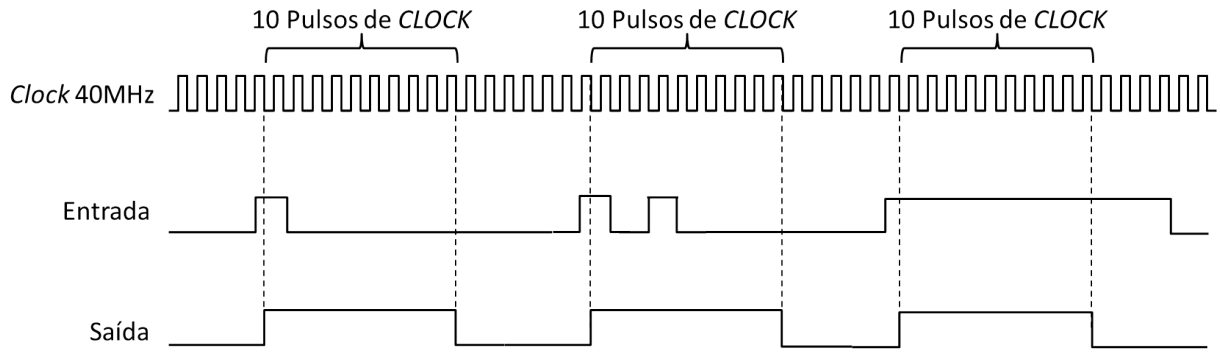


Figura 2.55: Sinais de entrada e saída do bloco de alongamento de sinal digital.

Bloco de Coincidência

Uma das dificuldades na montagem de um detector de raios cósmicos está na blindagem do sistema à ruídos, luz externa e corrente de escuro presente na maioria das fotomultiplicadoras. Para minimizar ou até mesmo eliminar a ocorrência desses agentes, uma lógica de coincidência é realizada, identificando apenas os sinais coerentes e, consequentemente, eventos oriundos da passagem de raios cósmicos.

O bloco de coincidência possui 17 sinais de saída, sendo suas entradas ligadas aos pulsos digitais, já alongados, dos 14 canais escolhidos. A lógica combinacional do primeiro conjunto está apresentada na Figura 2.56, sendo a mesma para o segundo conjunto.

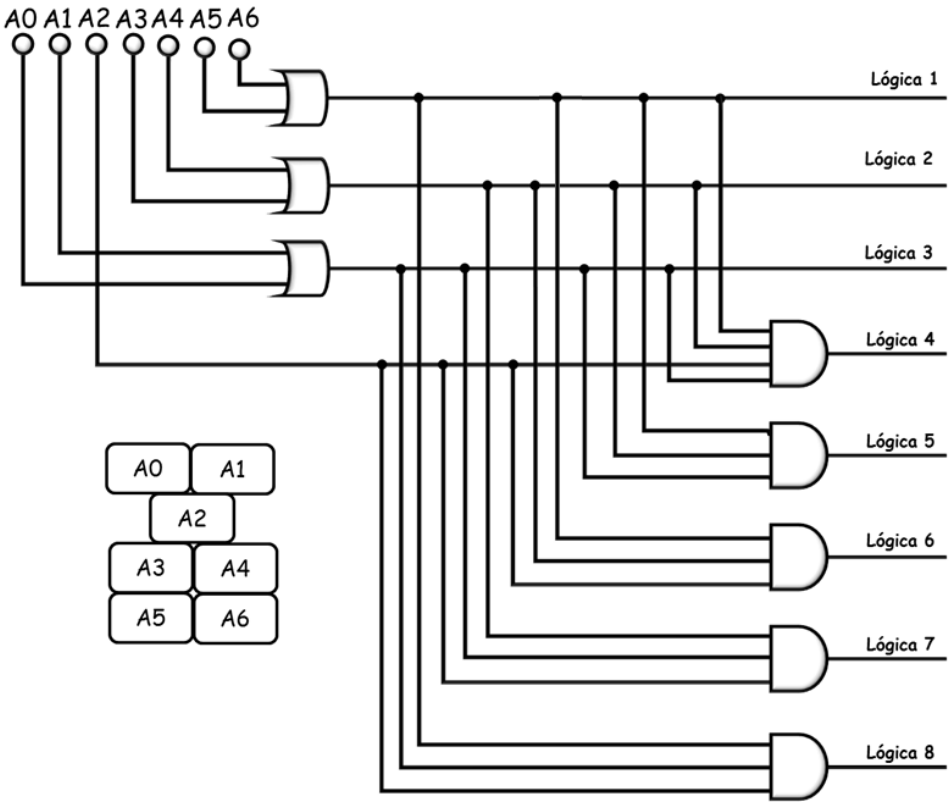


Figura 2.56: Lógica de coincidência de cada canal.

Além dos oito sinais lógicos gerados pelos dois conjuntos, é gerado também um sinal de coincidência *AND* entre todos os planos de detecção dos dois conjuntos. Os sinais lógicos da saída do bloco de coincidência entram em um bloco de contagem, onde seus respectivos registros são incrementados.

Embora a lógica de coincidência seja capaz de ignorar ruídos incoerentes, um grande problema na montagem desse tipo de detector é o seu *crosstalk* óptico. O *crosstalk* óptico, em sua maioria, ocorre nas células adjacentes da *MaPMT*. Por ser coerente, passa pela lógica de coincidência sem ser detectado, sendo confundido com eventos válidos. Uma solução para esse problema foi o embaralhamento dos canais, de forma que se evite ao máximo a lógica *AND* entre canais adjacentes da *MaPMT*. O mapa da Figura 2.57 foi utilizado para minimizar o problema de *crosstalk* óptico.

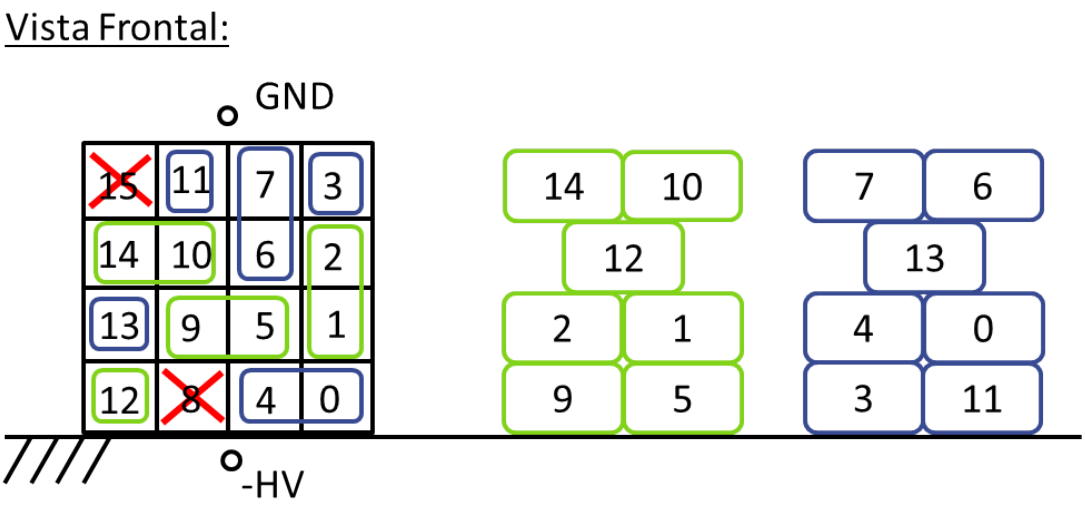


Figura 2.57: Mapa utilizado pelo detector do CREAT2.

Devido à necessidade de apenas 14 dos 16 canais, e por serem os menos eficientes, os canais 8 e 15 foram escolhidos para não participar da configuração. Entretanto, são utilizados como parâmetro indicativo da presença de luz externa na caixa.

Bloco de Contagem

O bloco de contagem tem a função de incrementar os 33 registros de contagem (16 de contagem individual de cada canal e 17 de coincidências) durante o tempo definido pelo usuário. Esse tempo representa o valor do registro de tempo de aquisição multiplicado por um segundo. Ao final, os registros são enviados para o bloco UART de envio de dados e então seus registros de contagem são zerados para realização de uma nova contagem. Na Figura 2.58 podemos observar o diagrama de blocos de sua lógica para um registro de contagem, o mesmo é feito para os demais.

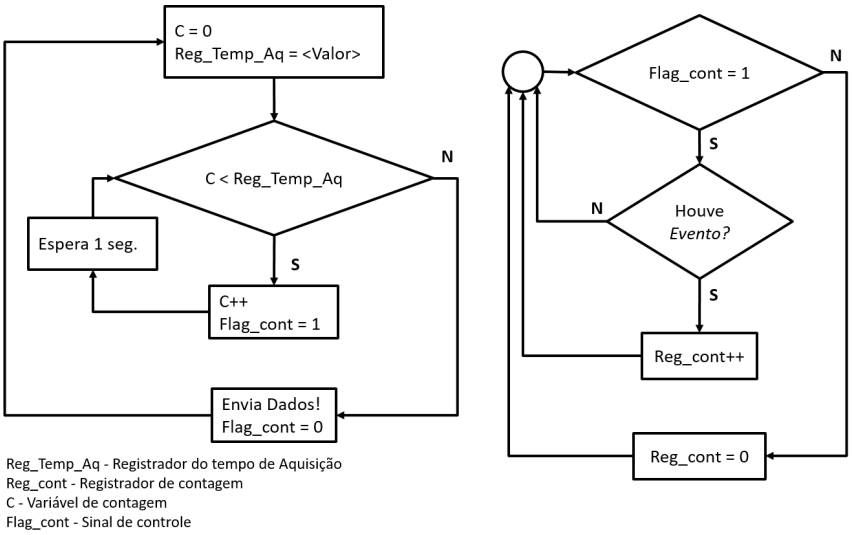


Figura 2.58: Diagrama de blocos do contador.

Bloco *UART* de Envio de Dados

O bloco inicialmente converte os valores de contagem de cada registrador para caracteres da tabela *ASCII* na base hexadecimal, formando assim uma palavra de 4 *bytes* que vai de 0000 a FFFF. Feito isso, um bloco UART coleta cada byte dessa palavra e o envia serialmente para uma saída lemo do módulo V1495. A taxa de transmissão dos dados (*baund rate*) é fixada em 9600 *bps*, sendo esta também configurada no equipamento que recebe os dados.

2.9.2 Software de Controle e Aquisição de Dados pelo *PC*

Para coleta, armazenamento e controle, um software chamado de *SaveDataAq.c* foi desenvolvido em *C/C++*. Tal software efetua comunicação com a eletrônica de *front-end*, através da porta serial, e com o módulo VME v1495, através do módulo VME v1718 e do barramento VME. O software é executado juntamente com os argumentos referentes aos parâmetros de controle que serão alterados, ou seja, "./SaveDataAq.c $< arg > < valor >$". A Tabela 2.2 apresenta todos os argumentos e descrição dos mesmos.

Tabela 2.2: Tabela dos argumentos de controle.

Argumento	Valor	Descrição
-F	"$< arquivo >$"	Local e nome do arquivo de armazenamento.
-A	Segundos (Dec)	Altera o registrador do FPGA correspondente ao tempo de aquisição em segundos.
-G	Pulsos de *clock* (Dec)	Altera o registrador da FPGA correspondente a janela de tempo do bloco alongador de pulso.
-n	(Dec)	Configura o número de eventos que serão salvos.
-t	Posição (Dec)	Configura o valor de *threshold* comum para todos os canais.
-v	Valor de alta tensão	Insere no arquivo gerado o valor de alta tensão do sistema.
-o	0000 a FFFF (Hex)	Altera o registrador do FPGA correspondente a saída lemo Out_0.
-O	0000 a FFFF (Hex)	Altera o registrador do FPGA correspondente a saída lemo Out_1.
-U	(Dec)	Informa a porta serial em que a *front-end* está conectada.

2.9.3 Armazenamento e Envio de Dados Via *TCP/IP*

O módulo de armazenamento e envio de dados é composto por um microcontrolador *Atmega 2560*, um *shield ethernet + SD card* e um módulo RTC (*Real Time Clock*). Para obtermos informações adicionais das condições externas do detector e verificar possíveis

correlações com as medidas, foram acrescentados um módulo BMP180, que fornece valores de pressão e temperatura, e um módulo MAG3110, que fornece as medidas de campo magnético nos eixos xyz.

O Arduino Mega utilizado é composto principalmente por um microcontrolador ATmega2560, com 54 pinos digitais de entrada e saída, sendo 14 deles com saída PWM (*Pulse Width Modulation*), 16 pinos de entrada analógicos, 4 portas seriais UARTs, um clock de *16MHz* e um conversor USB-Serial.

O shield ethernet permite que o microcontrolador se conecte à internet através de um cabo *RJ45*, possibilitando o envio dos dados via satélite pelo sistema *IRIDIUM Pilot* através do protocolo TCP/IP. Inserido no *shield*, temos um módulo *SD card*, que permite o armazenamento dos dados num cartão *SD*. A Figura 2.59, apresenta o diagrama de blocos do sistema.

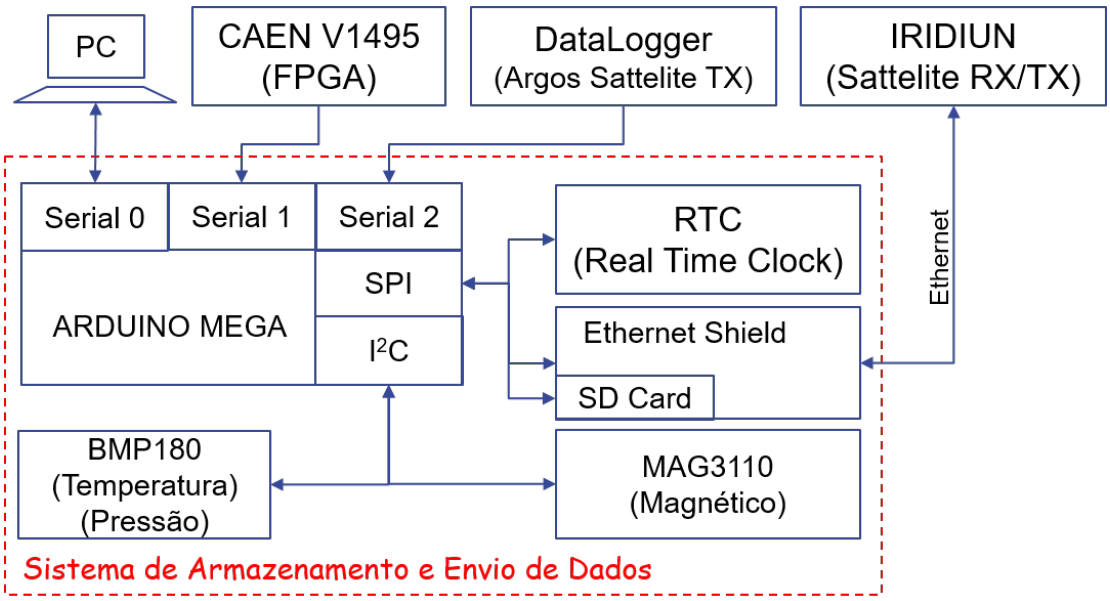


Figura 2.59: Diagrama de blocos do sistema de armazenamento e envio de dados.

O *firmware* do microcontrolador foi desenvolvido em linguagem C utilizando a IDE do Arduino. Nele, os dados que vem do módulo *VME V1495*, via protocolo *UART* (*Universal Asyncronous Receiver/Transmitter*), são armazenados no cartão *SD*. Os dados salvos possuem o seguinte formato:

- Data no formato: "*dia-mês-ano_hora:minuto:segundo*";
- UnId - Endereço único do evento em hexadecimal;

- As informações de temperatura, pressão e campo magnético no momento que o evento ocorreu;
- Valor dos 33 registros de contagem (16 de contagem individual de cada canal e 17 de coincidências), em hexadecimal, para um tempo de aquisição pré-configurado.

São utilizadas três das quatro portas seriais do microcontrolador: a UART0 é responsável pela comunicação com outros periféricos, possibilitando, assim, *debug* e controle do sistema de aquisição; a UART1 é dedicada ao recebimento dos dados enviados pelo FPGA; a UART2 é responsável pelo recebimento da data e hora para a atualização do RTC, pois seu atraso pode chegar em 1 minuto por mês.

O *firmware* desenvolvido pode ser observado no Apêndice G.

2.9.4 Ponto Ideal de *Threshold*

O ponto ideal de *threshold* do detector foi extraído variando o *threshold* do discriminador e observando a contagem da coincidência tripla. No teste, foi utilizada uma alta tensão de -750V e tempo de aquisição de 30 segundos. Os resultados estão presentes na Figura 2.60.

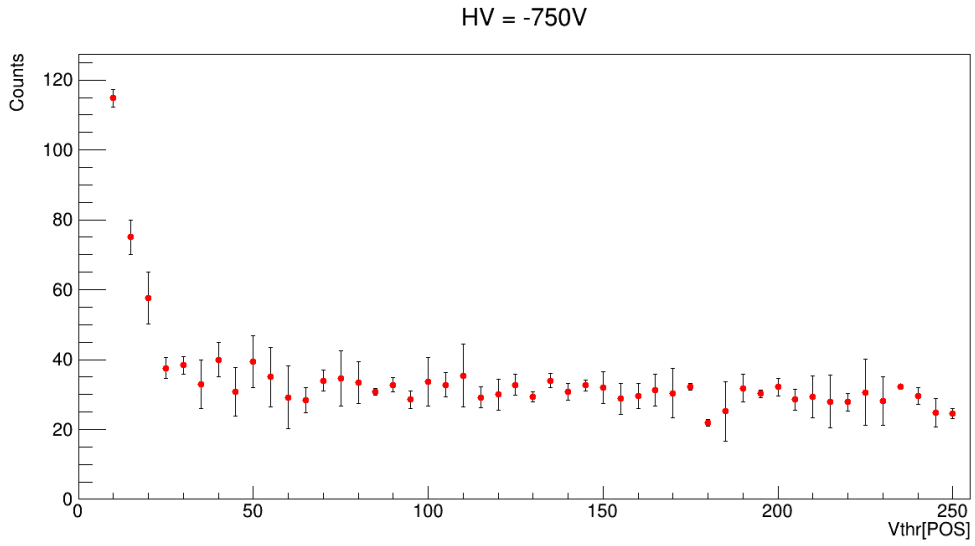


Figura 2.60: Gráficos de eficiência e fluxo de raios cósmicos para diferentes valores de alta tensão.

O gráfico indica uma contagem muito acima do esperado para valores de *threshold*

abaixo de 30. Isso acontece devido a alta taxa de ruído eletrônico e corrente de escuro. Portanto, através desses resultados foi estimado que a tensão ideal de *threshold* deveria ser de 30, ou seja, aproximadamente -50mV, compatível com os valores medidos na caracterização da eletrônica de *front-end*.

2.9.5 Validação do *Threshold* em Relação à Carga Detectada.

Para estimar a carga detectada, na incidência dos raios cósmicos, em função do valor de *threshold* escolhido, foi realizado um teste utilizando o módulo CAEN V792 QDC (*Charge to Amplitude Conversion*). O *setup* experimental está representado pelo diagrama de blocos da Figura 2.61

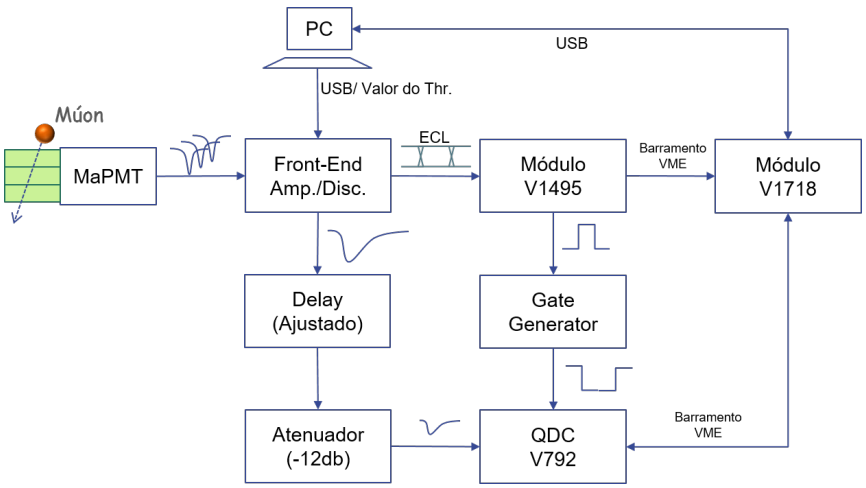


Figura 2.61: Diagrama de blocos do *setup* experimental para análise de carga do múon.

No *setup*, são dispostos três cintiladores, um em cima do outro. Os três pulsos de corrente gerados na passagem do múon são discriminados e a coincidência deles gera o *trigger* para leitura do sinal analógico de um deles pelo QDC. Como a amplitude do sinal é maior do que o pulso de entrada permitido pelo QDC, o sinal é atenuado em $-12db$. O *delay* e o módulo do *gate generator* foram configurados de forma que a janela temporal (*GATE*) de integração do sinal tenha a mesma largura que o pulso analógico. A análise dos dados coletados pelo QDC está apresentada na Figura 2.62.

O módulo CAEN V792 QDC possui um pedestal devido à corrente gerada ao aplicar um sinal de *GATE*. Esse pedestal não depende linearmente da largura do *GATE* aplicado, porém, seu valor deve ser subtraído para se obter apenas a carga do múon. Na análise, foi

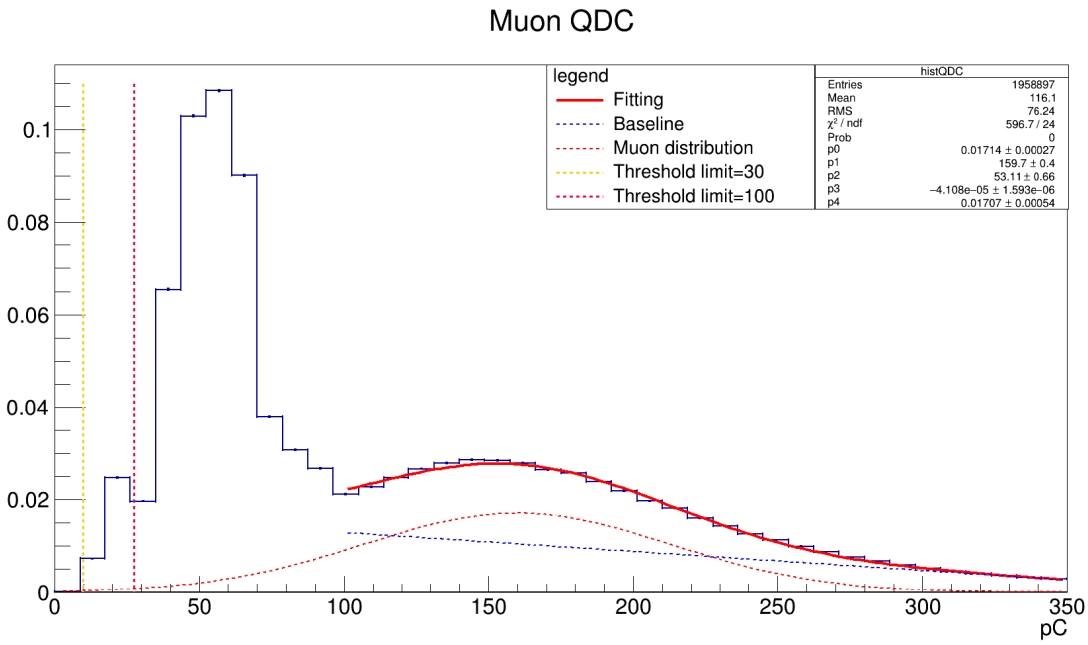


Figura 2.62: Histograma normalizado do QDC do múon.

realizado o ajuste (linha contínua vermelha) através da gaussiana somado a um polinômio de grau 1, representando um *baseline*. Depois disso, o *baseline* (linha tracejada azul) foi subtraído, restando apenas a carga do múon (linha tracejada vermelha).

Para estimar a carga de um sinal com amplitude máxima correspondente aos *thresholds* de 30 e 100, foi extraída, através do osciloscópio, a área aproximada Vdt desses sinais. A carga é dada pela Equação 2.11:

$$dQ = \frac{Vdt}{R} \tag{2.11}$$

Onde Q é a carga elétrica, V é a tensão, t é o tempo e R é a impedância de 50Ω em que a carga está sendo aplicada. Obtemos uma carga de aproximadamente $36pC$ e $144pC$ para *threshold* de 30 e 100, respectivamente.

Como o sinal da saída do amplificador passa pelo atenuador de $-12db$, atenuamos as cargas referentes aos dois *thresholds* de acordo com a Equação 2.12:

$$-12db = 20LOG(\frac{Saida}{Entrada}) \tag{2.12}$$

As linhas tracejadas laranja e roxa, apontam onde passam os pontos de corte dos

respectivos valores de *threshold* 30 e 100. Podemos observar que a quantidade de múons perdidos devido aos *thresholds* escolhidos é próxima de 1%.

2.9.6 Sistema de Injeção por Leds

Com o objetivo de testar a eficiência óptica e o funcionamento dos detectores, foi proposto o desenvolvimento de um sistema de injeção de luz, projetado pelo técnico Marcos Vinicius Koebcke, com auxílio dos demais integrantes do projeto.

O sistema foi concebido para ser utilizado em diversos tipos de detectores a base de fotomultiplicadoras, podendo ser utilizado tanto no CREAT2, como no CREAT3. Projetado para ser utilizado com até 16 leds, cada led é anexado à uma fibra *wavelength shifter*, e um circuito analógio e digital controla sua luminosidade e frequência de pulsação.

Para fins de análise, a frequência de pulsação dos LEDs precisa ser bem superior ao fluxo de raios cósmicos, porém sua contagem não poderia exceder o tamanho dos registradores de contagem dos detectores do CREAT 1 e 2, de 16 *bits*. Portanto, foi utilizada a frequência de $490Hz$, do PWM (*pulse-width modulation*) do microcontrolador utilizado. Nele, é possível selecionar qual dos 64 leds serão habilitados para pulsar, permitindo sua contagem pelo software de aquisição, ou emulação de trajetos de partículas cósmicas. Com ele, é possível identificar *crosstalk* óptico entre canais, definir a eficiência do conjunto fibra WLS + MAPMT, simular eventos de raios cósmicos e testar o funcionamento dos detectores. A Figura 2.63 apresenta uma imagem do sistema montado no detector do CREAT2.

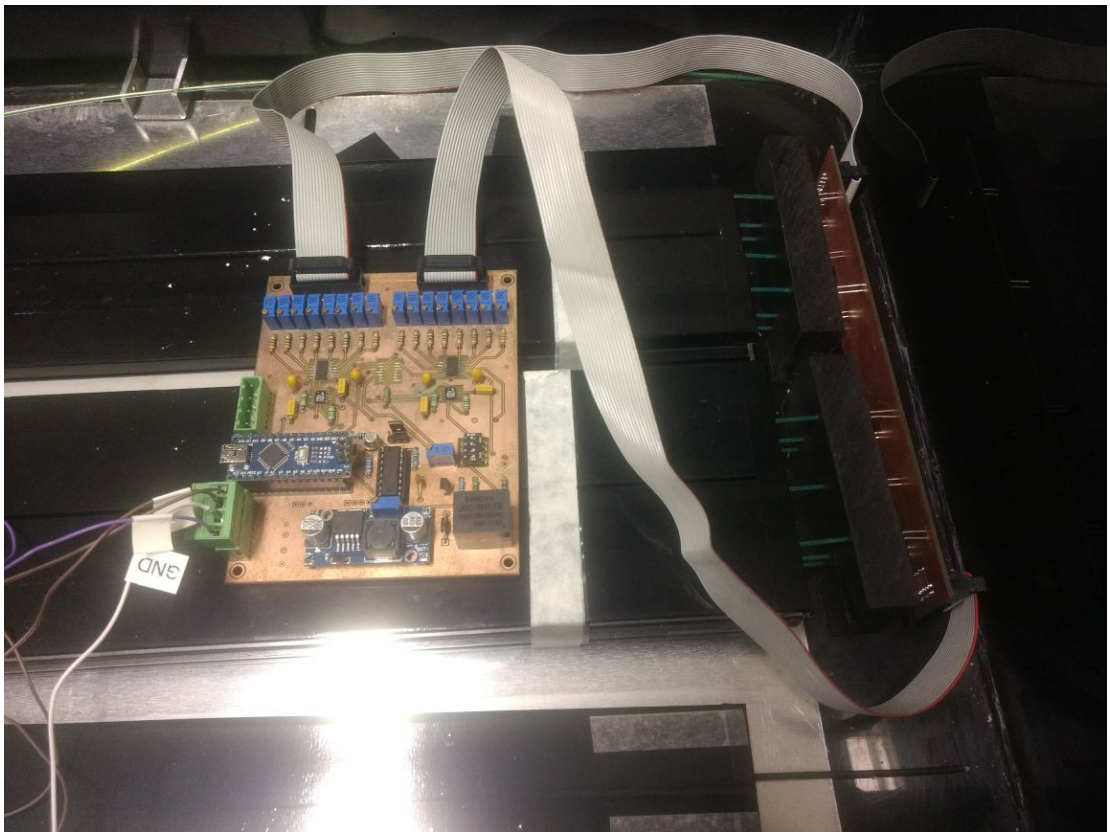


Figura 2.63: Foto do sistema de injeção de luz ligado ao detector do CREAT2.

2.9.7 Calibração do Sistema de Injeção de Luz

Para emularmos um sinal de pulso luminoso similar ao gerado na passagem de uma partícula múon pelo cintilador, é necessário que haja uma calibração do sistema de injeção de luz. Ela é feita controlando a corrente elétrica que passa pelos LEDs UV, assim como sua largura de pulso. Espera-se que a carga elétrica depositada pelos LEDs seja similar a de uma partícula cósmica. Logo, utilizando o mesmo *setup* mostrado anteriormente, refizemos os testes com o sistema de injeção de luz ligado. O *setup* experimental está representado pelo diagrama de blocos da Figura 2.64

Nesse *setup*, foi utilizado como *trigger* a contagem individual do canal e não a coincidência, como no QDC do múon. Embora os resultados obtidos tenham a presença da carga depositada por partículas cósmicas, sua frequência de incidência é centenas de vezes inferior à frequência de pulsação dos LEDs.

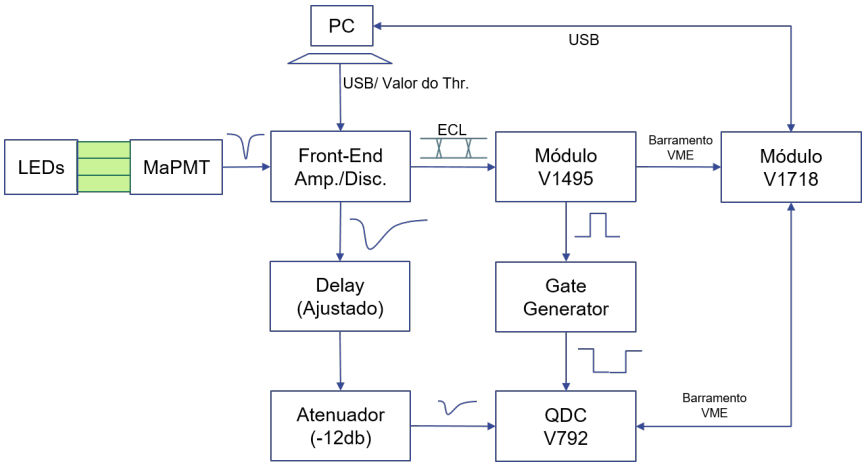


Figura 2.64: Diagrama de blocos do *setup* experimental para análise de carga depositado pelo sistema de injeção de luz.

Como resultado, obtivemos a seguinte análise da carga depositada pelos LEDs, depois de sua calibração.

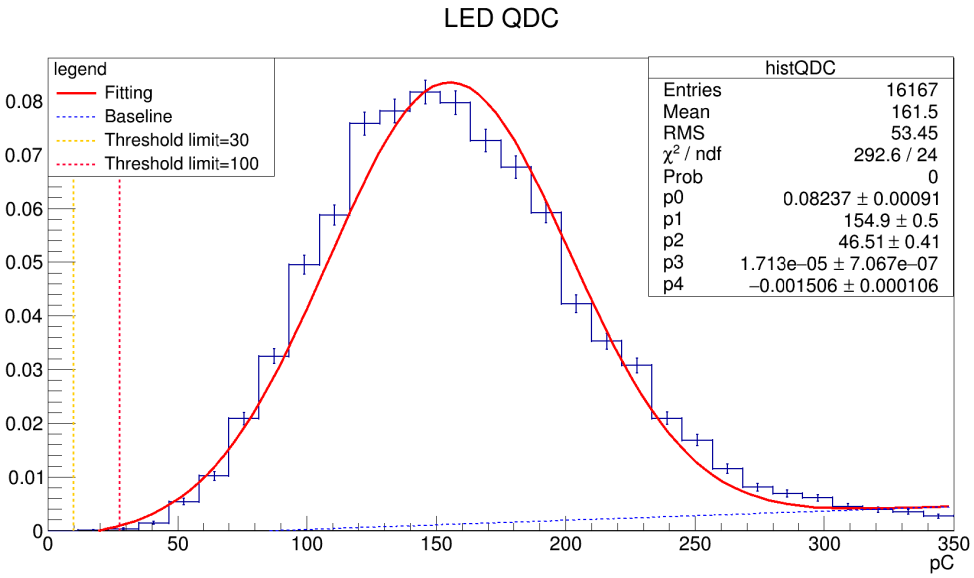


Figura 2.65: Histograma da carga depositada pelo sistema de injeção de luz.

2.9.8 Ponto Ideal de Alta Tensão

A MaPMT utilizada possui um ganho dependente da alta tensão que a alimenta. Sendo assim, uma análise do seu ponto ideal é necessária para se obter a maior eficiência,

com o mínimo de corrente de escuro.

Para se obter o ponto ideal de alta tensão, foi realizada a aquisição para diversos valores da mesma, variando-a de -630V até -780V. Foi utilizado o valor de *threshold* de 30, obtido anteriormente.

Um software de análise foi desenvolvido para que possamos observar a eficiência de cada plano e fluxo de cada detector em função da alta tensão aplicada. Os resultados podem ser observados na Figura 2.66.

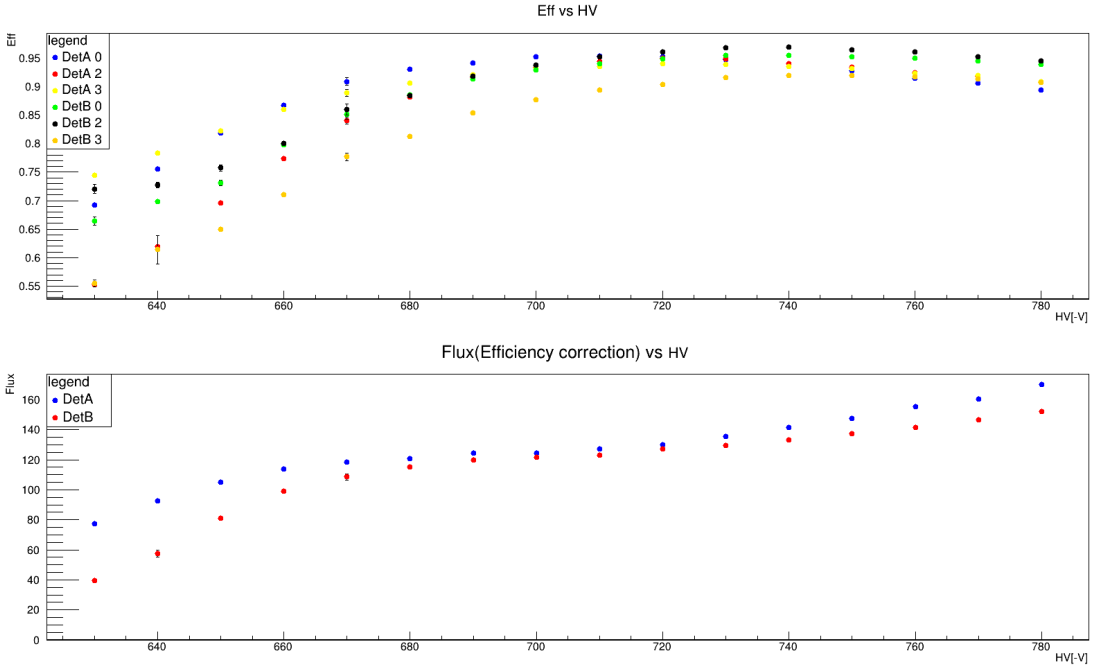


Figura 2.66: Gráficos de eficiência e fluxo de raios cósmicos para diferentes valores de alta tensão.

Com o aumento da alta tensão, o ganho de corrente de cada canal da MaPMT aumenta de forma diferente, o que implica em diferentes pontos ideais de alta tensão. Na análise, deve ser considerado o detector como um todo, já que essa multiplicadora não permite o ajuste individual da tensão de alimentação em cada canal.

Nos gráficos, é possível identificar um plateau de alta tensão de -680V até -720V, onde o fluxo se mantém constante. A partir desse valor, o aumento da corrente de escuro e tensão referente ao *crosstalk* óptico faz com que o fluxo aumente e a eficiência dos detectores caia.

Analisando os valores desse plateau de alta tensão, consideramos que seu ponto

ideal é de -720V, onde obtemos um fluxo similar entre os dois grupos de detectores e o máximo de eficiência dos planos de detecção.

2.9.9 Resultados

Para testes e comparação dos resultados que serão obtidos no continente antártico com o detector do experimento CREAT2, foram feitas aquisições em lugares distintos do Rio de Janeiro. A primeira aquisição foi realizada no próprio CBPF, em novembro de 2017, e a segunda e terceira aquisição foram realizadas na UERJ, durante os meses de fevereiro e março de 2018. Vale ressaltar que o local onde o detector foi instalado no quinto andar do CBPF, e está à aproximadamente 25 metros acima do nível do mar, já na UERJ, ele se encontrava no subsolo, à aproximadamente 16 metros acima do nível do mar.

O objetivo inicial dos testes era realizar uma aquisição em temperatura ambiente no CBPF e na UERJ, e outra utilizando a câmara fria da UERJ, onde a temperatura seria variada para ser observado o comportamento do detector. Entretanto, a mesma se danificou antes mesmo destes testes serem realizados, não podendo ser utilizada até então.

Em ambas as aquisições, o detector foi configurado com tempo de aquisição de 30 segundos, *GATE* de 10 pulsos de clock ($250ns$), *threshold* de 30 (em registro de contagem) e alta tensão de -720V. Os resultados dessas aquisições estão apresentados nas Figuras a seguir, onde os histogramas em cinza representam os resultados coletados no CBPF durante o período de novembro de 2017, e em azul e amarelo, os da UERJ, referentes a fevereiro e março de 2018. A nomenclatura utilizada nos gráficos, referencia o 1º plano de detecção, o plano auxiliar, o 3º plano de detecção e o 4º plano de detecção, com as letras k, l, m e n, respectivamente.

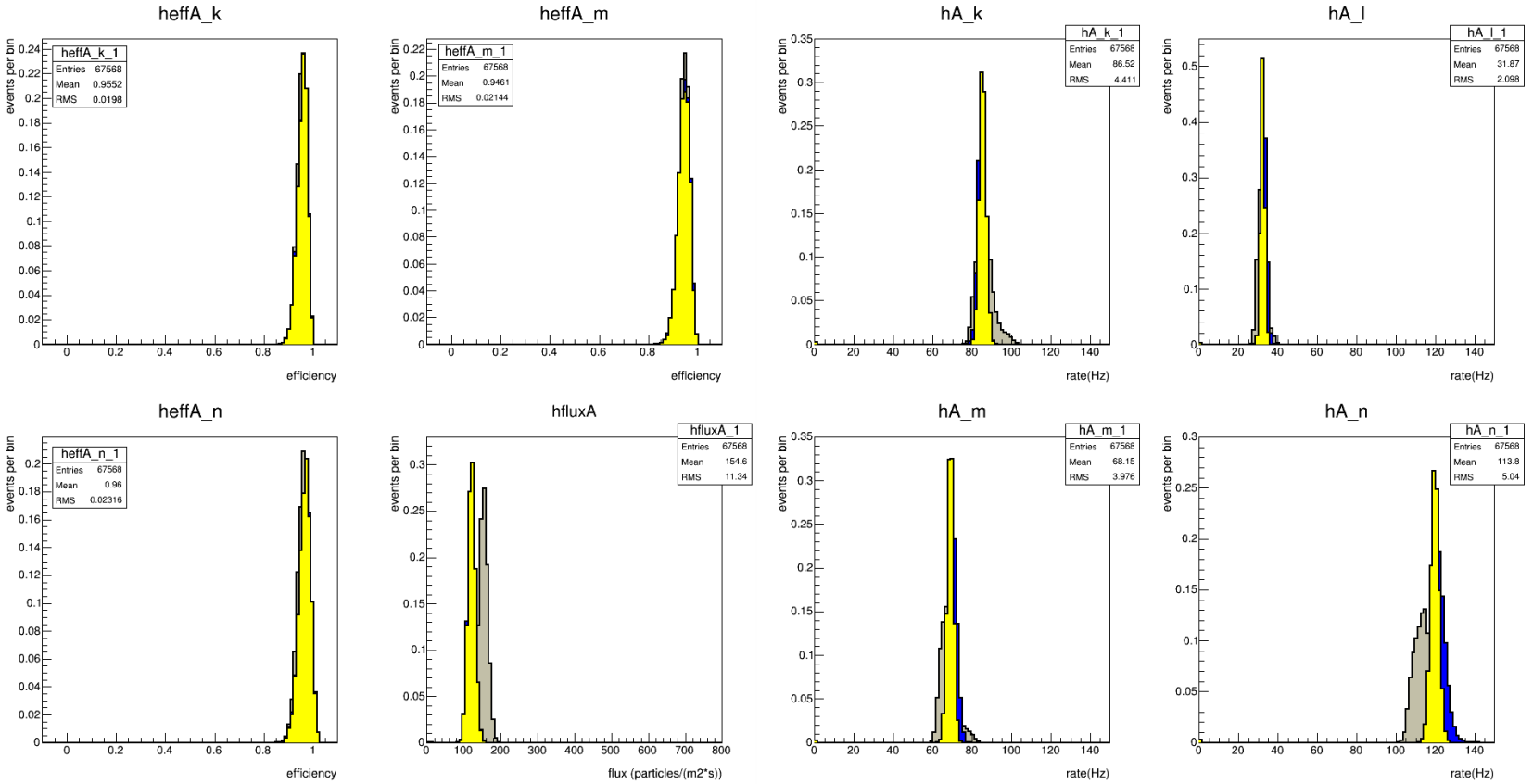


Figura 2.67: Histogramas de eficiência, fluxo e taxa de contagem de cada plano de detecção do detector A.

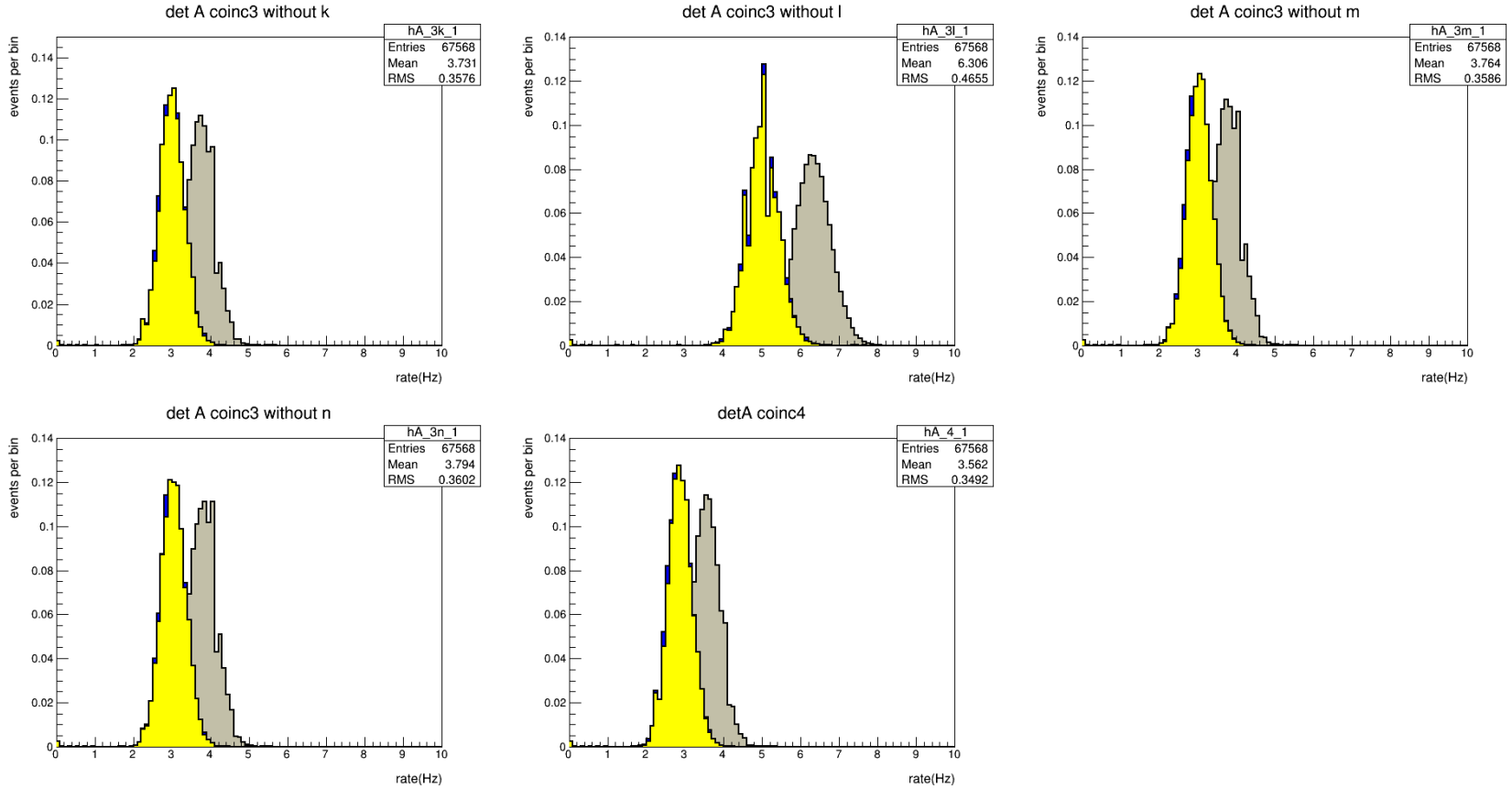


Figura 2.68: Histogramas da taxa de contagem da coincidência tripla entre três planos de detecção, e quádrupla de todos os planos de detecção do detector A.

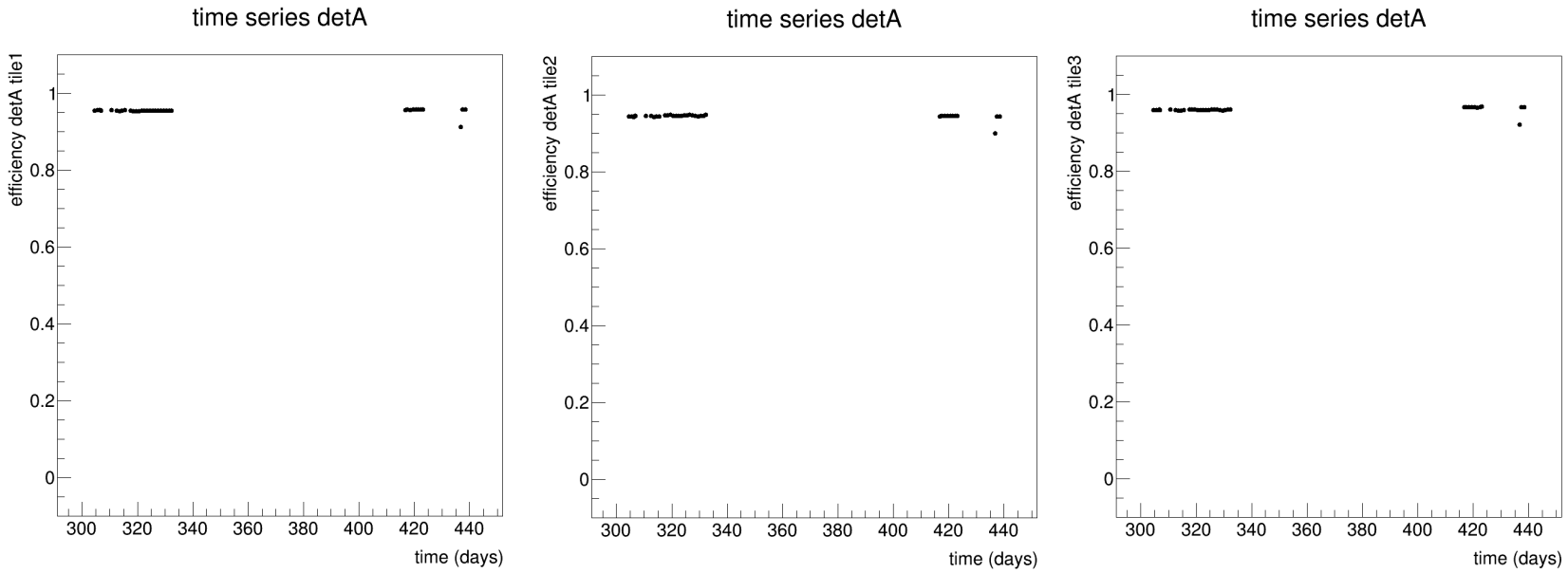


Figura 2.69: Série temporal da eficiência de todos os planos de detecção do detector A.

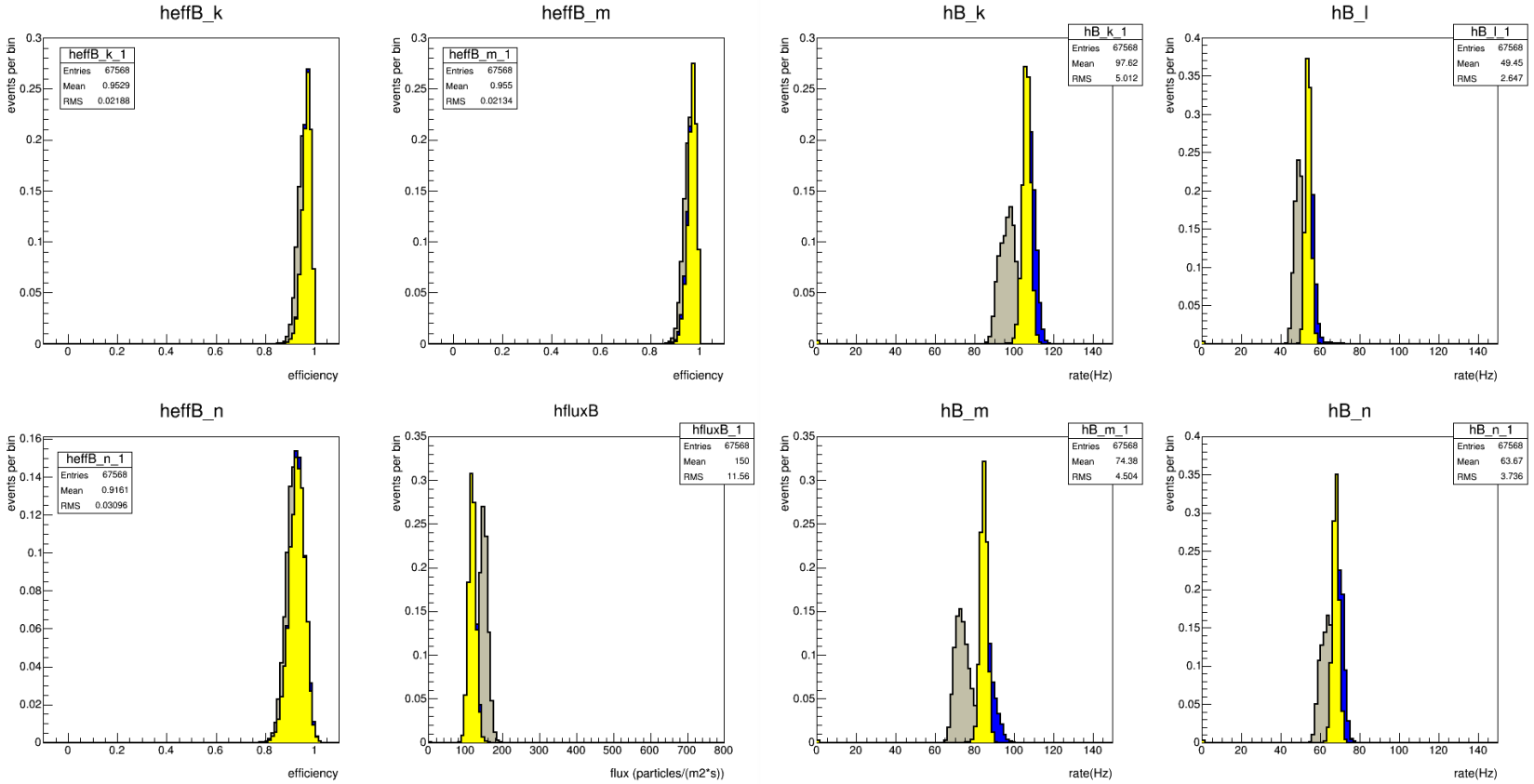


Figura 2.70: Histogramas de eficiência, fluxo e taxa de contagem de cada plano de detecção do detector B.

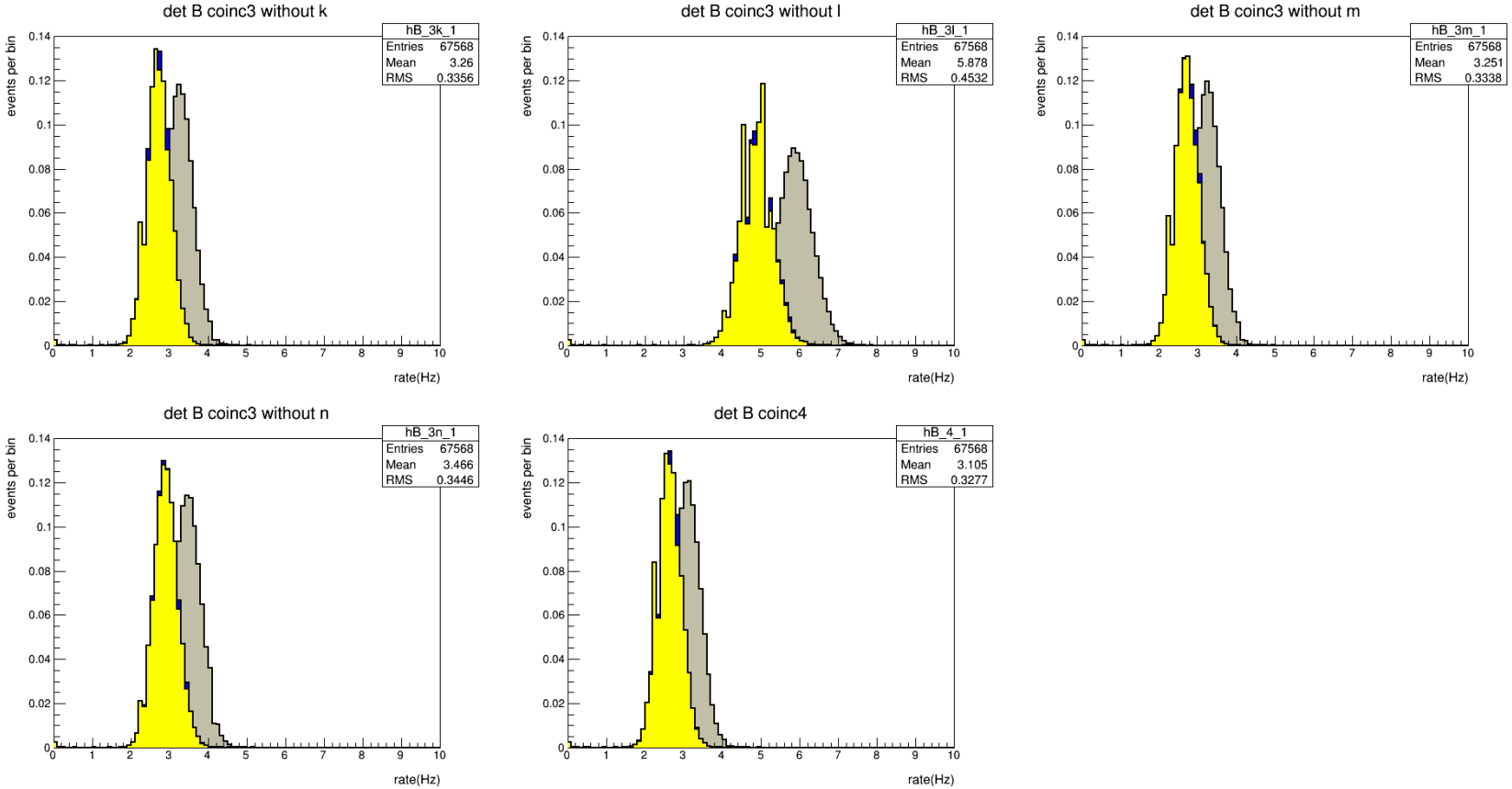


Figura 2.71: Histogramas da taxa de contagem da coincidência tripla entre três planos de detecção, e quádrupla de todos os planos de detecção do detector B.

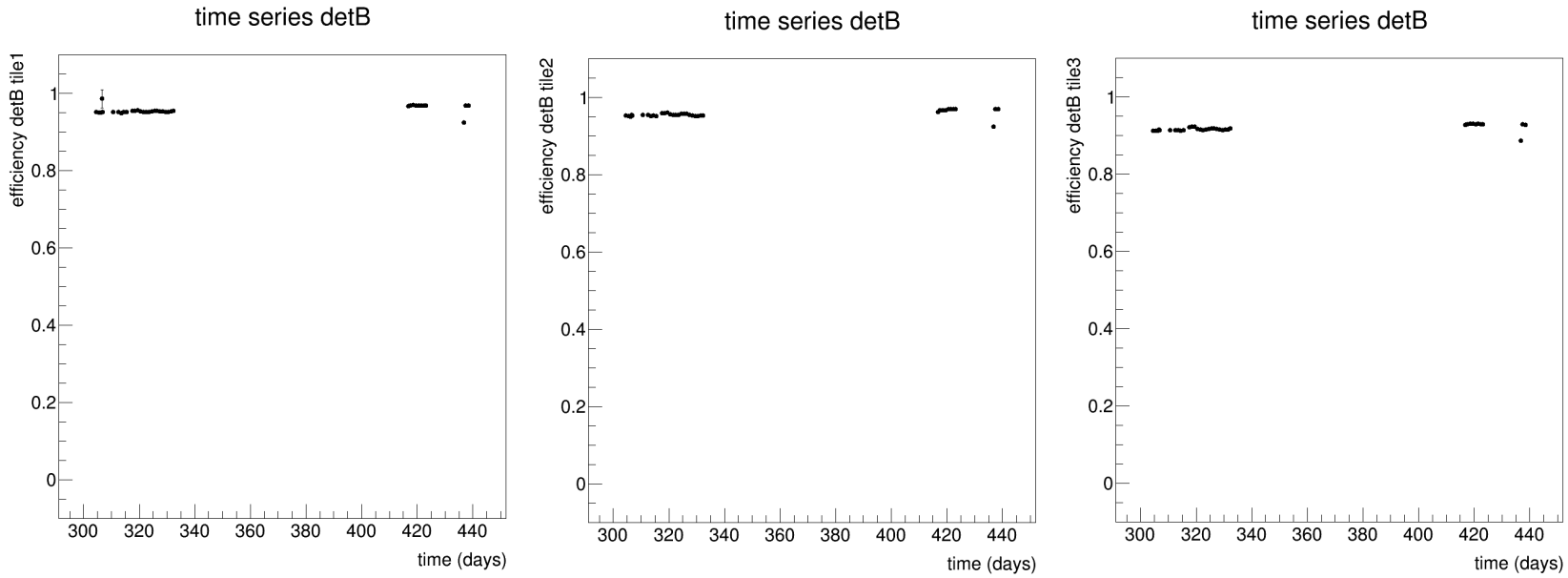


Figura 2.72: Série temporal da eficiência de todos os planos de detecção do detector B.

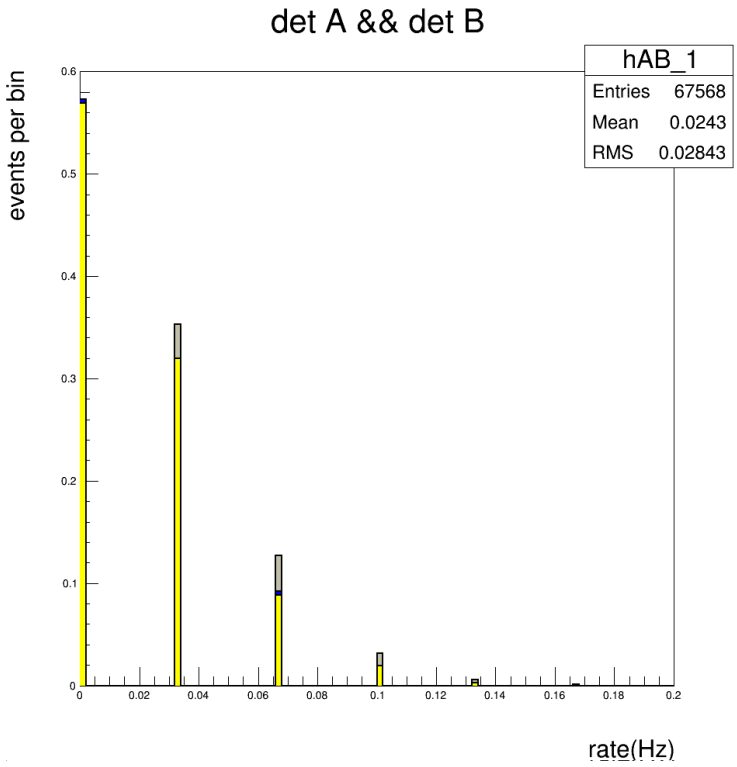


Figura 2.73: Histograma contendo a taxa de contagem da coincidência entre todos os planos de detecção, de ambos os detectores A e B.

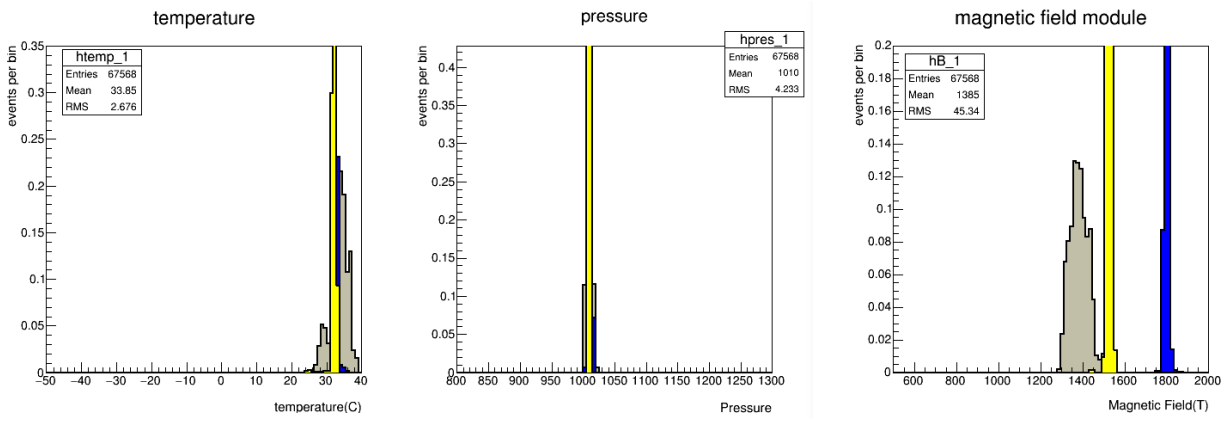


Figura 2.74: Histograma contendo os valores dos sensores de temperatura, pressão e campo magnético.

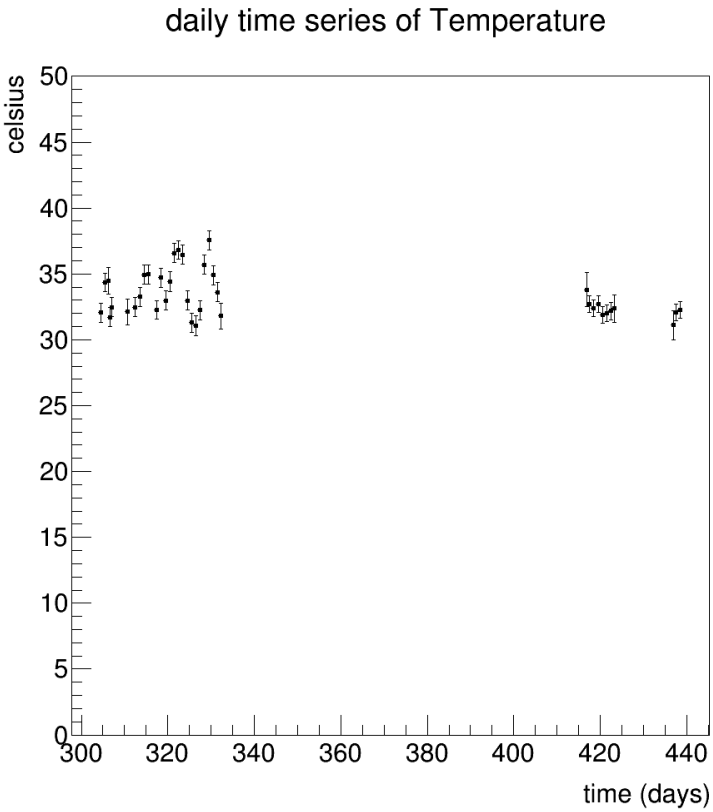


Figura 2.75: Gráfico de dias versus temperatura medido pelo sensor do experimento CREAT2 no CBPF e UERJ.

Através da análise individual de cada plano de detecção, podemos observar que ambos os detectores obtiveram resultados satisfatórios, garantindo uma ótima eficiência, de aproximadamente 95%.

Na Figura 2.76, podemos observar o fluxo de raios cósmicos medidos pelo CREAT2 no CBPF e UERJ.

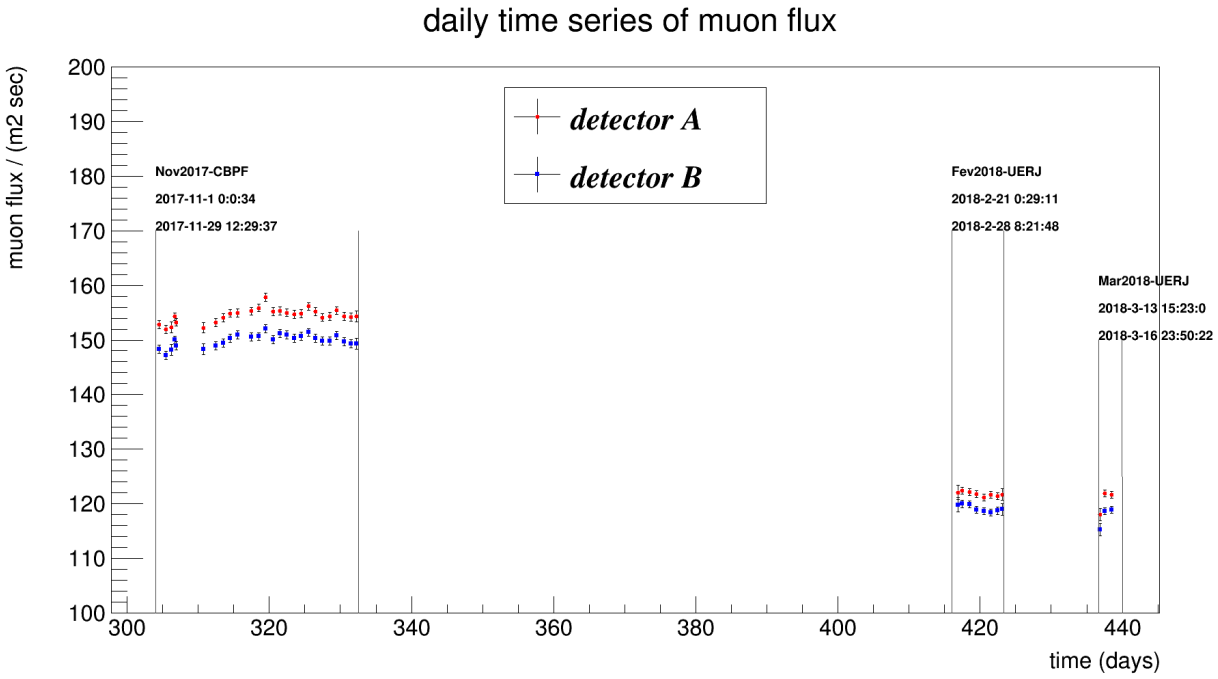


Figura 2.76: Gráfico de dias versus o fluxo de raios cósmicos medido pelo experimento CREAT2 no CBPF e UERJ.

Ao analisarmos os resultados de eficiência de cada conjunto de detecção, observamos que ela permaneceu constante em todo período de aquisição de dados. Entretanto, o fluxo medido segue uma oscilação natural, também observada nos resultados do CREAT1, coletados na Antárctica. Por serem períodos do ano diferentes, observamos um fluxo médio de RCG medido na UERJ cerca de 20% menor do que o medido no CBPF.

2.10 CREAT3

A primeira fase do experimento *CRE@AT* forneceu a experiência necessária para se trabalhar numa lógica complexa como o módulo Criosfera 1 e num ambiente complicado como o sul da Antártica. Depois de encaminhado o CREAT1 e 2, pudemos iniciar a 2ª

fase, que consiste no desenvolvimento e implementação de um novo detector que pudesse medir o modelo de produção angular dos raios cósmicos dessa região e sua influência em relação ao campo magnético terrestre e, em especial, ao campo magnético associado às partículas ionizantes de baixa energia emitidas pelos ventos solares.

A proposta deste novo detector é a realização da trajetografia de raios cósmicos. Utilizando a mesma tecnologia de cintiladores plásticos e uma MaPMT de 64 canais, o projeto propõe a construção de um telescópio composto por quatro planos de detecção, possibilitando a reconstrução da trajetória dos raios cósmicos.

2.10.1 Sistema Mecânico

Espaço físico é uma das maiores complicações no projeto dos detectores utilizados no experimento. Devido ao pouco espaço interno e tamanho da porta do módulo Criosfera 1, houve a necessidade de projetar uma estrutura grande e modular que pudesse ser desmontada e montada no interior do módulo. Para isso, a estrutura mecânica do detector de traços é composta por duas tampas e duas estruturas que se encaixem formando uma caixa com dimensões de aproximadamente $1,20m \times 1,20m \times 1,80m$. O material escolhido para a estrutura foi o polietileno preto de alta densidade, o mesmo utilizado nos demais detectores. A Figura 2.77 apresenta uma imagem em três dimensões da estrutura mecânica, projetada com o auxilio do técnico e estudante de engenharia Marcos Koebcke.

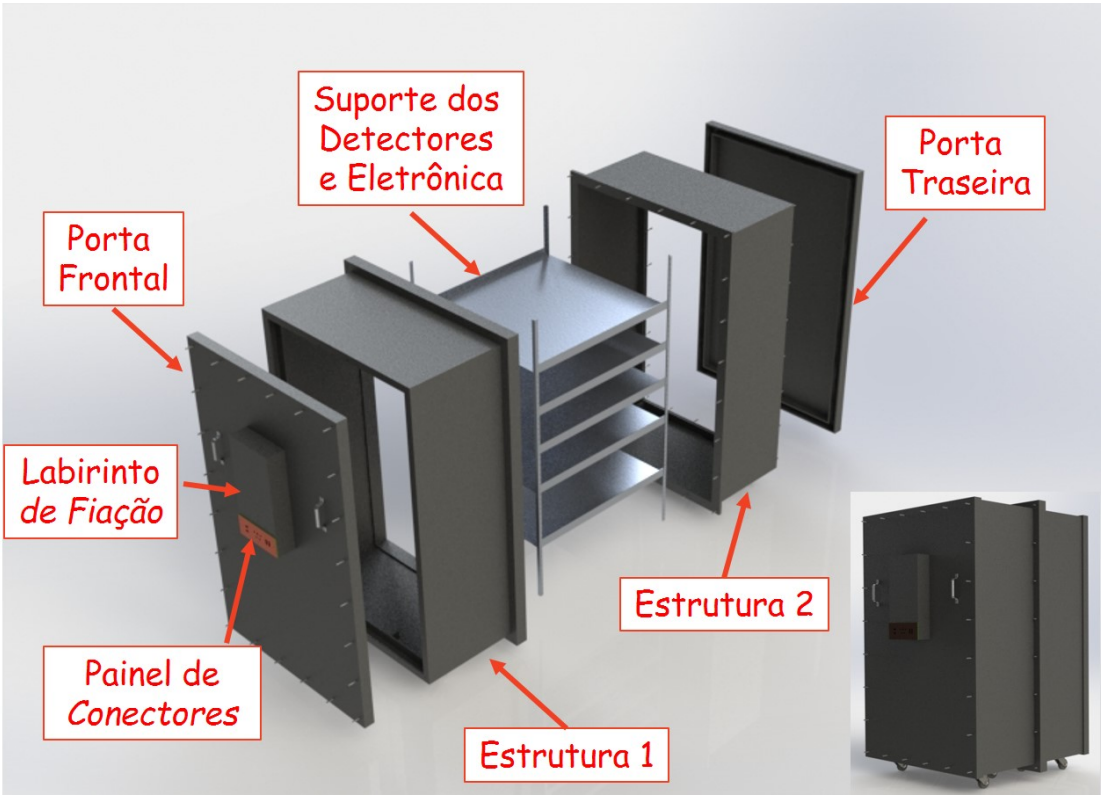


Figura 2.77: Desenho em três dimensões da estrutura mecânica do CREAT 3.

2.10.2 Detectores: PMT + WLS + Acoplamento Óptico

Um dos maiores desafios para a construção do sistema de trajetografia é o acoplamento óptico do conjunto: tira cintilante + fibra *wavelength shifter* + MaPMT. Devido à grande quantidade de canais (64) e o tamanho da estrutura do detector, são necessárias fibras WLS longas, podendo implicar na perda de eficiência. Outros aspectos importantes são: o raio de torção das fibras, em excesso pode diminui sua eficiência, além de danificá-las; e o contato das fibras com a MaPMT.

A MaPMT de 64 canais possui as mesmas dimensões que a de 16 canais, por isso, a alta densidade de células em sua superfície, utilizando fibras WLS do mesmo diâmetro, torna sua máscara ainda mais complexa.

2.10.3 Eletrônica de *Front-End*

Devido a complexidade de um circuito amplificador e discriminador de 64 canais, diferentemente do CREAT 1 e 2, optamos por utilizar a eletrônica de *front-end* baseada numa ASIC MAROC3 (*Multi Anode ReadOut Chip*) e sua placa de testes desenvolvidos pela Omega [42], centro de desenvolvimento de microeletrônica para física. Por já possuir todas a funcionalidades como amplificação, discriminação e digitalização de 64 canais em um único chip, sua utilização a torna financeiramente, e em relação ao tempo de trabalho, mais vantajosa que a eletrônica de *front-end* anterior.

Inserido na placa de teste do MAROC3, temos: um FPGA (*Field Programmable Gate Array*) Cyclone que, além de permitir o controle dos parâmetros da ASIC, possibilita as lógicas de coincidência e manipulação digital do sinais de entrada; um módulo de alimentação contendo 5 reguladores para fornecer as diversas tensões para seu funcionamento; chip ADC AD9220, responsável pela digitalização da saída analógica do MAROC3; e um chip controlador USB FTD245b para a comunicação com o computador através do USB[43].

2.10.4 MAROC3

O MAROC3 permite amplificar e discriminar 64 sinais independentes de uma fotomultiplicadora de polaridade negativa. Ela possui uma entrada para teste onde é possível multiplexar um sinal de entrada, possibilitando a realização de testes através de um gerador de função. Seus parâmetros são configurados de maneira serial, necessitando de um hardware e software de controle que estabeleça essa comunicação.

O diagrama funcional do MAROC3 está representado na Figura 2.78, e pode ser dividido em três blocos:

- Bloco de entrada e pré-amplificação;
- Bloco de leitura de carga;
- Bloco de *trigger*.

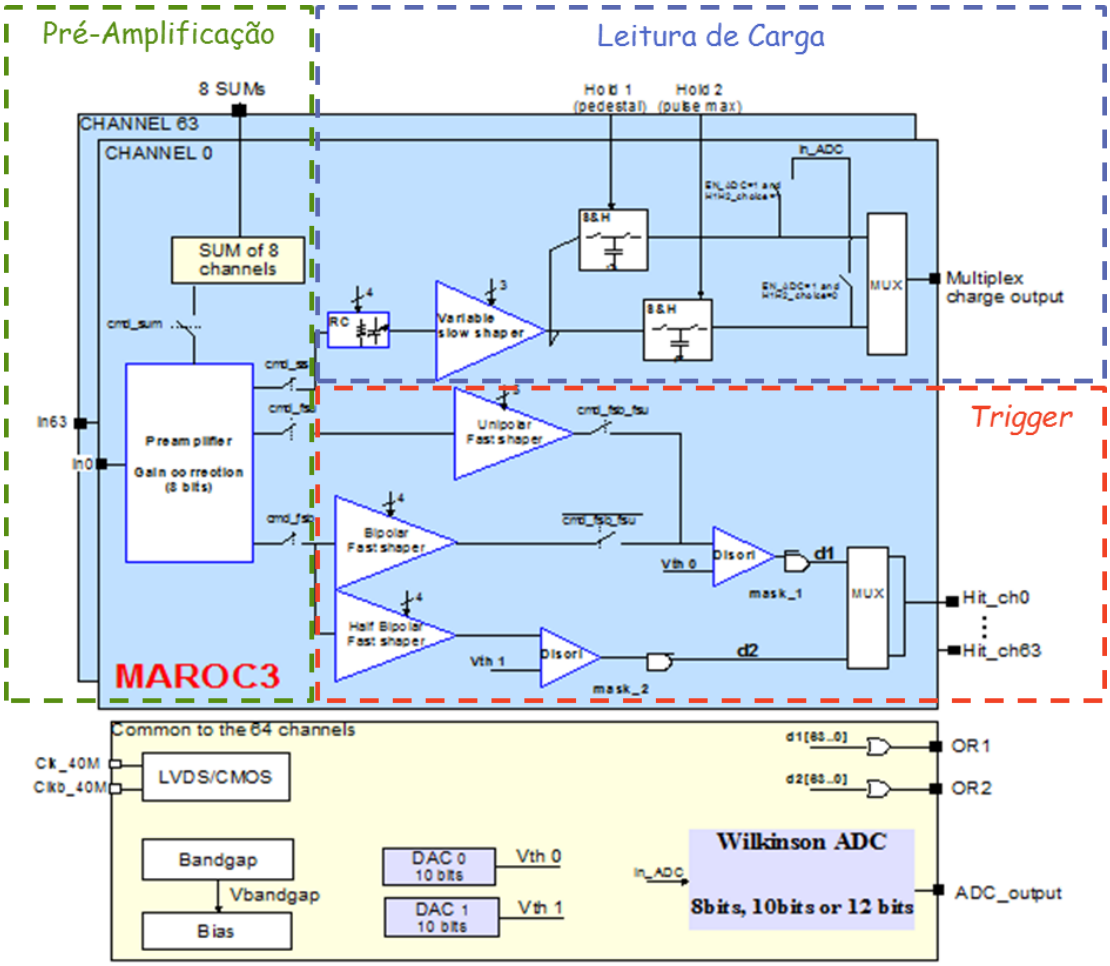


Figura 2.78: Arquitetura e bloco esquemático do MAROC3[43].

Bloco de Entrada e Pré-Amplificação

O bloco de entrada e pré-amplificação do MAROC3 é composto por 64 pré-amplificadores no modo super base comum, permitindo uma baixa impedância de entrada, entre 50Ω e 100Ω, baixo offset e baixa corrente de polarização ($5\mu A$) para minimizar o *crosstalk*. O ganho dos 64 pré-amplificadores é ajustável individualmente com uma precisão de *8bits*, o que permite compensar a diferença no ganho de cada canal da fotomultiplicadora. É possível obter um ganho de 0 à 3,984 ($= 1/64 + 1/32 \cdots + 1/2 + 1 + 2$), sendo que o valor 64 no registro de amplificação representa um ganho unitário, e 0 a inibição do sinal [43]. A Figura 2.79 apresenta seu bloco esquemático.

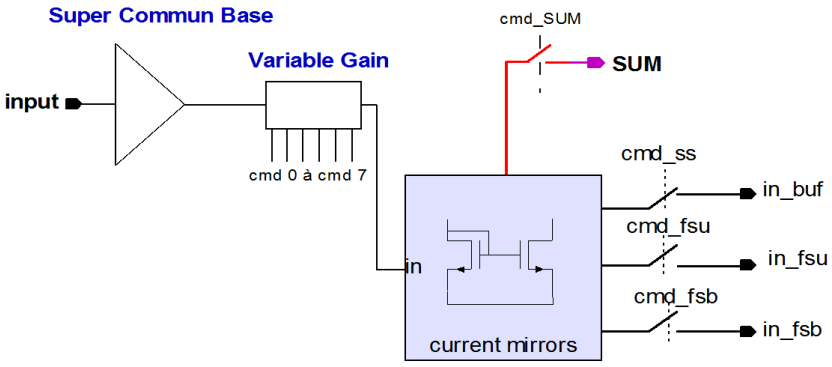


Figura 2.79: Bloco esquemático do circuito pré-amplificador [43].

Bloco de Leitura de Carga

A leitura de carga dos 64 canais é feita através de blocos de formatação lenta ligados as saídas de cada pré-amplificador. São constituídos por 64 circuitos CRRC capazes de formatar os sinais com larguras ajustáveis entre 30 e 210ns. A saída desses *shapers* produzem sinais com amplitudes proporcionais a carga de entrada, de até 30pC, sendo depois injetados em dois blocos *"Track and Hold"*.

Através de um sinal externo de "*hold*", é possível guardar uma memória analógica dos valores de pico e pedestal dos sinais em cada uma das unidades. Essas tensões são armazenadas em capacitores de 2pF, podendo ser multiplexadas e exportadas em uma saída analógica todos os 128 sinais (64 canais x 2 unidades *"Track and Hold"*). As saídas podem também ser digitalizadas por um ADC Wilkison de 12 bits integrado no MAROC3.

Os sinais de corrente da saída dos pré-amplificadores entram no bloco de medição de carga através de um espelho de corrente, gerando um sinal de amplitude de tensão proporcional a carga do sinal de entrada.

O sinal é injetado num *RC buffer*, Figura 2.80, com capacitâncias ajustáveis entre $0,25pF$ e $3,75pF$, para filtragem de altas frequências e ajuste de ganho. Com resistência de 50Ω fixa do filtro RC, a constante de tempo do circuito pode ser ajustada entre 125ns e 187ns.

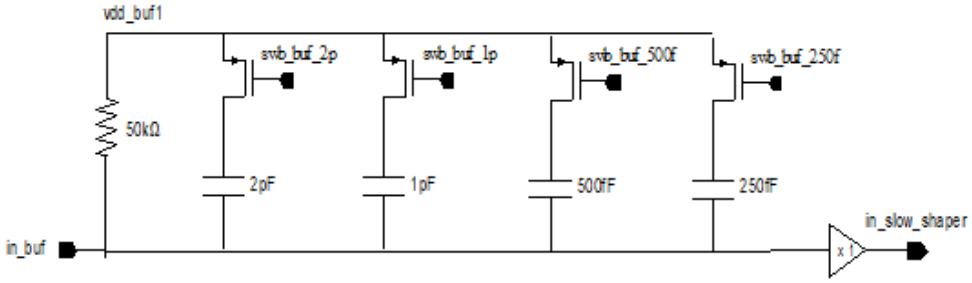


Figura 2.80: Esquemático do *RC buffer* [43].

O sinal de saída do *RC Buffer* é injetado no *Slow Shaper*, Figura 2.81, possibilitando o aumento da relação sinal-ruído através do ajuste de suas capacitâncias. Seu tempo de formatação pode ser ajustado entre $30ns$ e $210ns$.

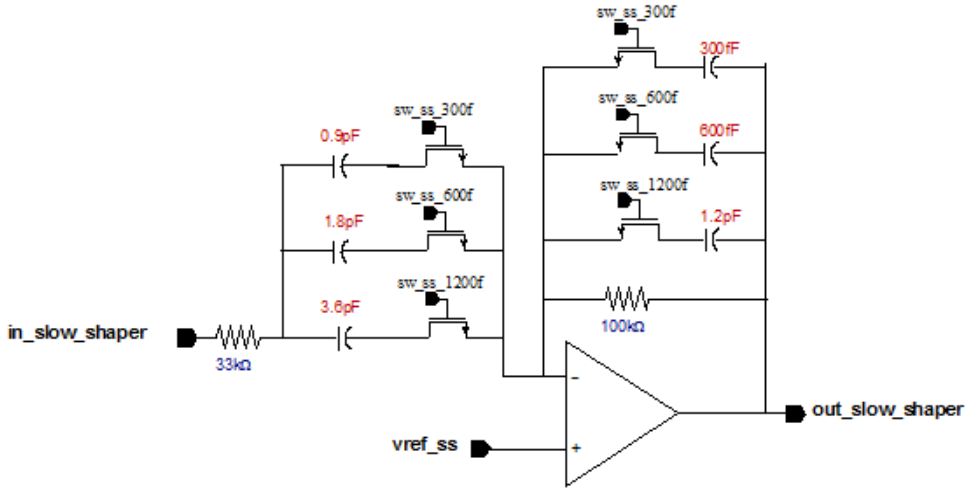


Figura 2.81: Esquemático do *Slow Shaper* [43].

As unidades *Track and Hold* seguintes registram em um capacitor de $2pF$ o valor da diferença de potencial do sinal formado no momento da transição negativa do sinal *HOLD1/HOLD2*, permanecendo guardado enquanto o sinal de controle seja baixo. No caso do sinal *HOLD* seja alto, é feito o *track* do sinal, ou seja, o valor do sinal de entrada será apresentado na saída. Existe um atraso de $80ns$ entre a transição negativa do sinal *hold* e o instante em que a unidade "*track and hold*"registra o valor analógico.

Bloco de *Trigger*

O bloco de *trigger* é composto por três circuitos de *shaper* de sinal. O *Unipolar Fast Shaper* e *Bipolar Fast Shaper* são responsáveis por gerar o *trigger* na presença de pequenos sinais de fótons únicos, obtendo 100% de eficiência na detecção de sinais na ordem de $10fC$. A saída de um desses *shapers* pode ser ligada no discriminador 0, comum a todos os canais. O *Half Bipolar Fast Shaper* tem um ganho inferior aos demais, pois é utilizado para geração de *trigger* na presença de sinais de maior carga, tendo este sua saída ligada ao discriminador 1, também comum a todos os canais. Os esquemáticos dos três *shapers* está apresentado na Figura 2.82.

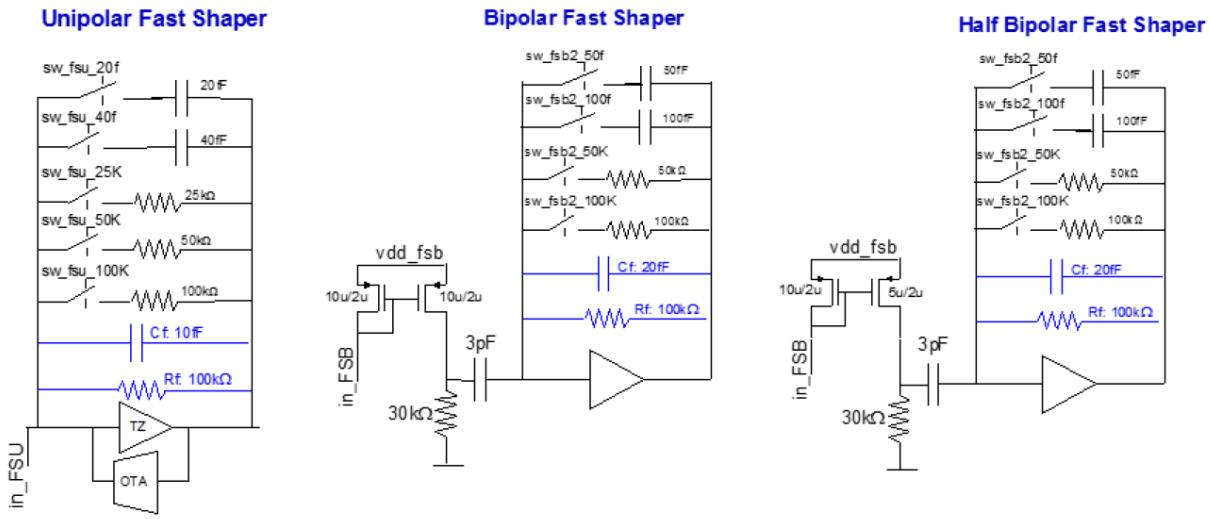


Figura 2.82: Esquemático dos *shapers* de sinal [43].

2.10.5 FPGA Altera Cyclone

O FPGA incluso na placa de testes do MAROC3 é um Altera Cyclone EP1C6Q240C6, responsável por todo o controle, configuração, aquisição e tratamento dos dados do MAROC3, além de permitir o interfaceamento com outros dispositivos, como o microcontrolador e o computador.

Os FPGAs da série Cyclone são de baixo custo comparados aos outros. O modelo utilizado foi o primeiro de sua linha, comercialmente disponibilizado em 2002, sua tecnologia é de $130nm$. Atualmente a série Cyclone já está em sua quinta geração, lançada em

2011 com tecnologia de $28nm$. A Serie Cyclone EP1C6, possui 5.980 elementos lógicos, 92.160 Bits de memória RAM e 185 pinos de entrada e saída [44].

A Seção 2.10.8 descreve os diversos blocos e funcionalidades do *firmware* do FPGA.

2.10.6 Modulo de Alimentação

O modulo de alimentação da placa de teste do MAROC3 provê várias tensões necessárias para seus diversos componentes. A partir de uma tensão típica, de $+6V$ e $-7,5V$, é possível alimentar seis reguladores com tensão máxima de alimentação de aproximadamente $\pm 20V$, fornecendo cada tensão indicada na Tabela 2.3.

Tabela 2.3: Tabela de tensões de alimentação.

Nome	Tensão
POWER SUPPLY	Typ: $+6V/-7,5V$ Max: $+20V/-20V$
DVDD	$3,5V$
AVDD	$3,5V$
VCCIO	$1,5V$
P1V8	$1,7V$
P3V3	$3,3V$
VCC	$5V$
-VCC	$-5V$

2.10.7 *Software*: *Firmware* + *Fast Monte Carlo* + Algoritmo de Trajetografia

Após tratamento e digitalização do sinal vindo da MaPMT, *softwares* de aquisição, armazenamento e análise dos dados precisaram ser desenvolvidos. Para isso, foram criados dois *firmwares*, um para o FPGA e outro para um microcontrolador ARDUINO MEGA, quatro *softwares*, sendo um de aquisição dos dados em tempo real pelo computador, um que realiza a trajetografia dos raios cósmicos, um responsável pela análise dos dados coletados e outro de simulação dos dados para teste e validação dos demais *softwares*.

2.10.8 *Firmware* do FPGA

Devido a existência de corrente de escuro e possíveis ruídos incoerentes e intrínsecos ao sistema (MaPMT + Eletrônica), há a necessidade de uma lógica de coincidência que gere um *trigger* primário capaz de selecionar apenas os eventos vindos do objeto de estudo. Essa lógica de *trigger* foi implementada em um FPGA CYCLONE presente na placa de teste do MAROC3. Seu *firmware* original foi substituído por outro projetado para suprir as necessidades do projeto, sendo ele composto pelos seguintes blocos:

- Dois blocos de comunicação (UART_Rx e UART_Tx) que permitisse realizar o *slow control* da placa e envio dos dados para o microcontrolador;
- Dois blocos de envio e recebimento dos dados via comunicação UART (SendDataUART e ReceiveDataUART);
- Um blocos de comunicação com o MAROC3 para realizar o *slow control* (SendDataMAROC3);
- Um bloco de coincidência (Coin_OUT), capaz de identificar eventos desejáveis;
- Um bloco de Alongamento do sinal digital (Along_trig) dos 64 canais para n ciclos de *clock*, corrigindo, assim, o sinal de coincidência prejudicado pela defasagem entre esses sinais;

Comunicação de UART

A placa de teste do MAROC3 possui um chip de comunicação USB e um bloco já pronto de interfaceamento. Entretanto, devido a necessidade de comunicação da placa com um microcontrolador, foi desenvolvido os blocos de comunicação de UART, similar ao usado no CREAT 1 e 2.

O UART (*Universal Asynchronous Receiver/Transmitter*) é um protocolo de comunicação serial muito utilizado em dispositivos embarcados. Ele permite a transferência e recebimento de dados de forma segura e com base na configuração do mesmo clock de operação entre dois dispositivos. As principais configurações do protocolo de UART, são:

- *BAUD RATE*: Onde é setada a frequência de transmissão de bits por segundo, sendo esta a mesma nos dois dispositivos em comunicação.

- Configuração do protocolo: O protocolo de UART pode ser configurado de diversas formas, tanto no tamanho do pacote de dados, podendo ser enviado 8 ou 9 bits, como na possibilidade de bits de paridade (par, ímpar, nenhum), com 1 ou 2 bits de parada. Para que a comunicação seja efetuada, há a necessidade que ambos os dispositivos possuam as mesmas configurações.

Os blocos do protocolo de UART possuem um *BAUD RATE* de 115200 $bits/s$, sem paridade e com 1 bit de parada. Seus códigos, assim como os demais, foram desenvolvidos em VHDL e podem ser encontrados no Apêndice H.

Os dados são convertidos para o ASCII (*American Standard Code for Information Interchange*) antes mesmo de serem enviados via serial. O ASCII é um código binário de 128 bits que representa 95 sinais gráficos, contendo letras do alfabeto latino, sinais de pontuação e matemáticos, e 33 sinais de controle. Foi escolhido utiliza-lo devido a ampla utilização em dispositivos, tanto microcontroladores como computadores. Cada conjunto de 4 *bits* dos registradores do FPGA são convertidos em um caractere correspondente da tabela ASCII em hexadecimal (0 a F). Devido à necessidade de *8 bits* do protocolo ASCII para o envio de dados de até 4 *bits* dos registradores, sua conversão diminui a velocidade de transmissão de dados. Entretanto, devido a baixa taxa de eventos por segundo do experimento, foi possível sua utilização sem grandes preocupações, facilitando assim o tratamento dos dados nos dispositivos receptores.

Registradores

Os dados enviados via serial contém informações como: uma palavra de 64 *bits* no formato hexadecimal, representando a existência ou não de *hits* em cada um dos 64 canais; informações do endereço único de cada evento para controle dos dados, sendo este de 16 *bits*; um registrador de 16 *bits* responsável pela informação temporal de cada evento. Esse ultimo registrador é incrementado a cada n pulsos de *clock*, ou seja, a cada $n \times 25ns$, sendo o valor n configurável. Dessa forma, é possível saber o tempo ocorrido entre cada evento e, assim, fazer medidas de fluxo, por exemplo. Cada transmissão de dados é realizada sempre que um evento de *trigger* ocorre, sendo ele gerado pela lógica de coincidência implementada.

Os dados recebidos pelo FPGA representam os diversos registradores de seu *slow control*. Neles, é possível configurar desde parâmetros do MAROC3, como o *threshold*, até parâmetros de configuração do próprio FPGA, como o tempo de alongamento do pulso

digital dos canais em pulsos de *clock*. Na Tabela 2.4 é possível observar o que representa cada registrador.

Tabela 2.4: Tabela de Registradores

Registrador	**Bit**	**Nome**	**Função**
Word_0	0	trigger_polarity	Define a polaridade dos sinais de trigger da saída disjuntiva
	1		
	2		
	3	sel_trig_ext*	
	4	en_serial_link	Ativar ou descativar a comunicação com o MAROC3
	5	gene_polarity*	
	6	Alt_trig_ext*	
	7		
Word_1	0		
	1	start_cycle	Iniciar o envio dos registos de controlo e leitura para o MAROC3 (1)
	2	RSTb	Reset dos blocos do FPGA
	3	RSTb_ADC_W	Sinal de controlo (Reset) do ADC interno do MAROC3
	4		
	5		
	6	RSTb_R	Sinal de controlo (Reset) para escrita do registo de leitura no MAROC3
	7	RSTb_SC	Sinal de controlo (Reset) para escrita do registo de controlo no MAROC3
Word_2	0	startadcb_in	Início manual da digitalização com o ADC interno o MAROC3
	1	Choice_start_external_adc	Controle do ADC externo
	2	Start_external_adc_man	Controle do ADC externo
	3	nb_Scurves[0]*	
	4	nb_Scurves[1]*	

	5	en_LVDS	Ativar conversor de nível do clock de 40Mhz (1)
	6	rstb_wilki_bloc	Reset do bloco StartADCWilkison e TransmitRAM
	7		
Word_3	0		
	1		
	2	EN_OTAQ	Ativação da unidade OTAQ do MAROC3
	3		
	4		
	5	EN_Test_Scurve*	
	6	Resetb_Scurve*	
	7	Sclckspeed*	
Word_4	[0..7]		
Word_5	[0..7]		
Word_6	0		
	1	EN_ALONG	Habilita o alongamento dos sinais dos 64 canais
	2	EN_UART	Habilita a comunicação UART e desabilita a comunicação com o *software* do LabVIEW
	3		
	4		
	[5..7]	Choose_coin_trigger	000 - Lógica OR entre todos os canais;
			100 - Coincidência OR entre detectores;
			010 - Combinação de coincidências AND entre dois detectores;
			110 - Combinação de coincidências AND entre três detectores;
			101 - Coincidência AND entre os quatro detectores;
N_gate	[0..7]		Largura em número de pulsos de *clock* dos sinais de cada canal;

DAC0	[0..7]		Valor de *threshold* DAC0;
DAC1	[0..7]		Valor de *threshold* DAC1;
C_Test	[0..7]		Habilita os canais em que é injetado o sinal ligado à entrada CTest da placa de teste do MAROC3;
OTHERS	[0..4]	choose_clk	Passo de incremento do Registrador temporal $2^n/1000$;
	5		
	6		
	7	small_dac	Aumenta a resolução dos DACs diminuindo a corrente de referência pela metade de sua intensidade;

Bloco de Coincidência e Geração de *Trigger*

Os ruídos e corrente de escuro precisam ser identificados para que não sejam confundidos com eventos de interesse. Dessa forma, uma espécie de triagem precisa ser realizada para que não haja grandes quantidades de dados para serem tratados pelo *software* de análise. Assim como no CREAT 1 e 2, um bloco de coincidência foi desenvolvido para solucionar esse problema e selecionar apenas eventos que serão estudados, gerando um *trigger* que permite o envio desses dados apenas quando uma condição é satisfeita.

Para compreender experimento em questão, é preciso primeiro entender como sua estrutura é montada. O CREAT3, como falado anteriormente, é composto por quarto detectores formados por dois planos de detecção, cada um com 8 conjuntos de tiras cintiladoras e fibras WLS. Cada plano de detecção representa um eixo cartesiano, que por terem suas tiras posicionadas ortogonalmente ao segundo plano, fazem com que haja um ponto x e y. O fato de termos 4 detectores iguais em alturas distintas nos dá o terceiro eixo, fornecendo quatro possíveis pontos para um único traço na coordenada cartesiana xyz.

Para descartar qualquer evento que não satisfaça sua condição, quatro lógicas de coincidência podem ser configuradas através dos *bits* de 5 a 7 do registrador Word_6, são elas:

- Lógica de *trigger* 0 - envia um pulso de *trigger* desde que haja um *hit* em pelo menos

um dos canais da fotomultiplicadora, permitindo salvar todo e qualquer evento que ocorra.

- Lógica de *trigger* 1 - envia um pulso de *trigger*, desde que haja, pelo menos, coincidência (AND) nos dois planos de um dos detectores.
- Lógica de *trigger* 2 - envia um pulso de *trigger*, desde que haja coincidência (AND) nos dois planos de, pelo menos, dois detectores, para isso, é feito uma combinação dois a dois entre os quatro detectores.
- Lógica de *trigger* 3 - envia um pulso de *trigger*, desde que haja coincidência (AND) nos dois planos de, pelo menos, três detectores, para isso, é feito uma combinação três a três entre os quatro detectores.
- Lógica de *trigger* 4 - envia um pulso de *trigger*, desde que haja coincidência (AND) nos dois planos dos quatro detectores.

Embora a lógica de coincidência seja muito eficaz quando se trata de ruído incoerente e corrente de escuro, um grande problema que ocorre na recepção dos sinais é o *crosstalk* elétrico ou óptico. Por ser uma interferência, possui mesma fase e, em alguns casos, amplitude do sinal interferente, não permitindo que seja discriminado ou até mesmo distinguido de um evento real pelo bloco de coincidência. Dessa forma, um método que minimiza os danos causados por esse fenômeno é o embaralhamento dos canais, de forma que os canais que possuem um *crosstalk* significativo não façam parte de uma mesma lógica AND, comprometendo sua coincidência. Para a escolha da posição de cada canal e seu respectivo plano e detector, é necessário um estudo minucioso considerando todas as possibilidades.

Bloco de Alongamento de sinal digital

Os sinais vindos dos circuitos amplificadores e discriminadores costumam ter larguras inferiores a $10ns$ e, devido à diferença de comprimento das fibras e distância entre os detectores, alguns sinais podem ficar defasados em relação a outros. Deste modo, optamos pela implementação de um bloco que aumente a largura dos sinais digitais, de forma que sua coincidência seja estabelecida mesmo nesses casos. A Figura 2.83 mostra com clareza a importância desse processo.

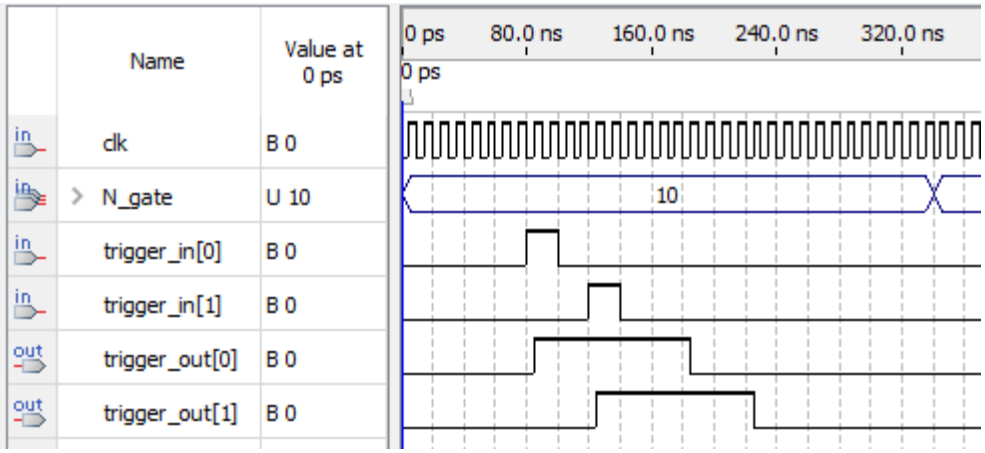


Figura 2.83: Sinais de entrada e saída do bloco.

A Figura 2.83 apresenta uma simulação realizada no Quartus II do bloco de alongamento dos sinais digitais, onde o sinal *clk* representa o *clock* do sistema, N_gate o registrador que configura a largura dos sinais de saída, em pulsos de *clock*, trigger_in[0] e trigger_in[1] dois dos 64 sinais de entrada, e trigger_out[0] e trigger_out[1] dois dos 64 sinais de saída.

2.10.9 Controle e Armazenamento de Dados

O *setup* experimental do CREAT 3 deve ser capaz de salvar os dados da aquisição de forma autônoma e sem a necessidade de um computador, evitando, assim, um maior consumo de energia e possíveis instabilidades correntes nos sistemas operacionais. Da mesma forma que foi realizado no CREAT 1 e 2, foi proposto que os dados seriam enviados para um microcontrolador ARDUINO MEGA e um shield ethernet com leitor de cartão SD, onde os dados seriam armazenados.

O *firmware* do microcontrolador tem a necessidade de ser capaz de receber e armazenar os dados vindos do FPGA, sendo reservada sua porta serial USART_0 para esse propósito, além de se comunicar com outros periféricos ligados a ela, utilizando a serial USART_0, que se comunica com um chip de conversão serial-USB. Suas principais funções são:

- Comunicação com o FPGA através de sua porta serial_1, para envio dos sinais de controle e recebimentos dos dados, onde esses dados são tratados e enviados para a porta Serial_0, podendo ser acessado por outros periféricos, como um computador;

- Os dados recebidos são armazenados em um cartão de memória, que com o auxilio de um módulo RTC (*real time clock*), é capaz de armazenar a data e hora de cada evento;
- Através de um módulo Ethernet, os dados armazenados podem ser suprimidos e enviados diariamente via TCP/IP para um servidor localizado no CBPF. O sistema *IRIDIUM Pilot* provê a internet necessária para o envio dos dados no Polo Sul;
- O Microcontrolador deve ser capaz de se comunicar com o sistema injetor de led, permitindo a realização de testes que validem o funcionamento do experimento.

2.10.10 *Software* de Aquisição e Armazenamento de Dados

Com o intuito de obter uma interface de aquisição e *slow control* mais amigável e de mais alto nível, foi desenvolvido um *software* em linguagem C para ser usado pelo computador. O *software* permite a aquisição de dados de forma paralela e independente do microcontrolador, sem que haja interrupção no armazenamento dos dados no cartão SD. Dois arquivos são gerado na aquisição, um de configuração, apresentando os valores dos registradores, mapa dos canais, dentre outros (Figura 2.84), e um similar aos aquivos salvos no cartão, de forma que não haja conflito na leitura de ambos pelo *software* de trajetografia e análise (Figura 2.85). O código do *software* de aquisição pode ser encontrado no Apêndice I.

Na Figura 2.84, em vermelho, temos os parâmetros físicos do *setup*, como: o mapa de cada canal, onde as linhas da matriz representam as posições de cada canal nos planos de detecção, sendo representadas pelos eixos x, y e z; em azul, temos os parâmetros de simulação, que por ser um arquivo comum aos três *softwares*, está presente, mas não é utilizado; e em verde, são salvos os valores de todos os registradores de controle.

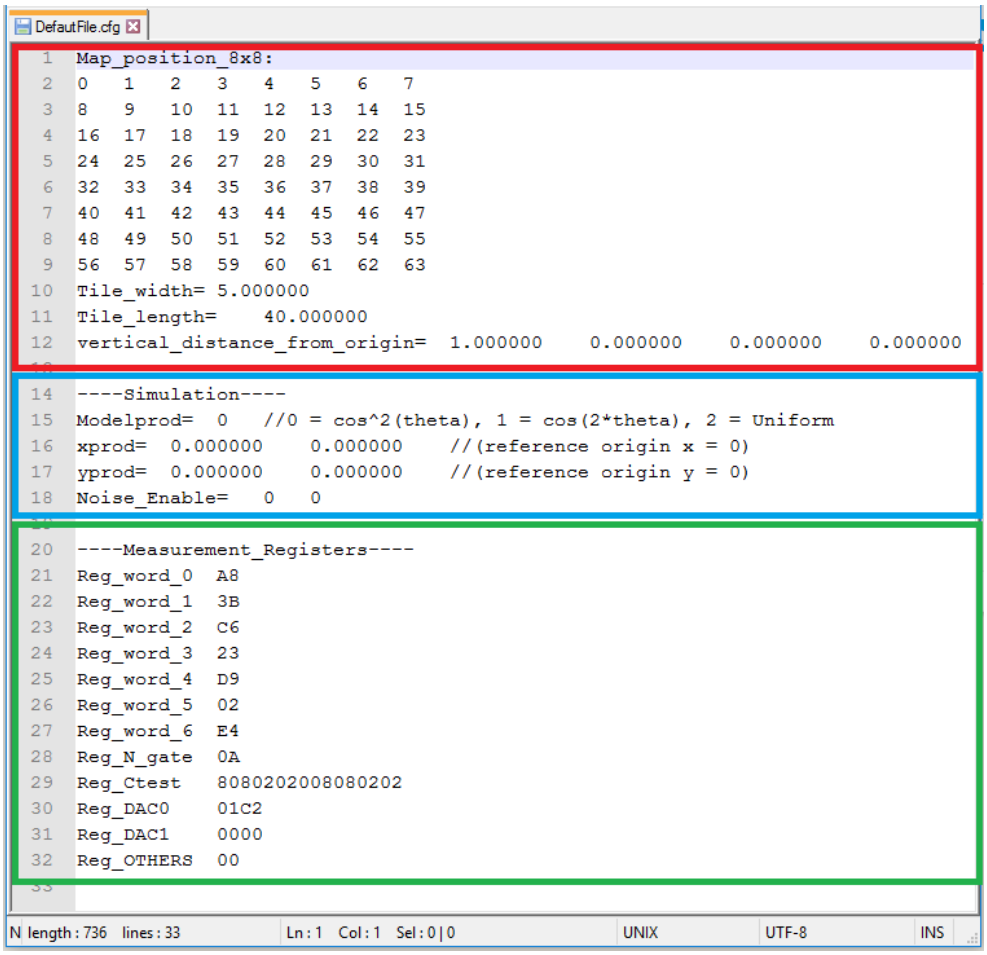
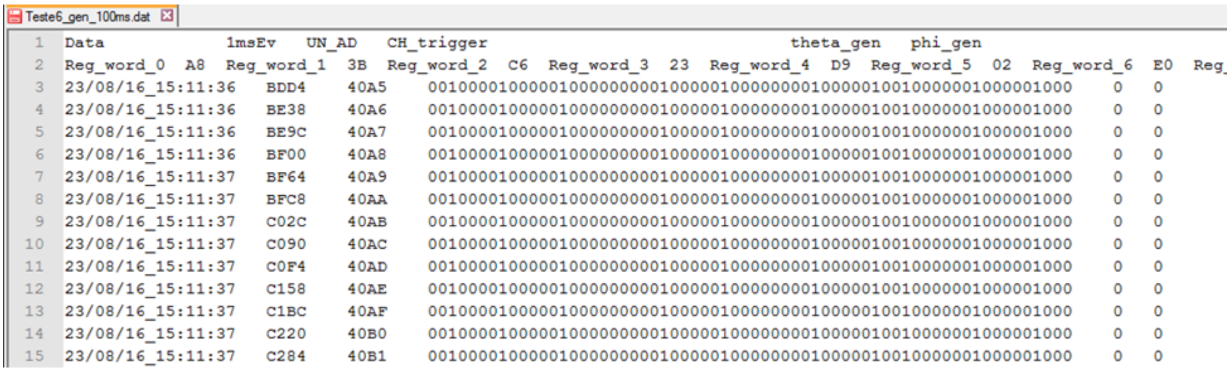


Figura 2.84: Arquivo de configuração (*.cfg)

A Figura 2.85 apresenta o formato dos dados que são salvos pelo *software*: a primeira linha do arquivo possui o cabeçalho; a segunda, o valor de todos os registradores de controle; e da terceira em diante, os valores de cada evento coletado, sendo que a primeira corresponde sua data e hora, a segunda, um registrador que informa o tempo entre cada evento, a terceira, um registrador de endereço, a quarta, os valores binários contendo os *hits* de cada canal, a quinta e sexta, valores exclusivos do *software* de simulação, informando, respectivamente, os ângulos θ e ϕ simulados.



```
Data             1msEv   UN_AD   CH_trigger                               theta_gen   phi_gen
Reg_word_0  A8  Reg_word_1  3B  Reg_word_2  C6  Reg_word_3  23  Reg_word_4  D9  Reg_word_5  02  Reg_word_6  E0  Reg_
23/08/16_15:11:36   BDD4   40A5   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:36   BE38   40A6   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:36   BE9C   40A7   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:36   BF00   40A8   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   BF64   40A9   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   BFC8   40AA   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   C02C   40AB   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   C090   40AC   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   C0F4   40AD   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   C158   40AE   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   C1BC   40AF   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   C220   40B0   001000010000010000000001000001000000000100000100100000001000001000   0   0
23/08/16_15:11:37   C284   40B1   001000010000010000000001000001000000000100000100100000001000001000   0   0
```

Figura 2.85: Arquivo de dados (*.dat)

2.10.11 *Software* de Simulação

Desenvolvido com base em um código de simulação *Fast Monte-Carlo*, criado pelo orientador do projeto, o *software* atual permite a simulação de eventos reais da trajetória de partículas cósmicas, sendo possível a inserção probabilística de ruído e/ou *crosstalk* nos canais adjacentes.

O *software*, também desenvolvido em linguagem C/C++, utiliza a ferramenta *Root* de análise de dados, desenvolvida no CERN. Os dados de simulação gerados, assim como no *software* original, podem seguir três modelos de produção: linear, $cos^2(\theta)$ e $cos(2\theta)$. A utilização de modelos de produção bem definidos nos permite avaliar a capacidade do *software* de análise de reconstruir o modelo original, função essa crucial para o experimento.

Os parâmetros da simulação são configurados no arquivo de configuração comum a todos os *softwares*, como mostrado em azul na Figura 2.84 da Subseção 2.10.10. O primeiro parâmetro de simulação é o modelo de produção, o segundo e terceiro parâmetros são a área onde são gerados os traços na altura do detector 0, o quarto parâmetro habilita a geração de ruído e/ou *crosstalk* em cada canal, sendo suas probabilidades de *hits* configuradas em seu código. No Apêndice J é possível observar o código do *software* de simulação.

2.10.12 *Software* de Trajetografia

Os dados obtidos, seja na aquisição real ou simulação, precisam ser tratados e convertidos em formatos lidos pelo *software* de análise. O código de trajetografia (*Tracking*) recebe os dados de cada evento e, com o auxilio do arquivo de configuração, identifica a

posição geométrica de cada canal. Em seguida, o programa é capaz de discriminar o ruído e *crosstalk* presente nos eventos legítimos, identificando o trajeto das partículas na coordenada esférica, como mostrado na figura 2.86.

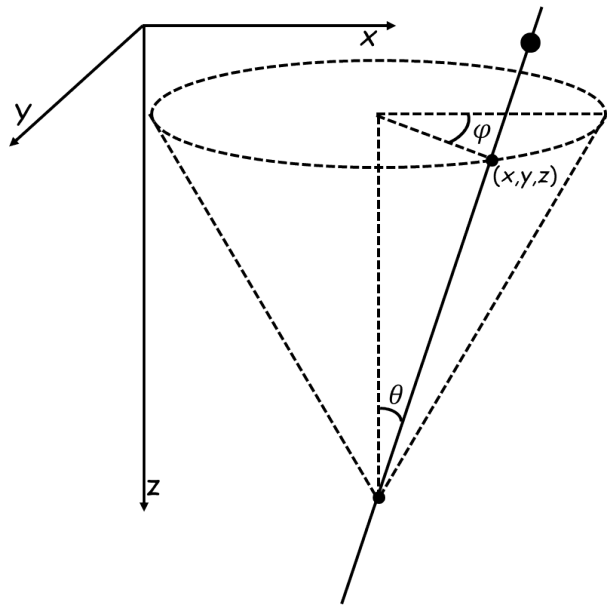


Figura 2.86: Representação de um traço no sistema de coordenada esférico.

O método utilizado para identificar as trajetórias é realizando o ajuste linear da combinação de todos os traços possíveis, considerando o menor erro quadrático médio como sendo da trajetória dos traços gerados. Na Figura 2.87, podemos observar, de forma ilustrativa, um exemplo de evento com ruído inserido em um dos canais do detector 0. Pode-se observar que o ajuste linear resultante de um traço ruidoso tem um erro quadrático médio superior ao de um evento real.

As informações são salvas em um objeto chamado *"Tree"* (árvore), contendo um conjunto de ramificações denominadas *Branches*. Cada *Branch* possui sua própria definição e lista de *buffers*, podendo ser arquivados automaticamente no disco ou mantidos na memória até que seu tamanho máximo seja atingido. Na tabela 2.5, são apresentados os *Branches* salvos pelo *software*. No Apêndice K é apresentado o código do *software* de trajetografia.

Tabela 2.5: Tabela de *Branches* do arquivo de saída.

Branch	**Descrição**

phi_sim	Valor do ângulo ϕ simulado, no caso de uma simulação.
theta_sim	Valor do ângulo θ simulado, no caso de uma simulação.
n_det	Numero de detectores pelos quais o traço passou.
hit_vector	Vetor contendo a existência de *hit* em cada canal.
*x_det0	Posição no eixo x, em centímetros, do primeiro ponto de ocorrência do traço, ou seja, detector mais próximo da origem.
*y_det0	Posição no eixo y, em centímetros, do primeiro ponto de ocorrência do traço, ou seja, detector mais próximo da origem.
*z_det0	Posição no eixo z, em centímetros, localizado o detector mais próximo da origem, em que ocorreu o traço.
*phi	Valor do ângulo ϕ calculado.
*theta	Valor do ângulo θ calculado.
*chisquaxz	Valor do erro quadrático médio do ajuste linear do plano xz.
*chisquayz	Valor do erro quadrático médio do ajuste linear do plano yz.
*err_phi	Valor do erro médio padrão do ângulo ϕ calculado.
*err_theta	Valor do erro médio padrão do ângulo θ calculado.
**ncells	Número de canais que o traço principal passou, utilizando os ângulos θ e ϕ calculados.
**ncellsdthetaP	Número de canais que o traço principal passou, utilizando os ângulos $(\theta + \delta\theta)$ e ϕ calculados.
**ncellsdthetaM	Número de canais que o traço principal passou, utilizando os ângulos $(\theta - \delta\theta)$ e ϕ calculados.
**ncellsdphiP	Número de canais que o traço principal passou, utilizando os ângulos θ e $(\phi + \delta\phi)$ calculados.
**ncellsdphiM	Número de canais que o traço principal passou, utilizando os ângulos θ e $(\phi - \delta\phi)$ calculados.

*Resultados obtidos para os três melhores traços, ou seja, os três menores erros quadráticos médios.

**Resultados obtidos para o melhor traço, ou seja, o menor erro quadrático médio.

2.10.13 *Software* de Análise

O *software* de análise (Apêndice L) é dividido em duas partes: análise de dados simulados e análise de dados reais. A análise dos dados simulados provê diversas informações

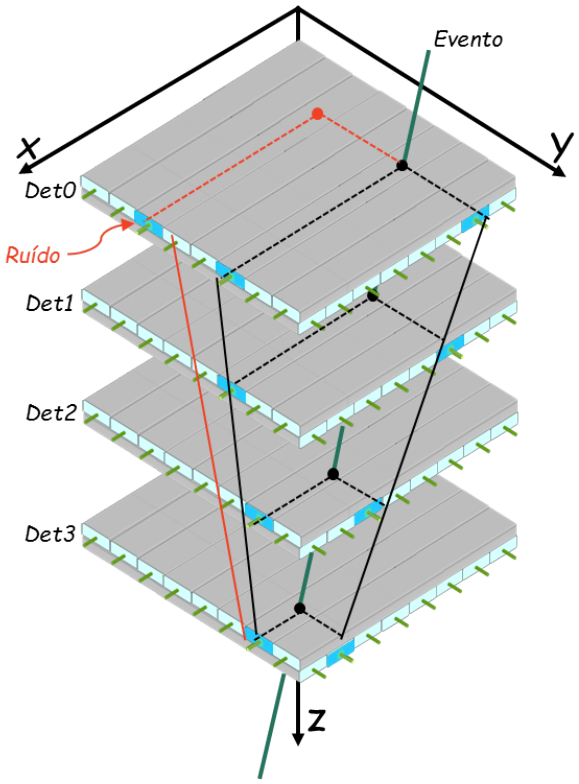


Figura 2.87: Representação de um evento real com ruído em um dos canais.

que validam os demais *softwares*, além de ajudar na identificação da melhor geometria e posicionamento de cada detector e seus respectivos canais. São gerados resultados como:

- Aceptância geométrica;
- Eficiência de cada canal;
- Diferença entre os valores reconstruídos e simulados de θ e ϕ;
- Modelo de produção reconstruído.

De acordo com a altura de cada detector, podemos aumentar ou diminuir a faixa de ângulos θ detectáveis, melhorando, assim, a aceptância geométrica do detector (probabilidade de uma partícula atravessar um ou mais detectores). Quanto menor a distância entre os detectores, maior será a aceptância, porém, menor a resolução de θ. O modelo da aceptância geométrica do sistema pode ser calculado dividindo-se o modelo dos ângulos reconstruídos pelos produzidos, como mostrado na Equação 2.13, valendo para θ e ϕ.

$$Hist_{Accep} = \frac{Hist_{rec}}{Hist_{prod}} \tag{2.13}$$

Sendo seus histogramas representados por: "$Hist_{Accep}$"o modelo da aceptância geométrica do sistema; "$Hist_{rec}$"o modelo dos ângulos dos traços reconstruídos; e "$Hist_{prod}$"do modelo dos ângulos dos traços produzidos.

A eficiência de cada canal é calculada dividindo a quantidade de *hits* efetivamente detectados nos canais, pela quantidade de *hits* que deveriam ser detectados. Sabe-se qual canal deveria haver um *hit* pela reconstrução do traço de um evento.

Através da diferença entre os valores reconstruídos e simulados de θ e ϕ, podemos obter o erro associado à segmentação do sistema de trajetografia. Esses valores nos ajudam a entender melhor a geometria e distância entre detectores, permitindo ajustar a resolução do ângulo θ.

De forma análoga ao modelo aceptância geométrica, o modelo de produção pode ser calculado dividindo-se o modelo dos ângulos reconstruídos pelo da aceptância do sistema, como mostrado na Equação 2.14, valendo para θ e ϕ.

$$Hist_{prod} = \frac{Hist_{rec}}{Hist_{Accep}} \tag{2.14}$$

2.10.14 Simulações

Embora já tenha sido projetado e desenvolvido o *firmware* e *softwares* de aquisição de dados do CREAT3, sua estrutura mecânica ainda falta ser produzida. Portanto, os resultados desse detector são apenas simulados para sua validação.

Com objetivo de obter a melhor altura entre os detectores, resultando em uma melhor resolução e aceptância geométrica, várias simulações com diferentes alturas entre os planos de detecção foram realizadas sem inserção de ruído ou *crosstalk*. Dentre elas, a que obteve melhor resultado singular foi utilizando distâncias de $40cm$, $80cm$ e $150cm$, em relação ao detector 0. A simulação foi realizada com base no modelo de produção $cos(\theta)^2$ e uniforme para ϕ, tendo sua origem um ponto aleatório na altura do detector 0.

Os histogramas da Figura 2.88 apresentam a diferença entre os valores dos ângulos θ e ϕ, simulados e reconstruídos pelo *software* de trajetografia. Em vermelho, amarelo e

verde, temos as contagens referentes à coincidências entre dois, três e quatro detectores, respectivamente. O erro nas medidas de ϕ são maiores devido à segmentação das tiras cintiladoras, já a distância entre os detectores melhora a resolução e consequentemente o erro do ângulo θ, pois a segmentação, além de ser menor no eixo z, acaba sendo desprezível em relação à distância entre os detectores.

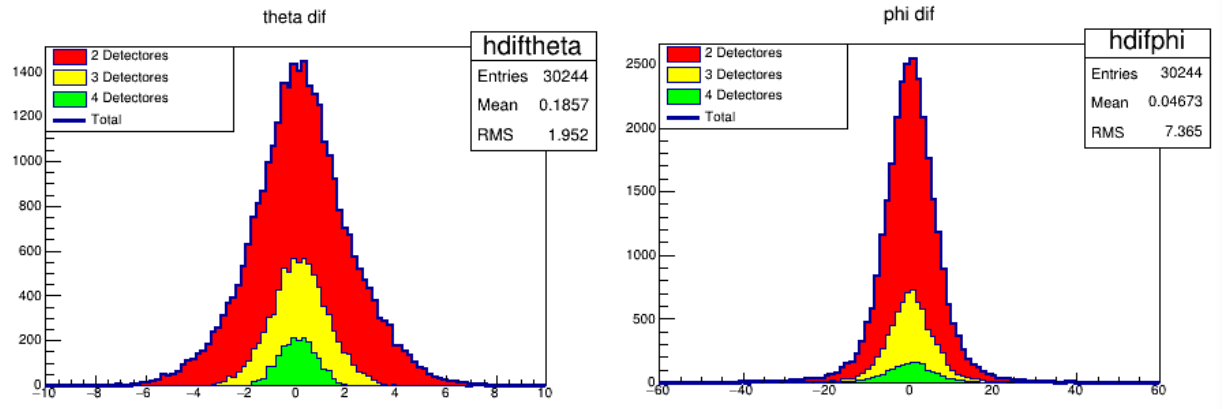


Figura 2.88: Gráfico de pilha (*stack graph*) contendo a diferença entre os ângulos simulados e reconstruídos de θ (esquerda) e ϕ (direita).

A Figura 2.89 apresenta os histogramas da aceptância geométrica de θ e ϕ. Nela podemos observar a influência das distâncias entre os detectores em sua aceptância e, assim, estabelecer a melhor geometria para o sistema. A aceptância geométrica é independente do modelo de produção e intrínseca à geometria dos detectores. Através dela, podemos observar que ângulos de incidência maiores que 50° não poderão ser detectados por essa geometria e apenas ângulos menores de 20° conseguem ser detectados por todos os planos de detecção.

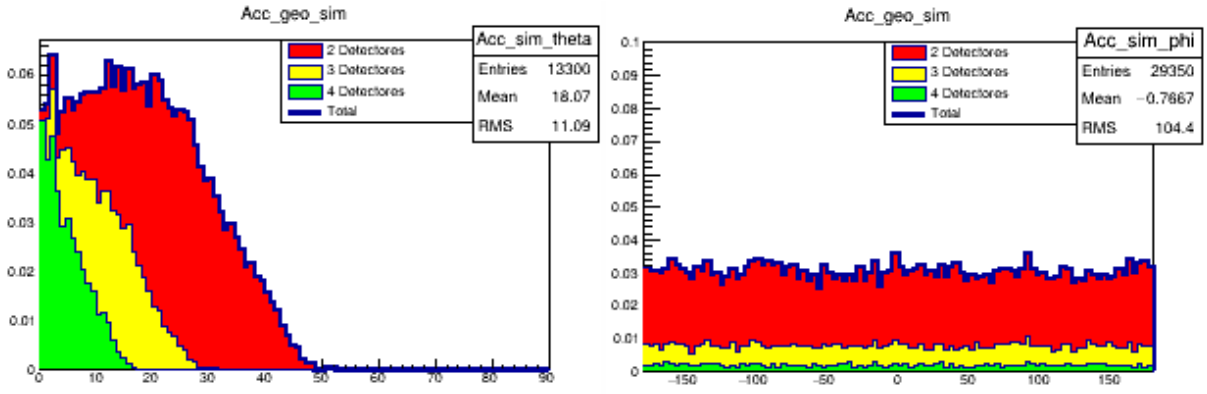


Figura 2.89: Gráfico de pilha (*stack graph*) contendo as aceptâncias geométricas de θ (esquerda) e ϕ (direita).

Através do modelo de aceptância geométrica de θ, podemos reconstruir o modelo de produção, na Figura 2.90. A função de ajuste utilizada foi $[p0] \times cos(\theta)^{[p1]} \times sen(\theta) \times cos(\theta)$, segundo o modelo de produção simulado $cos(\theta)^2 sen(\theta) cos(\theta)$ [41]. Podemos observar que o erro nos parâmetros de reconstrução é pequeno, validando a geometria do sistema.

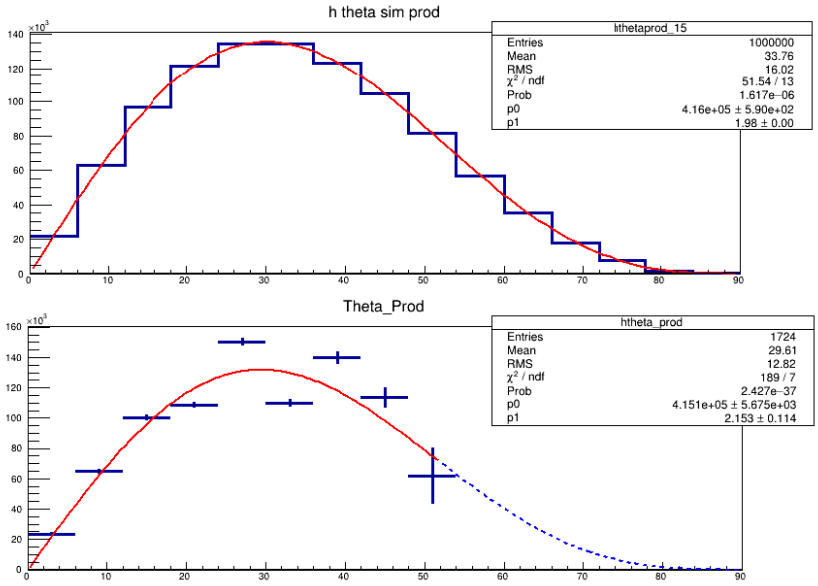


Figura 2.90: Histogramas relacionados ao ângulo θ da simulação de traços.

Capítulo 3

Conclusões

A observação da cobertura de nuvens de baixa altitude aparentemente relacionada com a incidência de raios cósmicos, em especial nos ciclos de 11 anos das atividades solares, nos leva a pensar na existência de uma relação até então desconhecida. Embora muitas teorias sejam especuladas, suas confirmações demandam de um longo estudo e análise de dados coletados por experimentos de diferentes áreas.

Através desta motivação foi criado o projeto CRE@AT (*Cosmic Ray Experiment at Antarctic*). O projeto propôs o desenvolvimento de dois experimentos: um dedicado ao fluxo de raios cósmicos na Antártica; outro dedicado a sua trajetografia. A escolha desse continente se deve às características únicas do local, como sua atmosfera reduzida, campo magnético diferenciado, baixa umidade, ausência de elementos pesados e o mínimo de interferência do homem em seu ecossistema, tornam esse continente um excelente laboratório natural.

A primeira fase do experimento contou com o envio do detector de raios cósmicos do experimento CREAT1 em 2014, versão piloto que validou a implementação do projeto no módulo avançado de pesquisa científica brasileiro Criosfera 1, que possui uma lógica de operação complexa num ambiente complicado como o sul da Antártica. Depois de sua instalação, foi iniciada a construção do detector do CREAT2, com o dobro da área de incidência de partículas do seu antecessor e uma organização que permite um melhor cálculo de eficiência de seus planos de detecção. Após conclusão da primeira fase do projeto, foi então iniciado o desenvolvimento do experimento CREAT3, projetado para detectar e realizar a trajetografia de raios cósmicos.

Quando experimento CREAT1 foi instalado, houveram alguns problemas energéticos no módulo Criosfera 1, cuja responsabilidade é do INCT da Criosfera e do Instituto Nacional de Pesquisas Espaciais, não permitindo que o detector tomasse dados de maneira contínua, problema esse que esperamos que seja resolvido em missões futuras.

Os resultados obtidos no continente antártico nos indicaram um fluxo de raios cósmicos medido entre 220 e 450 $muons/m^2s$, cerca de duas vezes maior que os obtidos no Rio de Janeiro. O maior fluxo detectado na região é devido a atmosfera reduzida e linhas de campo magnético ortogonais a superfície, permitindo que partículas de menor energia consigam entrar na atmosfera. Verificamos também uma variação na eficiência dos seus conjuntos de detecção no decorrer dos anos, principalmente nas missões que ocorreram em 2016 e 2017. Acreditamos que tal relação tenha ocorrido devido á dilatação térmica do material, desalinhando sua óptica e comprometendo sua eficiência. Dois dos cinco conjuntos de detecção tiveram seus resultados descartados, pois sua eficiência caiu de tal forma que seus fluxos não puderam ser corrigidos.

Na missão realizada em 2015, foi utilizada uma alta tensão de alimentação do detector de -750V, nos demais anos, essa tensão foi modificada para -700V. Essa mudança mostrou uma queda no fluxo de raios cósmicos medidos, indicando que partículas de mais baixa energia estavam sendo rejeitadas como o ruído e os valores escolhidos não estavam no plateau de alta tensão ideal do detector. Embora esses detalhes afetem as medidas absolutas deste detector, como medidas relativas seus resultados são válidos. A partir das análises dos resultados obtidos pelo experimento CREAT1, foi possível identificar seus pontos fracos, assim como corrigi-los na versão do CREAT2.

Com o detector do experimento CREAT2 foram realizadas duas medidas em locais e épocas do ano distintos. A primeira medida foi realizada no CBPF no período de novembro de 2017 e a segunda medida foi realizada na UERJ nos períodos de fevereiro e março de 2018. Por serem e épocas do ano distintas, o fluxo médio medido no CBPF foi de 150 $muons/m^2s$ e na UERJ foi de 120 $muons/m^2s$, ou seja, 20% menor. A eficiência observada em ambos conjuntos de detecção é de aproximadamente 95%.

Por ainda não ter sua estrutura construída, os resultados do experimento CREAT3 foram apenas simulados. Através das simulações, podê-se validar a *firmware* e *softwares* projetados para a reconstrução do modelo de produção de raios cósmicos, além de obter a altura ideal dos planos de detecção para se obter um boa resolução de θ com uma maior aceptância geométrica.

Para verificarmos a influência da temperatura na eficiência dos detectores, observada nos resultados do CREAT1, um estudo utilizando o CREAT2 na câmara fria da UERJ foi proposto. Entretanto, devido a alguns problemas elétricos, a câmara necessitou de manutenção, o que levou o adiamento destes testes. Comprovada a influência da temperatura na eficiência do detector, serão testados outros material com coeficientes de expansão linear menores que o polietileno para a produção das novas máscaras e *holder* de suporte das fibras WLS e MaPMT.

Após finalização dos testes com a câmara fria, faremos a instalação definitiva do injetor de luz, onde poderemos realizar, com maior precisão, as medidas de eficiência óptica e funcionamento do detector, visto que a frequência dos pulsos de luz injetados é fixa e controlada.

A conclusão de todos os testes do experimento CREAT2 está prevista para outubro de 2018, estando pronto para ser enviado à Antártica. Em seguida, será iniciada a montagem da estrutura do detector do CREAT3, cujo *firmware*, *software* de controle e aquisição, e eletrônica de *front-end*, já foram desenvolvidos.

Referências Bibliográficas

[1] F. Arnold, "Ion nucleation - a potential source for stratospheric aerosols,"Nature, no. 299, p. 134, 1982.

[2] J.-K. Z. a. F. Y. R.P. Turco, "A new source of tropospheric aerosols: ion-ion,"Geophys. Res. Lett., no. 25, p. 635, 1998.

[3] F. Y. a. R. Turco, "Ultrafine aerosol formation via ion-mediated nucleation,"Geophys. Res. Lett., no. 27, p. 883, 2000.

[4] Fauth A. C. Gonzalez L. Grizolli W. C. Kemp E. Penereiro. Consalter, D. M. Raios cósmicos detectados através de um telescópio de partículas.

[5] Daniel Martelozo Consalter et al. Estudo de raios cósmicos com e $>10^{18}$ev do detector de superfície do observatório Pierre Auger. 2009.

[6] Claudio Manuel Gomes de Sousa Sérgio Antônio Leitão do Vale. A importância do estudo dos raios cósmicos: O método de detecção do observatório Pierre Auger. 2012.

[7] Cinquenta anos do méson pi. Acesso em Novembro de 2013, disponível em http://www.ghtc.usp.br/meson.htm.

[8] AC Fauth, JC Penereiro, E Kemp, WC Grizolli, DM Consalter, LFG Gonzalez, et al. "Demonstração experimental da dilatação do tempo e da contração do espaço dos múons da radiação cósmica." *Revista Brasileira de Ensino de Física*, 2007.

[9] Climate4you, Solar activity. Acesso em Novembro de 2015, disponível em http://www.climate4you.com/Sun.htm.

[10] NN Das Gupta and SK Ghosh. A report on the Wilson cloud chamber and its applications in physics. *Reviews of Modern Physics*, 18(2):225, 1946.

[11] *Cloud Chamber*. David Darling. Acesso em Novembro de 2016, disponível em http://www.daviddarling.info/encyclopedia/C/cloud_chamber.html.

[12] *Virtual Cloud Chamber*. Nucleonica, 2018. Acesso em Novembro de 2016, disponível em https://www.nucleonica.com/wiki/images/5/5f/Cloud1.png.

[13] Alice Marlene Grimm. Meteorologia básica–notas de aula. *Primeira versão eletrônica, setembro*, 1999.

[14] F. Marcastel, CERN CLOUD research team adds new pieces to puzzle of cloud formation, 2011. Acesso em Novembro de 2015, disponível em http://phys.org/news/2011-08-cern-cloudteam-pieces-puzzle.html.

[15] Henrik Svensmark. Cosmoclimatology: a new theory emerges. *Astronomy & Geophysics*, 48(1):1–18, 2007.

[16] C. Faria, Antártida - Continente Gelado - Geografia - InfoEscola, Info Escola. Acesso em Novembro de 2013, disponível em http://www.infoescola.com/geografia/antartica-antartida/.

[17] Myhre, Jeffrey D. The Antarctic Treaty System: Politics, Law, Abd Diplomacy. Westview Press, 1986.

[18] Leonardo Faria de Mattos. A inclusão da antártica no conceito de entorno estratégico brasileiro. *CAPA-REVISTA DA EGN*, 20(1):165–191, 2016.

[19] Rogério M Gandra and JC Simões. Dialética científico-ambiental na geopolítica antártica: Repercussão no programa antártico brasileiro (proantar) iii [scientific and environmental dialectics in antarctic geopolitics: Repercussions in the brazilian antarctic programme]. *SIMPÓSIO NACIONAL DE GEOGRAFIA POLÍTICA REVISTA GEONORTE, Edição Especial*, 3(7):434–47, 2013.

[20] U. Carneiro, DESENVOLVIMENTO DO EXPERIMENTO ANTÁRTICO DE MONITORAÇÃO DE RAIOS CÓSMICOS PARA O MÓDULO CRIOSFERA I, CEFET/RJ, RIO DE JANEIRO, 2015.

[21] K.-E. Hines, Centralized Coincidence Trigger Processing for COMPET Using Both a Synchronous and an Using Both a Synchronous and an., 2012. Acesso em Novembro de 2015, disponível em https://www.duo.uio.no/bitstream/handle/10852/11079/hines_master_final.pdf?sequence=1.

[22] CERN, LHCb. Acesso em Novembro de 2015, disponível em http://lhcb-public.web.cern.ch/lhcb-public/en/detector/Calorimeters2-en.html.

[23] L. BARBOSA. Técnicas de detecção. *Notas de Aula*, 2004.

[24] Phototube test for miniplug calorimeter. Acesso em Novembro de 2015, disponível em http://hep.baylor.edu/hatake/miniplug/pmt_test.html.

[25] Hamamatsu Photonics, K. K. "Multianode photomultiplier assembly H6568, H6568-10, 1998."Technical Information.

[26] 800 mhz, 50 mw-current feedback amplifier-ad8001. Acesso em Março de 2017, disponível em http://www.analog.com/media/en/technical-documentation/data-sheets/AD8001.pdf.

[27] Ultrafast comparators ad96685/ad96687. Technical report. Acesso em Março de 2017, disponível em http://www.pa.msu.edu/ edmunds/HAWC/Manuals/-feb_ad96685_96687_comparator.pdf.

[28] B. Lawless. Fundamental digital electronics, unit 4 ecl emitter coupled. Acesso em Março de 2017, disponível em http://www.physics.dcu.ie/bl/digicont.html.

[29] 4-/6-channel- digital potentiometers-ad5204/ad5206. Technical report. Acesso em Março de 2017, disponível em http://pdf.datasheetcatalog.com/datasheet/analogdevices/1408 08745AD5204_6_0.pdf.

[30] 8-bit microcontroller with 4/8/16/32k bytes in-system programmable flash. Technical report. Acesso em Março de 2017, disponível em http://www.atmel.com/Images/doc8161.pdf.

[31] FLUKE, Fluke 8845A/8846A-Digital Multimeters, FLUKE. Acesso em Fevereiro 2017, disponível em http://support.fluke.com/calibration-sales/Download/Asset/2547535_6200_ENG_A_W.PDF.

[32] Tektronix, Arbitrary Waveform Generators. Acesso em Novembro de 2015, disponível em http://www.tek.com/datasheet/awg7000-arbitrary-waveform-generator.

[33] ICEL, MANUAL DE INSTRUÇÕES DA FONTE DIGITAL MODELO PS-6100. Acesso em Novembro de 2015, disponível

em http://www.composul.com.br/images/default/products/8a964-de86a9c57c4b7943fc300e3394a.pdf.

[34] Tektronix, MSO/DPO70000 Digital & Mixed Signal Oscilloscope. Acesso em Novembro de 2015, disponível em http://www.tek.com/oscilloscope/dpo70000-mso70000.

[35] CAEN Electronic Instrumentation, V1495-General Purpose VME Board. Acesso em Dezembro de 2015, disponível em http://www.caen.it/csite/CaenProd.jsp?idmod=484&parent=11.

[36] CAEN Electronic Instrumentation, V1718-VME-USB2.0 Bridge. Acesso em Dezembro de 2015, disponível em http://www.caen.it/csite/CaenProd.jsp?idmod=417&parent=11.

[37] ROOT Data Analysis Framework , About ROOT. Acesso em Abril de 2017, disponível em https://root.cern.ch/about-root.

[38] Wolfram MathWorld, Normal Distribution. Acesso em Dezembro de 2015, disponível em http://mathworld.wolfram.com/NormalDistribution.html.

[39] Robert M EISBERG. Física, fundamentos e aplicações. volume ii. 1a. edição. 1983.

[40] A Torres. Envelhecimento físico-químico de tubulações de polietileno de alta densidade empregadas em redes de distribuição de derivados de petróleo. *PUC-Rio, Rio de Janeiro*, 2007.

[41] BD'Ettorre Piazzoli, G Mannocchi, S Melone, P Picchi, and R Visentin. Aperture and counting rate of rectangular telescopes for single and multiple parallel particles. *Nuclear instruments and methods*, 135(2):223–233, 1976.

[42] P Barrillon, S Blin, M Bouchel, T Caceres, C de La Taille, G Martin, P Puzo, and N Seguin-Moreau. Maroc: Multi-anode readout chip for mapmts. In *Nuclear Science Symposium Conference Record, 2006. IEEE*, volume 2, pages 809–814. IEEE, 2006.

[43] S Blin et al. Maroc3 datasheet, october 2010. OMEGA website: http://omega. in2p3. fr.

[44] FPGA Cyclone. Family data sheet. 2003. Altera, março.

Appendices

Apêndice A

Esquemático da *Front-End* de 16 Canais

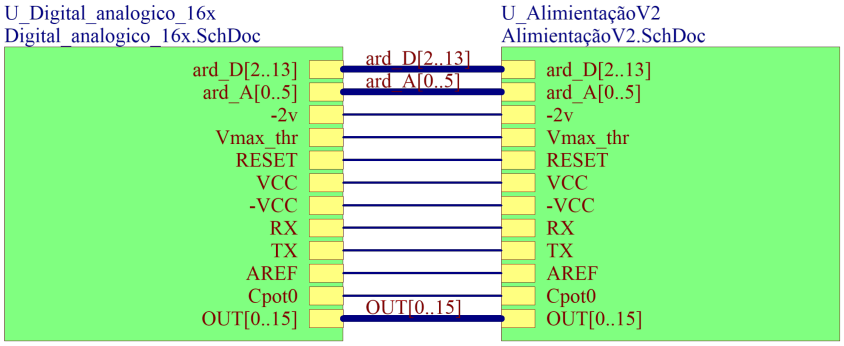


Figura A.1: Projeto principal: Main_Schematic.SchDoc.

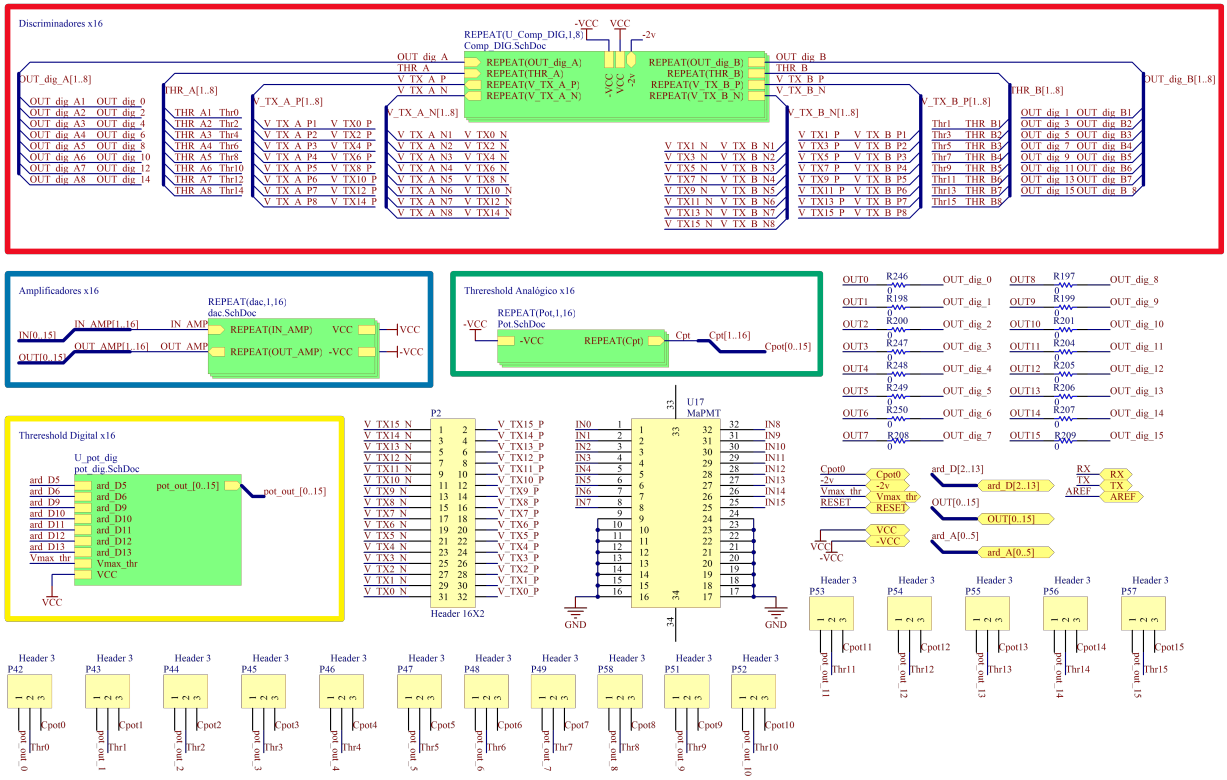


Figura A.2: Bloco Digital_analógico_16x.SchDoc.

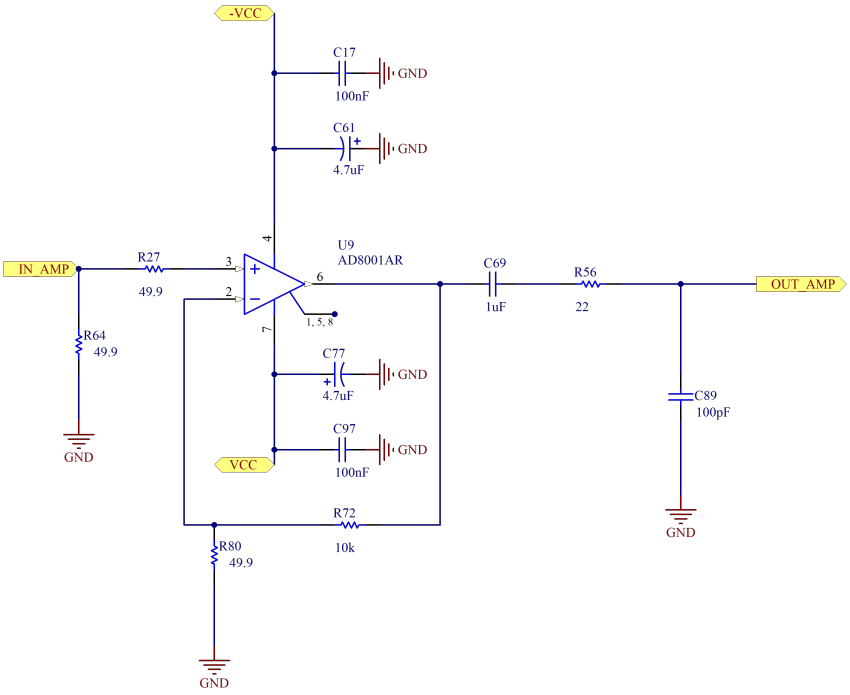


Figura A.3: Bloco dac.SchDoc.

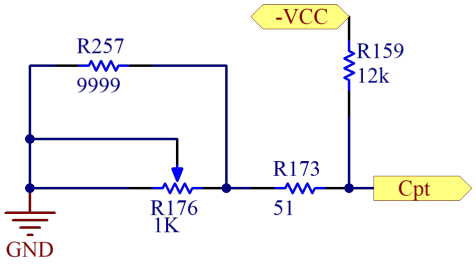


Figura A.4: Bloco Pot.SchDoc.

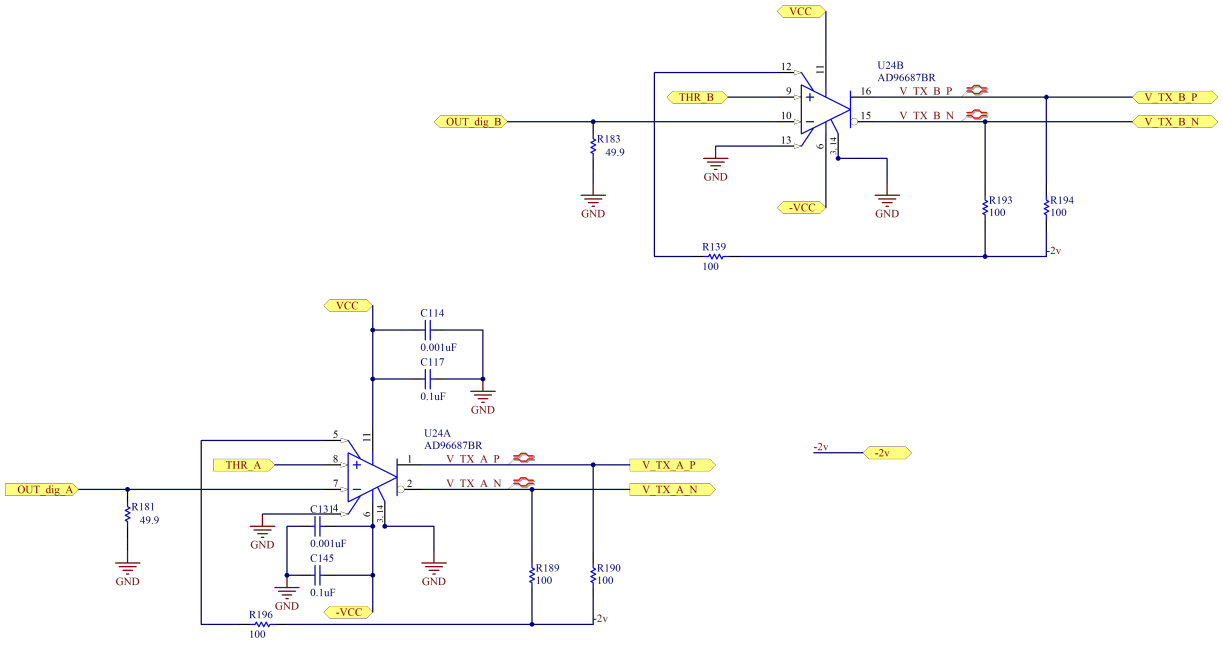


Figura A.5: Bloco Comp_DIG.SchDoc.

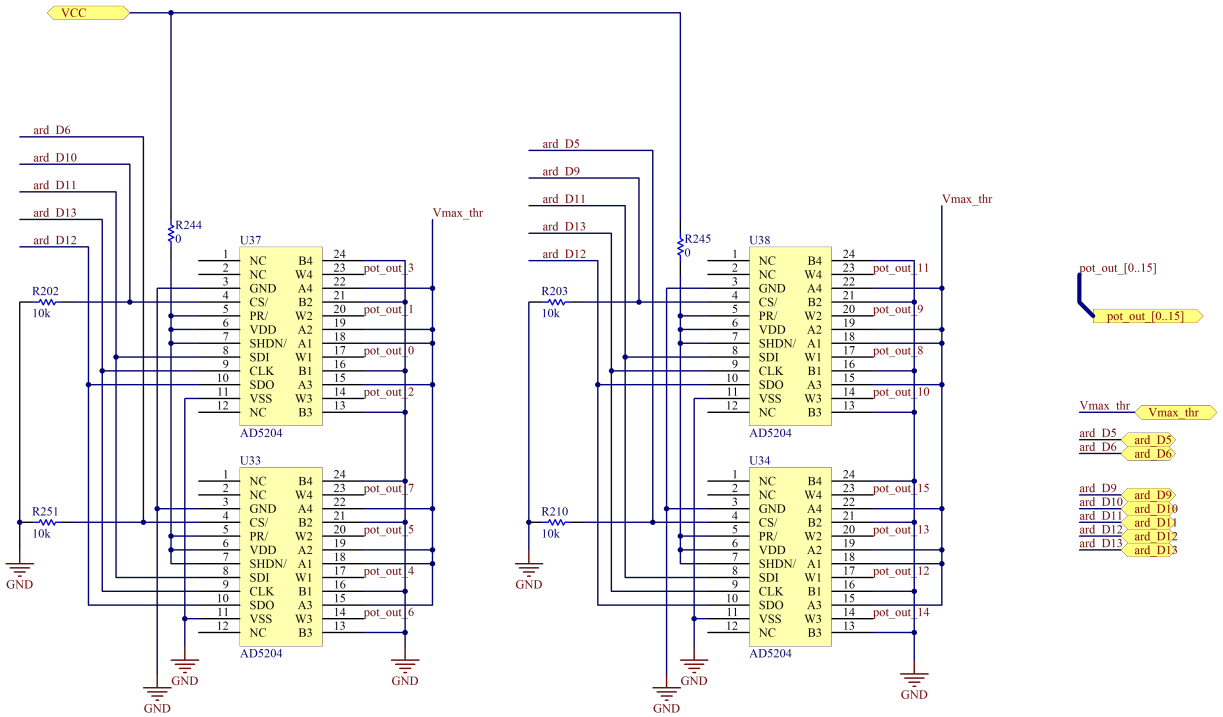


Figura A.6: Bloco pot_dig.SchDoc.

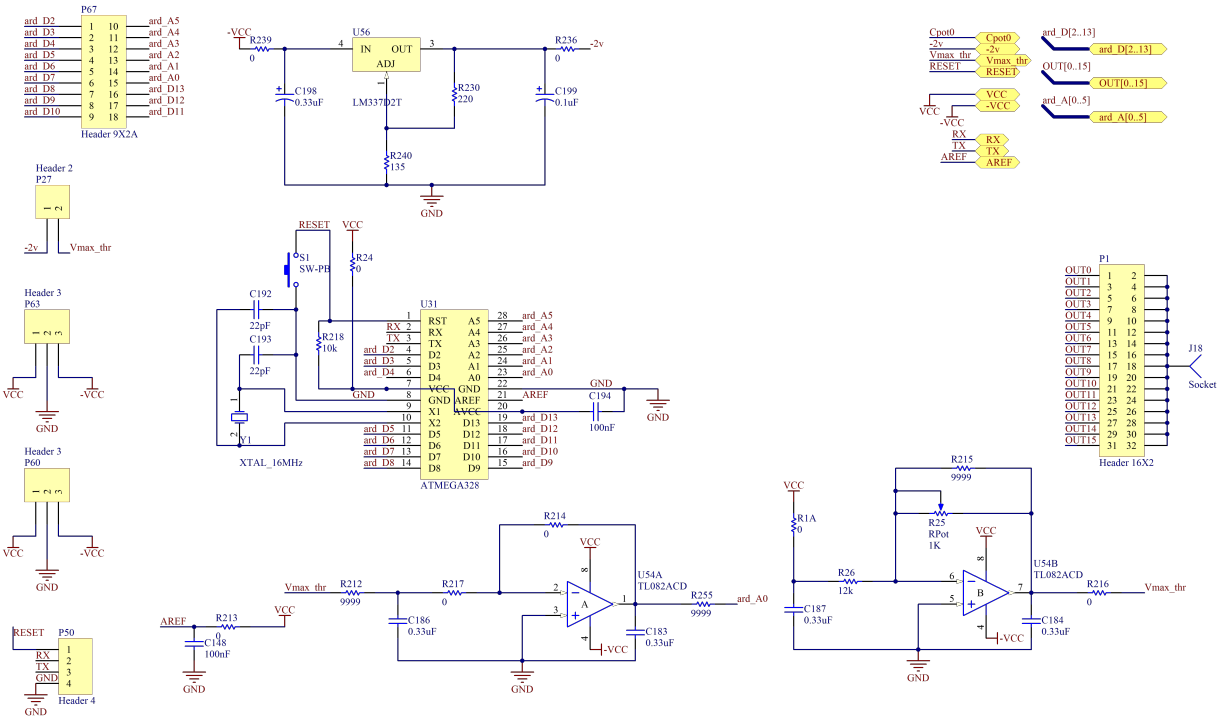


Figura A.7: Bloco AlimentaçãoV2.SchDoc.

Apêndice B

Código de Aquisição de Dados para Calibração dos Potenciômetros Digitais

```
//////////////////////////////////////////////////////////
// Código de aquisição de dados para calibração do potenciômetro digital
// Author: Leonardo Chaves Ruiz Guedes (leocrg@cbpf.br)
//////////////////////////////////////////////////////////


#include <errno.h>      // Error number definitions
#include <stdint.h>     // C99 fixed data types
#include <stdio.h>      // Standard input/output definitions
#include <stdlib.h>     // C standard library
#include <string.h>     // String function definitions
#include <unistd.h>     // UNIX standard function definitions
#include <fcntl.h>      // File control definitions
#include <termios.h>    // POSIX terminal control definitions
#include <stdarg.h>
#include <time.h>
#include <string.h>
#include <math.h>
#include <libgen.h>

////////ARDUINO//////////////
// Abre da porta COM onde o microcontrolador ARDUINO está conectado
```

```
int open_port1(void){

  int fd1;     // File descriptor for the port
  fd1 = open("/dev/ttyUSB0", O_RDWR | O_NOCTTY);
  if (fd1 == -1){
    fprintf(stderr, "open_port: Unable to open ARDUINO %s\n",strerror(errno)
   );
    exit(EXIT_FAILURE);
  }
  return fd1;
}


/////////MULTIMETRO//////////
// Abre da porta COM onde o Multímetro está conectado
int open_port2(void){
  int fd2;     // File descriptor for the port
  fd2 = open("/dev/ttyUSB1", O_RDWR | O_NOCTTY);
  if (fd2 == -1){
    fprintf(stderr, "open_port: Unable to open MULTIMETER %s\n",strerror(
   errno));
    exit(EXIT_FAILURE);
  }
  return fd2;
}

int main(void){
  //#### Settings #####
  int ch=   7;  // Canal de análise
  int Thr_max =     255;  // Valor máximo de Threshold
  int Thr_min =     0;  // Valor mínimo de Threshold
  int Thr_step =  1;  // Step de incremento do valor de Threshold
  //####################

  int s,f;
  int fd1 =   0;      // File
  int fd2 =   0;      // File
  int c=    0;
  struct termios  options;    // Terminal options
  int             rc;
  char tensao[26]="";
```

```
char tensao2[15]="";
char dados[4] = "";
char tensao_ADC[4] = "";
char n_file[30];
int thr = Thr_min;

FILE *pFile;
f = sprintf(n_file, "data_Thr/Teste_linearidade_CH%d.txt", ch);

fd1 = open_port1();             // Abre a porta para leitura e escrita
fd2 = open_port2();             // Abre a porta para leitura e escrita

// Get the current options for the port
if((rc = tcgetattr(fd1, &options)) < 0){
  fprintf(stderr, "failed to get attr: %d, %s\n", fd1, strerror(errno));
  exit(EXIT_FAILURE);
}

if((rc = tcgetattr(fd2, &options)) < 0){
  fprintf(stderr, "failed to get attr: %d, %s\n", fd2, strerror(errno));
  exit(EXIT_FAILURE);
}

// Set the baud rates to 9600
cfsetispeed(&options, B9600);

// Set the baud rates to 9600
cfsetospeed(&options, B9600);

cfmakeraw(&options);
options.c_cflag |= (CLOCAL | CREAD);    // Enable the receiver and set
 local mode
options.c_cflag &= ~CSTOPB;             // 1 stop bit
options.c_cflag &= ~CRTSCTS;            // Disable hardware flow control
options.c_cc[VMIN]  = 1;
options.c_cc[VTIME] = 2;

// Set the new attributes
if((rc = tcsetattr(fd1, TCSANOW, &options)) < 0){
  fprintf(stderr, "failed to set attr: %d, %s\n", fd1, strerror(errno));
  exit(EXIT_FAILURE);
```

```
}

if((rc = tcsetattr(fd2, TCSANOW, &options)) < 0){
  fprintf(stderr, "failed to set attr: %d, %s\n", fd2, strerror(errno));
  exit(EXIT_FAILURE);
}

////////////////////////////////
// Código de leitura e escrita//

pFile = fopen (n_file , "w");

char    ch_v[16] = "abcdefghijklmnop";
char    thrmax[5]="";

if (pFile == NULL) freopen(n_file, "w", pFile);
else
{

  ///////////Comandos iniciais do multímetro//////////////////////////
  write(fd2,"INIT\r\n",6); //Inicialização de comunicação
  sleep(2);
  write(fd2,"SYST:REM\r\n",10); //Inicialização do controle remoto
  sleep(2);
  ////////////////////////////////////////////////////////////////////////



  printf ("Start...\r\n");
  write(fd1,"Q",1);    // comando para informar a tensão máxima de Thr
  sleep(1);
  read(fd1,thrmax,5); // leitura da tensão máxima de Thr
  fprintf (pFile, "THR_MAX:   %s\r\n",thrmax);
  printf ("THR_MAX:   %s\r\n",thrmax);
}

fprintf(pFile, "THRMax  ch  thr tensao\r\n", tensao_ADC, ch, thr, tensao);

while (thr<=Thr_max) {
```

```
    //converte valores de THR no padrão que o uC consiga entender
    if (thr<10) f = sprintf(dados, "%c00%d", ch_v[ch], thr);
    else if (thr<100) f = sprintf(dados, "%c0%d", ch_v[ch], thr);
    else f = sprintf(dados, "%c%d", ch_v[ch], thr);

    write(fd1,dados,4);
    sleep(2);
    write(fd2,"READ?",5); //comando padrão de leitura do Multímetro
    sleep(2);
    write(fd2,"\r\n",2);
    sleep(2);
    read(fd2,tensao,18);
    fprintf(pFile, "CH%d      %d       %s\r\n", ch, thr, tensao);
    printf("%s      %s\r\n",dados,tensao);
    thr = thr + Thr_step;
    sleep(2);
  }

  f = sprintf(dados, "%c%d", ch_v[ch], 255);
  s = strlen (dados);
  write(fd1, dados, s);

  int fclose (FILE *pFile);
  printf("FIM DA AQUISICAO\r\n");


  ////////////////////////////////
  // Fecha os arquivos e finaliza o programa
  close(fd1);
  close(fd2);
  return EXIT_SUCCESS;
}
```

Apêndice C

Código de Análise dos Dados da Calibração dos Potenciômetros Digitais

```
//////////////////////////////////////////////////////////
// Código de análise para calibração do potenciômetro digital
// Author: Leonardo Chaves Ruiz Guedes (leocrg@cbpf.br)
//////////////////////////////////////////////////////////

#include <stdio.h>
#include "TGraph.h"
#include "TGraphErrors.h"
#include "TAxis.h"
#include "TCanvas.h"
#include "TLine.h"
#include <time.h>

Double_t V_P0;
Double_t V_Err_P0;
Double_t V_P1;
Double_t V_Err_P1;
Double_t V_P2;
Double_t V_Err_P2;
Double_t V_P3;
Double_t V_Err_P3;
```

```
void analyse_file(char * namedata) {
  Char_t Data[3];
  Double_t V_bit[10000];
  Double_t V_volt[10000];
  Double_t err_V_volt[10000];
  Int_t nrun=0;

  ifstream in;
  in.open(namedata);

  Char_t header[191];
  in.getline(header,190);
  in.getline(header,190);

  while (1) {
    in >> Data >> V_bit[nrun] >> V_volt[nrun];
    V_volt[nrun]*=-1000;
    err_V_volt[nrun]=1.54;
    if (!in.good()) break;
    nrun++;
  }

  printf(" found %d Entries \n",(nrun));
  in.close();

  TCanvas *c = new TCanvas("graf","graf",200,10,900,1000);
  TGraphErrors *gr = new TGraphErrors(nrun,V_bit,V_volt,0,err_V_volt);

  gStyle->SetOptFit(0111);
  gStyle->SetFuncWidth(4);

  gr->SetTitle("Pos[dec] x Threshold[-mV]");
  gr-> GetXaxis()->SetTitle ("Pos[dec]");
  gr-> GetYaxis()->SetTitle ("Threshold[-mV]");
  gr-> SetMarkerStyle(20) ;
  gr-> SetMarkerColor(4) ;
  gr-> SetMarkerSize(3) ;

  TF1 *f1 = new TF1("f1","pol3",-200,200);

  gr->Fit("f1","Q");
```

```
  gr->Draw("AP");

  V_P0=f1->GetParameter(0);
  V_Err_P0=f1->GetParError(0);
  V_P1=f1->GetParameter(1);
  V_Err_P1=f1->GetParError(1);
  V_P2=f1->GetParameter(2);
  V_Err_P2=f1->GetParError(2);
  V_P3=f1->GetParameter(3);
  V_Err_P3=f1->GetParError(3);

  cout << V_P0 << "\t" << V_Err_P0 << "\t" << V_P1 << "\t" << V_Err_P1 << "\
    t" << V_P2 << "\t" << V_Err_P2 << "\t" << V_P3 << "\t" << V_Err_P3 <<
    endl;

}
```

Apêndice D

Código de Caracterização da *Front-End* de 16 Canais

```
//////////////////////////////////////////////////////////
// Código de Caracterização da Front-End
// Author: Leonardo Chaves Ruiz Guedes (leocrg@cbpf.br)
//////////////////////////////////////////////////////////


#include <stdio.h>
#include "TGraph.h"
#include "TAxis.h"
#include "TCanvas.h"
#include "TLine.h"
#include <time.h>

Float_t Print_v_mean[16][1000];
Float_t Print_v_sigma[16][1000];
Float_t Print_erro_mean[16][1000];
Float_t Print_erro_sigma[16][1000];
Float_t Print_Gain[16];
Float_t Print_offset[16];
Float_t Print_erro_Gain[16];
Float_t Print_erro_offset[16];
Double_t V_P0 = 6.95319;
Double_t V_Err_P0 = 0.0817453;
Double_t V_P1 = 1.61414;
```

```
Double_t V_Err_P1 = 0.00278165;
Double_t V_P2 = -0.00190676;
Double_t V_Err_P2 = 2.53766e-05;
Double_t V_P3 = 2.87088e-06;
Double_t V_Err_P3 = 6.54092e-08;

Int_t c=0;

Double_t funGauss(Double_t *x, Double_t *par)
{
  Double_t G;
  Double_t cte = 2.5066297;
  Double_t N, Function;
  N = 1.0/(cte*par[2]);
  Function = N*(TMath::Exp(- (par[1]-x[0]) * (par[1] - x[0]) / (2.0 * par[2]
    * par[2]) ) );
  G = par[0]*Function;
  return G;
}

Double_t SCurve(Double_t *x, Double_t *par){
  Function = par[0]*(0.5-0.5*TMath::Erf((x[0]-par[1])/(1.4142*par[2])));
  return Function;
}

void loop_makeTree(char* cnamedata, TString Channels_gr = "0123456789ABCDEF"
   ){
  TString String_ch = "0123456789ABCDEF";
  Int_t ii=0;
  Int_t auxi=0;
  for (Int_t i=0;i<16;i=i+2) {
    if (String_ch[i] == Channels_gr[ii]){
      char* namedata = Form("%s%i_%i.dat",cnamedata, i,(i+1));
      TString SChannels_gr(Channels_gr(ii,2));
      cout << "makeTree(" << namedata << "," << SChannels_gr.Data() << ");"
   <<endl;
      makeTree(namedata,SChannels_gr.Data());
      Int_t ii=ii+2;
    }
  }
}
```

```
void makeTree(char * namedata, char*  Channels_gr = "0123456789ABCDEF") {
  char* nametree = Form("tree_%s.root",Channels_gr);
  ifstream in;
  in.open(namedata);
  const int nGroups = 16 ;
  Float_t vcont[nGroups][1000] ;
  Int_t v_N_ev;
  Float_t v_Amp[1000];
  Float_t v_T_aq[1000];
  Float_t v_vThr[1000];
  Int_t n = 0;
  Int_t nrun = 0;
  TFile *f = new TFile(nametree,"RECREATE");
  Char_t evtId[10];
  Int_t dt;
  Char_t dt_string[50] ;
  Char_t Data[20];
  Int_t N_ev;
  Float_t Amp;
  Int_t T_aq;
  Float_t vThr, counter;
  Int_t cont_0, cont_1, cont_2, cont_3, cont_4, cont_5;
  Int_t cont_6, cont_7, cont_8, cont_9, cont_10, cont_11;
  Int_t cont_12, cont_13, cont_14, cont_15;

  TTree creat("creat","creat");
  creat.Branch("cont_0",&cont_0,"cont_0/I");
  creat.Branch("cont_1",&cont_1,"cont_1/I");
  creat.Branch("cont_2",&cont_2,"cont_2/I");
  creat.Branch("cont_3",&cont_3,"cont_3/I");
  creat.Branch("cont_4",&cont_4,"cont_4/I");
  creat.Branch("cont_5",&cont_5,"cont_5/I");
  creat.Branch("cont_6",&cont_6,"cont_6/I");
  creat.Branch("cont_7",&cont_7,"cont_7/I");
  creat.Branch("cont_8",&cont_8,"cont_8/I");
  creat.Branch("cont_9",&cont_9,"cont_9/I");
  creat.Branch("cont_10",&cont_10,"cont_10/I");
  creat.Branch("cont_11",&cont_11,"cont_11/I");
  creat.Branch("cont_12",&cont_12,"cont_12/I");
  creat.Branch("cont_13",&cont_13,"cont_13/I");
```

```
  creat.Branch("cont_14",&cont_14,"cont_14/I");
  creat.Branch("cont_15",&cont_15,"cont_15/I");
  creat.Branch("Data",&Data,"Data/C");
  creat.Branch("evtId",&evtId,"evtId/C");
  creat.Branch("N_ev",&N_ev,"N_ev/I");
  creat.Branch("Amp",&Amp,"Amp/F");
  creat.Branch("T_aq",&T_aq,"T_aq/I");
  creat.Branch("nrun",&nrun,"nrun/I");
  creat.Branch("vth",&vThr,"vth/F");

  if(nrun == 0) {
    Char_t header[191];
    in.getline(header,190);
    nrun++;
  }

  while (1) {
    in >> Data >> evtId >> N_ev >> Amp >> vThr >> T_aq >>
    cont_0 >> cont_1 >> cont_2 >> cont_3 >> cont_4 >> cont_5 >> cont_6 >>
    cont_7 >> cont_8 >> cont_9 >> cont_10 >> cont_11 >> cont_12 >> cont_13
   >>
    cont_14 >> cont_15;

    if (!in.good()) break;
    if (nrun < 30) printf("%d: %4f -- %d, %d, %d, %d , %d , %d , %d , %
   d , %d , %d , %d , %d , %d , %d, %d \n",
    nrun,vThr,cont_0, cont_1, cont_2, cont_3, cont_4, cont_5, cont_6, cont_7
    , cont_8, cont_9, cont_10, cont_11, cont_12, cont_13, cont_14, cont_15);

    creat.Fill();

    nrun++;
  }

  printf(" found %d Entries \n",(nrun-1));
  in.close();
  f->Write();
}

void loop_analyse_tree(Float_t CUT_Min = 0 , Float_t CUT_Max = 255, char*
    Thr_scale = "b", TString Channels_gr = "0123456789ABCDEF", Float_t
```

```
    freq_base=10000){
  Int_t i=0;
  Int_t a=0;
  while (i < strlen(Channels_gr)){
    TString SChannels_gr(Channels_gr(i,2));
    char* nametree = Form("tree_%s.root",SChannels_gr.Data());
    analyse_tree( nametree, Thr_scale , SChannels_gr.Data(), CUT_Min,
   CUT_Max,freq_base);
    i=i+2;
  }

  cout << "CANAL\tmean\tsigma\tmean\tsigma\tmean\tsigma\tGain\toffset" <<
    endl;
  for (i=0;i<16;i++) {
    cout << i << "\t";
    for (a=0;a<3;a++) {
      cout << Print_v_mean[i][a] << " +- "  << Print_erro_mean[i][a] << "\t"
    << Print_v_sigma[i][a]  << " +- "  << Print_erro_sigma[i][a] <<"\t" ;
    }

    cout << Print_Gain[i] << " +- "  << Print_erro_Gain[i] << "\t" <<
    Print_offset[i] << " +- "  << Print_erro_offset[i] << endl;//<< sqrt(pow(
    Print_erro_offset[i],2)+pow(V_Err_P0,2)) << endl;
  }
}

void analyse_tree(char* nametree, char* Thr_scale = "b", char* Channels_gr =
    "0123456789ABCDEF", Float_t CUT_Min = 0 , Float_t CUT_Max = 255, Float_t
    freq_base=10000)
{
  TFile *file = new TFile(nametree,"UPDATE");
  TTree *j = (TTree*)file->Get("creat");
  TString String_ch = "0123456789ABCDEF";
  const int nGroups = 16 ;
  Float_t vcont[nGroups][1000][1000];
  Int_t v_N_ev;
  Float_t v_Amp[1000];
  Float_t T_Amp[1000];
  Float_t v_T_aq[1000];
  Float_t v_vThr[1000][1000];
  Float_t err_v_vThr[1000][1000];
```

```
Int_t n = 0;
Char_t evtId[10];
Char_t dt_string[50] ;
Char_t Data[20];
Int_t N_ev;
Float_t Amp;
Int_t T_aq;
Float_t vThr;
Int_t cont_0, cont_1, cont_2, cont_3, cont_4, cont_5;
Int_t cont_6, cont_7, cont_8, cont_9, cont_10, cont_11;
Int_t cont_12, cont_13, cont_14, cont_15;

j->SetBranchAddress("cont_0",&cont_0);
j->SetBranchAddress("cont_1",&cont_1);
j->SetBranchAddress("cont_2",&cont_2);
j->SetBranchAddress("cont_3",&cont_3);
j->SetBranchAddress("cont_4",&cont_4);
j->SetBranchAddress("cont_5",&cont_5);
j->SetBranchAddress("cont_6",&cont_6);
j->SetBranchAddress("cont_7",&cont_7);
j->SetBranchAddress("cont_8",&cont_8);
j->SetBranchAddress("cont_9",&cont_9);
j->SetBranchAddress("cont_10",&cont_10);
j->SetBranchAddress("cont_11",&cont_11);
j->SetBranchAddress("cont_12",&cont_12);
j->SetBranchAddress("cont_13",&cont_13);
j->SetBranchAddress("cont_14",&cont_14);
j->SetBranchAddress("cont_15",&cont_15);
j->SetBranchAddress("Data",&Data);
j->SetBranchAddress("N_ev",&N_ev);
j->SetBranchAddress("Amp",&Amp);
j->SetBranchAddress("T_aq",&T_aq);
j->SetBranchAddress("vth",&vThr);
j->SetBranchAddress("vth",&vThr);
Int_t a=0;
Int_t nentries = (Int_t)j->GetEntries();
Int_t T_nentries[1000];
Int_t ii=0;
for (Int_t i=0;i<nentries;i++) {
  j->GetEntry(i);
```

```
ii++;
v_Amp[i]=Amp;
if (i>0){
  if (v_Amp[i]!=v_Amp[i-1]){
    a++;
    ii=1;
  }
}

T_nentries[a]=ii;

if ( Thr_scale[0] == 'b') {
  v_vThr[a][ii-1]=vThr;
}
else {
  v_vThr[a][ii-1]=V_P3*pow(vThr,3) + V_P2*pow(vThr,2) + V_P1*vThr + V_P0
;
  err_v_vThr[a][ii-1]=sqrt(pow(V_Err_P3*pow(vThr,3),2)+pow(V_Err_P2*pow(
vThr,2),2)+pow(V_Err_P1*vThr,2)+pow(V_Err_P0,2));
  cout << "debug valor" << v_vThr[a][ii-1] << " err= " << err_v_vThr[a][
ii-1] << endl;
}
v_T_aq[ii-1]=T_aq;
v_N_ev[ii-1]=N_ev;
T_Amp[a]=Amp*(-1); //valores positivos
vcont[a][0][ii-1]= cont_0;
vcont[a][1][ii-1]= cont_1;
vcont[a][2][ii-1]= cont_2;
vcont[a][3][ii-1]= cont_3;
vcont[a][4][ii-1]= cont_4;
vcont[a][5][ii-1]= cont_5;
vcont[a][6][ii-1]= cont_6;
vcont[a][7][ii-1]= cont_7;
vcont[a][8][ii-1]= cont_8;
vcont[a][9][ii-1]= cont_9;
vcont[a][10][ii-1]= cont_10;
vcont[a][11][ii-1]= cont_11;
vcont[a][12][ii-1]= cont_12;
vcont[a][13][ii-1]= cont_13;
vcont[a][14][ii-1]= cont_14;
vcont[a][15][ii-1]= cont_15;
```

```
}

Float_t T_vcont[nGroups][1000][1000];
Float_t v_mean[1000];
Float_t v_sigma[1000];
Float_t erro_mean[1000];
Float_t erro_sigma[1000];
Float_t v_sigma_med[1000];
Float_t T_vThr[1000][1000];
Float_t err_T_vThr[1000][1000];
Float_t Err_max[1000][nGroups][1000];
Float_t Err_min[1000][nGroups][1000];
Float_t Err_med[1000][nGroups][1000];
Float_t T_tan[1000][nGroups][1000];
Float_t T_vThr_med[1000][1000];
Float_t Thr_Min_curve[1000][nGroups];
Int_t k[1000];
Int_t Q_cont[1000];
Int_t n_amp = a+1;
Int_t x=0;

for (Int_t a=0; a<n_amp; a++){

  for (Int_t ii=0;ii<16;ii++){
    T_vcont[a][ii][0]=-1;
    Q_cont[a] = 1;
    k[a]=0;
    Err_max[a][ii][0]=vcont[a][ii][0];
    Err_min[a][ii][0]=vcont[a][ii][0];
    for (Int_t i=1;i<T_nentries[a];i++){
      if ((v_vThr[a][i-1]>CUT_Min)&&(v_vThr[a][i-1]<=CUT_Max)){
        if (T_vcont[a][ii][0] == -1) {
          T_vcont[a][ii][0]=vcont[a][ii][i-1];
        }
        if (v_vThr[a][i]==v_vThr[a][i-1]){
          Q_cont[a]=Q_cont[a]+1;
          T_vcont[a][ii][k[a]] = T_vcont[a][ii][k[a]] + vcont[a][ii][i];
          if (vcont[a][ii][i]>vcont[a][ii][i-1]) {
            Err_max[a][ii][k[a]]=vcont[a][ii][i];
          }
          if (vcont[a][ii][i]<vcont[a][ii][i-1]) {
```

```
            Err_min[a][ii][k[a]]=vcont[a][ii][i];
          }
        }
        else if ((v_vThr[a][i]!=v_vThr[a][i-1]) || (i==T_nentries[a]-1)){
 //nentries-1)){
          T_vcont[a][ii][k[a]] = T_vcont[a][ii][k[a]] / Q_cont[a];
          T_vThr[a][k[a]]=v_vThr[a][i-1];
          err_T_vThr[a][k[a]]=err_v_vThr[a][i-1];
          Err_med[a][ii][k[a]]=(Err_max[a][ii][k[a]]-Err_min[a][ii][k[a]])
 /2;
          Q_cont[a]=1;
          k[a]=k[a]+1;
          T_vcont[a][ii][k[a]]=vcont[a][ii][i];
        }
      }
    }
  }
}


for (Int_t a=0; a<n_amp; a++){
  for (Int_t ii=0;ii<16;ii++){
    for (Int_t i=1; i<k[a]; i++){
      T_tan[a][ii][i-1]= abs(T_vcont[a][ii][i] -T_vcont[a][ii][i-1])/abs(
 T_vThr[a][i]-T_vThr[a][i-1]);
      T_vThr_med[a][i-1]=T_vThr[a][i-1] + (T_vThr[a][i]-T_vThr[a][i-1])/2;
    }
  }
}

TCanvas *c[16];
TGraphErrors *gr[16];
TGraph *gr_Tan[16];
TGraphErrors *Amp_mean[1000];
TGraphErrors *Amp_sigma[1000];
TVirtualPad * c11[16];
TVirtualPad * c1[16][1000];
TVirtualPad * c21[16];
TVirtualPad * c2[16];
TVirtualPad * c3[16];
TF1* f1;
```

```
Int_t  ii=0;
Int_t  iii=0;

for (Int_t  i=0;i<16;i++){
  if (String_ch[i] == Channels_gr[ii]){
    ii++;
    c[i] = new TCanvas(Form("c%d",i),Form("Channel%d: ",i)
 ,200,10,900,1000);
    if (n_amp>1) c[i]->Divide(2,1);
    else c[i]->Divide(1,1);
    c11[i] = c[i]->cd(1);
    c11[i]->Divide(1,n_amp);
    x=1;
    for (Int_t a=0; a<n_amp; a++){
      c1[i][a] = c11[i]->cd(x);
      gr_Tan[i] = new TGraph(k[a]-1,T_vThr_med[a],T_tan[a][i]);
      gr_Tan[i]->SetTitle(Form("Amplitude_injetada = -%fmV ",T_Amp[a])) ;
      gr_Tan[i]-> GetXaxis()->SetTitle ("Vthr") ;
      gr_Tan[i]-> GetXaxis()->SetLimits(CUT_Min , CUT_Max ) ;
      gr_Tan[i]-> GetYaxis()->SetTitle ("Deriv") ;
      gr_Tan[i]-> SetMarkerStyle(20) ;
      gr_Tan[i]-> SetMarkerSize(1) ;
      gr_Tan[i]-> SetMinimum (0) ;
      f2 = new TF1("f2", "gaus",CUT_Min, T_vThr_med[a][k[a]] );
      gStyle->SetOptFit(0111);
      gr_Tan[i]->Fit("f2", "Q");
      gr[i] = new TGraphErrors(k[a],T_vThr[a],T_vcont[a][i],err_T_vThr[a
 ],0);
      gStyle->SetOptFit(0111);
      gr[i]->SetTitle(Form("Amplitude_injetada = -%.2fmV ",T_Amp[a])) ;
      if ( Thr_scale[0] == 'b')   gr[i]-> GetXaxis()->SetTitle ("Vthr[bit]
 ");
      else gr[i]-> GetXaxis()->SetTitle ("Vthr[-mV]");
      gr[i]-> GetXaxis()->SetLimits(CUT_Min , CUT_Max) ;
      gr[i]-> GetYaxis()->SetTitle ("Cont") ;
      gr[i]-> SetMarkerStyle(20) ;
      gr[i]-> SetMarkerSize(1) ;
      gr[i]-> SetMinimum (0) ;

      TF1 *f3 = new TF1("f3",SCurve,CUT_Min,CUT_Max,3);
      f3->FixParameter(0,freq_base);
```

```
      f3->SetParameter(1,f2->GetParameter(1));
      f3->SetParameter(2,f2->GetParameter(2));

      gr[i]->Fit("f3","WQ");
      gr[i]->Draw("AP");

      v_mean[a]=f3->GetParameter(1);
      Print_v_mean[i][a]=f3->GetParameter(1);
      erro_mean[a]=f3->GetParError(1);
      Print_erro_mean[i][a]=f3->GetParError(1);
      v_sigma[a]=f3->GetParameter(2);
      Print_v_sigma[i][a]=f3->GetParameter(2);
      erro_sigma[a]=f3->GetParError(2);
      Print_erro_sigma[i][a]=f3->GetParError(2);
      x++;
    }

    if (n_amp>1){
      c21[i] = c[i]->cd(2);
      c21[i]->Divide(1,2);
      Amp_mean[i] = new TGraphErrors(n_amp,T_Amp,v_mean,0,erro_mean);
      Amp_sigma[i] = new TGraphErrors(n_amp,T_Amp,v_sigma,0,erro_sigma);
      gStyle->SetOptFit(0111);
      Amp_mean[i]->SetTitle("Vth@Eff_50% vs Amplitude");
      if (Thr_scale[0] == 'b')   Amp_mean[i]-> GetYaxis()->SetTitle("
Vth@Eff_50%(bit)");
      else       Amp_mean[i]-> GetYaxis()->SetTitle("Vth@Eff_50%(-mV)");
      Amp_mean[i]-> GetXaxis()->SetTitle("Amp(-mV)");
      Amp_mean[i]-> SetMinimum(0);
      Amp_mean[i]-> SetMarkerStyle(20);
      Amp_mean[i]-> SetMarkerSize(1);
      gStyle->SetOptFit(0111);
      Amp_sigma[i]->SetTitle("ENV vs Amplitute");

      if (Thr_scale[0] == 'b')   Amp_sigma[i]-> GetYaxis()->SetTitle("
ENV(bit)");
      else       Amp_sigma[i]-> GetYaxis()->SetTitle("ENV(mV)");

      Amp_sigma[i]-> GetXaxis()->SetTitle("Amp(-mV)");
      Amp_sigma[i]-> SetMinimum(0);
      Amp_sigma[i]-> SetMarkerStyle(20);
```

```
            Amp_sigma[i]-> SetMarkerSize(1) ;
            gStyle->SetOptFit(0111);
            TF1 *f4 = new TF1("f4","pol1",-200,200);
            TF1 *f5 = new TF1("f5","pol0",-200,200);

            c2[i] = c21[i]->cd(1);
            Amp_mean[i]->Fit("f4","Q");
            Amp_mean[i]->Draw("AP");

            Print_Gain[i]=f4->GetParameter(1);
            Print_erro_Gain[i]=f4->GetParError(1);
            Print_offset[i]=f4->GetParameter(0);
            Print_erro_offset[i]=f4->GetParError(0);

            c3[i] = c21[i]->cd(2);
            Amp_sigma[i]->Fit("f5","Q");
            Amp_sigma[i]->Draw("AP");
            v_sigma_med[iii]= f5->GetParameter(0);
            iii++;
          }
          c[i]->SaveAs(Form("Characterization_CH%d.pdf",i));
        }
      }
    }

void loop_analyse_tree_gauss(Float_t CUT_Min = 0 , Float_t CUT_Max = 255,
    char* Thr_scale = "b", TString Channels_gr = "0123456789ABCDEF", Float_t
    freq_base=10000){
  Int_t i=0;
  while (i < strlen(Channels_gr)){
    TString SChannels_gr(Channels_gr(i,2));
    char* nametree = Form("tree_%s.root",SChannels_gr.Data());
    analyse_tree_gauss( nametree, Thr_scale , SChannels_gr.Data(), CUT_Min,
   CUT_Max,freq_base);
    i=i+2;
  }
}

void analyse_tree_gauss(char* nametree, char* Thr_scale = "b", char*
    Channels_gr = "0123456789ABCDEF", Float_t CUT_Min = 0 , Float_t CUT_Max =
     255, Float_t freq_base=10000)
```

```
{

  TFile *file = new TFile(nametree,"UPDATE");
  TTree *j = (TTree*)file->Get("creat");

  TString String_ch = "0123456789ABCDEF";
  const int nGroups = 16 ;
  Float_t vcont[nGroups][1000][1000];
  Int_t v_N_ev;
  Float_t v_Amp[1000];
  Float_t T_Amp[1000];
  Float_t v_T_aq[1000];
  Float_t v_vThr[1000][1000];
  Int_t n = 0;
  Char_t evtId[10];
  Char_t dt_string[50] ;
  Char_t Data[20];
  Int_t N_ev;
  Float_t Amp;
  Int_t T_aq;
  Float_t vThr;
  Int_t cont_0, cont_1, cont_2, cont_3, cont_4, cont_5;
  Int_t cont_6, cont_7, cont_8, cont_9, cont_10, cont_11;
  Int_t cont_12, cont_13, cont_14, cont_15;

  j->SetBranchAddress("cont_0",&cont_0);
  j->SetBranchAddress("cont_1",&cont_1);
  j->SetBranchAddress("cont_2",&cont_2);
  j->SetBranchAddress("cont_3",&cont_3);
  j->SetBranchAddress("cont_4",&cont_4);
  j->SetBranchAddress("cont_5",&cont_5);
  j->SetBranchAddress("cont_6",&cont_6);
  j->SetBranchAddress("cont_7",&cont_7);
  j->SetBranchAddress("cont_8",&cont_8);
  j->SetBranchAddress("cont_9",&cont_9);
  j->SetBranchAddress("cont_10",&cont_10);
  j->SetBranchAddress("cont_11",&cont_11);
  j->SetBranchAddress("cont_12",&cont_12);
  j->SetBranchAddress("cont_13",&cont_13);
  j->SetBranchAddress("cont_14",&cont_14);
  j->SetBranchAddress("cont_15",&cont_15);
```

```
j->SetBranchAddress("Data",&Data);
j->SetBranchAddress("N_ev",&N_ev);
j->SetBranchAddress("Amp",&Amp);
j->SetBranchAddress("T_aq",&T_aq);
j->SetBranchAddress("vth",&vThr);

j->SetBranchAddress("vth",&vThr);
Int_t a=0;
Int_t nentries = (Int_t)j->GetEntries();
Int_t T_nentries[1000];
Int_t ii=0;
for (Int_t i=0;i<nentries;i++) {
  j->GetEntry(i);

  ii++;
  v_Amp[i]=Amp;
  if (i>0){
    if (v_Amp[i]!=v_Amp[i-1]){
      a++;
      ii=1;
    }
  }
  T_nentries[a]=ii;

  if ( Thr_scale[0] == 'b') {
    v_vThr[a][ii-1]=vThr;
  }
  else v_vThr[a][ii-1]=(0.000000004*vThr*vThr*vThr - 0.000003*vThr*vThr +
 0.0017*vThr + 0.0529)*1000;

  v_T_aq[ii-1]=T_aq;
  v_N_ev[ii-1]=N_ev;
  T_Amp[a]=Amp*(-1); //valores positivos
  vcont[a][0][ii-1]= cont_0;
  vcont[a][1][ii-1]= cont_1;
  vcont[a][2][ii-1]= cont_2;
  vcont[a][3][ii-1]= cont_3;
  vcont[a][4][ii-1]= cont_4;
  vcont[a][5][ii-1]= cont_5;
  vcont[a][6][ii-1]= cont_6;
  vcont[a][7][ii-1]= cont_7;
```

```
  vcont[a][8][ii-1]= cont_8;
  vcont[a][9][ii-1]= cont_9;
  vcont[a][10][ii-1]= cont_10;
  vcont[a][11][ii-1]= cont_11;
  vcont[a][12][ii-1]= cont_12;
  vcont[a][13][ii-1]= cont_13;
  vcont[a][14][ii-1]= cont_14;
  vcont[a][15][ii-1]= cont_15;
}

Float_t T_vcont[nGroups][1000][1000];
Float_t v_mean[1000];
Float_t v_sigma[1000];
Float_t erro_mean[1000];
Float_t erro_sigma[1000];
Float_t v_sigma_med[1000];
Float_t T_vThr[1000][1000];
Float_t Err_max[1000][nGroups][1000];
Float_t Err_min[1000][nGroups][1000];
Float_t Err_med[1000][nGroups][1000];
Float_t T_tan[1000][nGroups][1000];
Float_t T_vThr_med[1000][1000];
Float_t Thr_Min_curve[1000][nGroups];
Int_t k[1000];
Int_t Q_cont[1000];
Int_t n_amp = a+1;
Int_t x=0;

for (Int_t a=0; a<n_amp; a++){
  for (Int_t ii=0;ii<16;ii++){
    T_vcont[a][ii][0]=-1;
    Q_cont[a] = 1;
    k[a]=0;
    Err_max[a][ii][0]=vcont[a][ii][0];
    Err_min[a][ii][0]=vcont[a][ii][0];
    for (Int_t i=1;i<T_nentries[a];i++){
      if ((v_vThr[a][i-1]>CUT_Min)&&(v_vThr[a][i-1]<=CUT_Max)){
        if (T_vcont[a][ii][0] == -1) {
          T_vcont[a][ii][0]=vcont[a][ii][i-1];
        }
        if (v_vThr[a][i]==v_vThr[a][i-1]){
```

```
            Q_cont[a]=Q_cont[a]+1;
            T_vcont[a][ii][k[a]] = T_vcont[a][ii][k[a]] + vcont[a][ii][i];
            if (vcont[a][ii][i]>vcont[a][ii][i-1]) {
              Err_max[a][ii][k[a]]=vcont[a][ii][i];
            }
            if (vcont[a][ii][i]<vcont[a][ii][i-1]) {
              Err_min[a][ii][k[a]]=vcont[a][ii][i];
            }
          }
          else if ((v_vThr[a][i]!=v_vThr[a][i-1]) || (i==T_nentries[a]-1)){
 //nentries-1)){
            T_vcont[a][ii][k[a]] = T_vcont[a][ii][k[a]] / Q_cont[a];
            T_vThr[a][k[a]]=v_vThr[a][i-1];
            Err_med[a][ii][k[a]]=(Err_max[a][ii][k[a]]-Err_min[a][ii][k[a]])
 /2;
            Q_cont[a]=1;
            k[a]=k[a]+1;
            T_vcont[a][ii][k[a]]=vcont[a][ii][i];
          }
        }
      }
    }
  }

for (Int_t a=0; a<n_amp; a++){
  for (Int_t ii=0;ii<16;ii++){
    for (Int_t i=1; i<k[a]; i++){
      T_tan[a][ii][i-1]= abs(T_vcont[a][ii][i] -T_vcont[a][ii][i-1])/abs(
 T_vThr[a][i]-T_vThr[a][i-1]);
      T_vThr_med[a][i-1]=T_vThr[a][i-1] + (T_vThr[a][i]-T_vThr[a][i-1])/2;
    }
  }
}

TCanvas *c[16];
TGraph *gr[16];
TGraph *gr_Tan[16];
TGraphErrors *Amp_mean[1000];
TGraphErrors *Amp_sigma[1000];
TVirtualPad * c11[16];
TVirtualPad * c1[16][1000];
```

```
TVirtualPad * c21[16];
TVirtualPad * c2[16];
TVirtualPad * c3[16];

TF1* f1;
Int_t ii=0;
Int_t iii=0;

for (Int_t i=0;i<16;i++){
  if (String_ch[i] == Channels_gr[ii]){
    ii++;
    c[i] = new TCanvas(Form("c%d",i),Form("Channel%d: ",i)
 ,200,10,900,1000);
    if (n_amp>1) c[i]->Divide(2,1);
    else c[i]->Divide(1,1);
    c11[i] = c[i]->cd(1);
    c11[i]->Divide(1,n_amp);
    x=1;
    for (Int_t a=0; a<n_amp; a++){
      c1[i][a] = c11[i]->cd(x);
      gr_Tan[i] = new TGraph(k[a]-1,T_vThr_med[a],T_tan[a][i]);
      gr_Tan[i]->SetTitle(Form("Amplitude_injetada = -%fmV ",T_Amp[a])) ;
      gr_Tan[i]-> GetXaxis()->SetTitle ("Vthr") ;
      gr_Tan[i]-> GetXaxis()->SetLimits(CUT_Min , CUT_Max ) ;
      gr_Tan[i]-> GetYaxis()->SetTitle ("Deriv") ;
      gr_Tan[i]-> SetMarkerStyle(20) ;
      gr_Tan[i]-> SetMarkerSize(1) ;
      gr_Tan[i]-> SetMinimum (0) ;
      f2 = new TF1("f2", "gaus",CUT_Min, T_vThr_med[a][k[a]-2] );
      gStyle->SetOptFit(0111);
      gr_Tan[i]->Fit("f2", "Q");
      cout << " mean = " << f2->GetParameter(1) << endl;
      cout << " sigma = " << f2->GetParameter(2) << endl;
      gr[i] = new TGraph(k[a],T_vThr[a],T_vcont[a][i]);
      gStyle->SetOptFit(0111);
      gr[i]->SetTitle(Form("Amplitude_injetada = -%.2fmV ",T_Amp[a])) ;
      if ( Thr_scale[0] == 'b')   gr[i]-> GetXaxis()->SetTitle ("Vthr[bit]
 ");
      else       gr[i]-> GetXaxis()->SetTitle ("Vthr[-mV]");
      gr[i]-> GetXaxis()->SetLimits(CUT_Min , CUT_Max) ;
      gr[i]-> GetYaxis()->SetTitle ("Cont") ;
```

```
      gr[i]-> SetMarkerStyle(20) ;
      gr[i]-> SetMarkerSize(1) ;
      gr[i]-> SetMinimum (0) ;

      TF1 *f3 = new TF1("f3",SCurve,CUT_Min,CUT_Max,3);
      f3->FixParameter(0,freq_base);
      f3->SetParameter(1,f2->GetParameter(1));
      f3->SetParameter(2,f2->GetParameter(2));
      cout << " mean = " << f2->GetParameter(1)  << endl;
      cout << " sigma = " << f2->GetParameter(2)  << endl;
      cout << " mean erro= " << f2->GetParError(1)  << endl;
      cout << " sigma erro= " << f2->GetParError(2)  << endl;

      gr[i]->Fit("f3","Q");
      gr_tan[i]->Draw("AP");

      v_mean[a]=f3->GetParameter(1);
      erro_mean[a]=f3->GetParError(1);
      v_sigma[a]=f3->GetParameter(2);
      erro_sigma[a]=f3->GetParError(1);
      x++;
    }

    if (n_amp>1){
      c21[i] = c[i]->cd(2);
      c21[i]->Divide(1,2);

      Amp_mean[i] = new TGraphErrors(n_amp,T_Amp,v_mean,0,erro_mean);
      Amp_sigma[i] = new TGraphErrors(n_amp,T_Amp,v_sigma,0,erro_sigma);
      gStyle->SetOptFit(0111);

      Amp_mean[i]->SetTitle("Vth@Eff_50% vs Amplitute") ;

      if ( Thr_scale[0] == 'b')    Amp_mean[i]-> GetYaxis()->SetTitle ("
Vth@Eff_50%(bit)");
      else        Amp_mean[i]-> GetYaxis()->SetTitle ("Vth@Eff_50%(-mV)");

      Amp_mean[i]-> GetXaxis()->SetTitle ("Amp(-mV)") ;
      Amp_mean[i]-> SetMinimum (0) ;
      Amp_mean[i]-> SetMarkerStyle(20) ;
      Amp_mean[i]-> SetMarkerSize(1) ;
```

```
        gStyle->SetOptFit(0111);
        Amp_sigma[i]->SetTitle("ENV vs Amplitute") ;

        if ( Thr_scale[0] == 'b')    Amp_sigma[i]-> GetYaxis()->SetTitle ("
    ENV(bit)") ;
        else        Amp_sigma[i]-> GetYaxis()->SetTitle ("ENV(mV)") ;

        Amp_sigma[i]-> GetXaxis()->SetTitle ("Amp(-mV)") ;
        Amp_sigma[i]-> SetMinimum (0) ;
        Amp_sigma[i]-> SetMarkerStyle(20) ;
        Amp_sigma[i]-> SetMarkerSize(1) ;
        gStyle->SetOptFit(0111);
        TF1 *f4 = new TF1("f4","pol1",-200,200);
        TF1 *f5 = new TF1("f5","pol0",-200,200);

        c2[i] = c21[i]->cd(1);
        Amp_mean[i]->Fit("f4","Q");
        Amp_mean[i]->Draw("AP");

        c3[i] = c21[i]->cd(2);
        Amp_sigma[i]->Fit("f5","Q");
        Amp_sigma[i]->Draw("");
        v_sigma_med[iii]= f5->GetParameter(0);
        iii++;
      }
      c[i]->SaveAs(Form("Characterization_CH%d.pdf",i));
    }
  }

  cout << "CH Sigma" << endl;
  for (Int_t i=0;i<16;i++){
    cout << "CH" << i << "   " << v_sigma_med[i] << endl;
  }
}
```

Apêndice E

Código de Aquisição de Dados e Controle do CREAT1

Código principal de aquisição e controle do CREAT1.

```
////////////////////////////////////////////////////////
// Código de Aquisição de dados e controle do CREAT1
// Authors:   André Massafferri Rodrigues (massafferri@cbpf.br)
//         Ulisses Carneiro
//         Leonardo Chaves Ruiz Guedes (leocrg@cbpf.br)
////////////////////////////////////////////////////////

#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include "CAENVMElib.h"
#include "CAENVMEtypes.h"
#include "CAENVMEoslib.h"
#include <unistd.h>
#include <time.h>
#include <math.h>
#include <fcntl.h>      //Incluido para utilizar ctes O_RDWR | O_NOCTTY |
   O_NDELAY
#include <termios.h>    //Esta é a biblioteca de comunicação serial
#include <string.h>
#include <libgen.h>
#include <time.h>
#define ninput 16
```

```
#include <stdio.h>
#include <math.h>

extern void CaenVmeManual(long,short);
int cont_event;

void myArduino(const char* const thresholdValue, char* const v_USB)
{
  struct termios options;
  int fd;
  int k;
  tcgetattr(fd, &options);
  cfsetispeed(&options, B9600);        //Define velocidade de entrada
  cfsetospeed(&options, B9600);        //Define velocidade de saída
  options.c_cflag &= ~PARENB;          //Desliga a paridade
  options.c_cflag &= ~CSTOPB;          //Define 1 stop bit
  options.c_cflag &= ~CSIZE;           //Prepara a máscara de bit size
  options.c_cflag |= CS8;              //Select 8 data bits
  options.c_cc[VMIN] = 0;               //Do código exemplo original
  options.c_cc[VTIME] = 1;              //Do código exemplo original
  options.c_cflag |= (CLOCAL | CREAD); //Enable receiver and set local mode
  options.c_cflag &= ~CRTSCTS;     //NoHardwareFlowCtl Config.
  tcsetattr(fd, TCSANOW, &options);
  char c_USB[80]="/dev/ttyUSB1";
  k = sprintf(c_USB, "/dev/ttyUSB%c",*v_USB);
  const char *device = c_USB;

  fd = open(device, O_RDWR | O_NOCTTY | O_NDELAY);
  if(fd == -1) {
    printf( "Erro na abertura da porta\n" );
  }

  FILE * pFile;
  static char thr [65],temp[64];
  char dados[4] = "";
  int aux_thr=0;
  int aux_ch=0;
  int wordsWritten;
  int s,f;
  char ch_v[16] = "abcdefghijklmnop";
  char ch_Read[16] = "ABCDEFGHIJKLMNOP" ;
```

```
// get common threshold value from argument
if ( thresholdValue ) {
  register int c ;
  for (c=0; c<16; c++) {
    thr[4*c+0] = ' ' ;
    thr[4*c+1] = thresholdValue[0] ;
    thr[4*c+2] = thresholdValue[1] ;
    thr[4*c+3] = thresholdValue[2] ;
  }
}

pFile = fopen ("thr_chs.txt" , "r");
if (pFile == NULL) perror ("Error opening file");
else
{
  if (fgets (temp , 65 , pFile) != NULL )
  puts (temp);
  if (fgets (temp , 65 , pFile) != NULL )
  puts (temp);
  if ( thresholdValue )
  puts(thr) ;
  else if (fgets (thr , 65 , pFile) != NULL )
  puts(thr) ;
  fclose (pFile);
}

while (aux_thr<64)
{
  f = sprintf(dados, "%c%c%c%c", ch_v[aux_ch], thr[aux_thr+1], thr[aux_thr
 +2], thr[aux_thr+3]);
  printf("writing '%s' (%d) -> Arduino\n", dados, strlen(dados)) ;
  s = strlen (dados);
  wordsWritten = write(fd, dados, s);
  if ( wordsWritten == -1 )
  perror(device) ;
  else if ( wordsWritten != s )
  (void)fprintf(stderr,
  " ### Only %d bytes written on %s (expected %d)\n",
  wordsWritten, device, s) ;
  aux_thr=aux_thr+4;
```

```
    aux_ch++;
    sleep(1);
  }
  sleep(2);
  if ( close(fd) == -1 )
  perror(device) ;
}

static void writeFile(
FILE* fp,
int itt,
unsigned short evtId,
unsigned short  Cont_input[],
char*   vThreshold,
int cont_event,
int Amplitute,
int nEvents,
unsigned int data_taq,
unsigned int out_0,
unsigned int out_1
)
{
  const int nGroups = 6 ;
  const int n_channels = 16 ;
  register int i ;
  cont_event++;
  time_t myTime = time(0) ;
  struct tm* now_time ;
  now_time = localtime(&myTime) ;
  char timeString[20] ;
  (void)strftime(timeString, sizeof(timeString),
  "%d-%m-%Y %H:%M:%S", now_time) ;
  (void)fprintf(fp, "%s ", timeString) ;
  (void)fprintf(fp, "0x%x ", evtId) ;
  (void)fprintf(fp, "%d ", nEvents ) ;
  (void)fprintf(fp, "%d ", Amplitute ) ;
  (void)fprintf(fp, "%s ", vThreshold ) ;
  (void)fprintf(fp, "%d ", data_taq ) ;
  for (i=0; i<n_channels; i++){
    (void)fprintf(fp, "%d ", Cont_input[i]) ;
  }
```

```
  (void)fprintf(fp, "\n") ;
  (void)fflush(fp) ;
}

// function to print values
static void printValues(
FILE* fp,
int itt,
unsigned short evtId,
unsigned short  Cont_input[]
)
{
  const int n_channels = 16 ;
  register int i ;
  (void)fprintf(fp, "\r\n");
  (void)fprintf(fp,
  " ---------------------------------------------------- \r\n");
  (void)fprintf(fp,
  " Interface de teste da front-End do detector SciTile  \r\n");
  (void)fprintf(fp, "\r\n");
  (void)fprintf(fp,
  " ---------------------------------------------------- \r\n");
  (void)fprintf(fp, " EvtId = 0x%x \r\n", evtId);
  (void)fprintf(fp,
  " ---------------------------------------------------- \r\n");
  for (i=0; i<n_channels; i++){
    (void)fprintf(fp, "CONT%d:     %d\r\n", i, Cont_input[i]) ;
  }
  (void)fprintf(fp,
  " ---------------------------------------------------- \r\n");
  (void)fflush(fp) ;
}

static int newEvent(unsigned short evtId)
{
  static int firstTime = 1 ;
  static unsigned short prev_evtId ;

  if ( firstTime ) {
    firstTime = 0 ;
    prev_evtId = evtId ;
```

```
    return(0) ;
  }

  if ( evtId != prev_evtId ) {
    prev_evtId = evtId ;
    return(1) ;
  } else
  return(0) ;
}

// function to print values
static int anythingChanged( unsigned short  Cont_input[] )
{
  static unsigned short   prev_Cont_input[ninput] ;
  static int firstTime = 1 ;
  int s ;

  if ( firstTime ) {
    firstTime = 0 ;
    for (s=0; s<ninput; s++) {
      prev_Cont_input[s] = Cont_input[s] ;
    }
    return(1) ;
  }

  int somethingChanged = 0 ;
  for (s=0; s<ninput; s++) {
    if ( prev_Cont_input[s] != Cont_input[s] )
    somethingChanged = 1 ;
    prev_Cont_input[s] = Cont_input[s] ;
  }

  return(somethingChanged) ;
}

static void readVME(
long BHandle,
CVAddressModifier Am,
CVDataWidth Dw16,
unsigned short  Cont_input[],
unsigned short* evtId
```

```
)
{
  unsigned short data ;
  CAENVME_ReadCycle(BHandle, 0x32131300, &data, Am, Dw16);
  Cont_input[0] = data;
  CAENVME_ReadCycle(BHandle, 0x32131302, &data, Am, Dw16);
  Cont_input[1] = data;
  CAENVME_ReadCycle(BHandle, 0x32131304, &data, Am, Dw16);
  Cont_input[2] = data;
  CAENVME_ReadCycle(BHandle, 0x32131308, &data, Am, Dw16);
  Cont_input[3] = data;
  CAENVME_ReadCycle(BHandle, 0x3213130A, &data, Am, Dw16);
  Cont_input[4] = data;
  CAENVME_ReadCycle(BHandle, 0x3213130C, &data, Am, Dw16);
  Cont_input[5] = data;
  CAENVME_ReadCycle(BHandle, 0x32131310, &data, Am, Dw16);
  Cont_input[6] = data;
  CAENVME_ReadCycle(BHandle, 0x32131312, &data, Am, Dw16);
  Cont_input[7] = data;
  CAENVME_ReadCycle(BHandle, 0x32131314, &data, Am, Dw16);
  Cont_input[8] = data;
  CAENVME_ReadCycle(BHandle, 0x32131318, &data, Am, Dw16);
  Cont_input[9] = data;
  CAENVME_ReadCycle(BHandle, 0x3213131A, &data, Am, Dw16);
  Cont_input[10] = data;
  CAENVME_ReadCycle(BHandle, 0x3213131C, &data, Am, Dw16);
  Cont_input[11] = data;
  CAENVME_ReadCycle(BHandle, 0x32131320, &data, Am, Dw16);
  Cont_input[12] = data;
  CAENVME_ReadCycle(BHandle, 0x32131322, &data, Am, Dw16);
  Cont_input[13] = data;
  CAENVME_ReadCycle(BHandle, 0x32131324, &data, Am, Dw16);
  Cont_input[14] = data;
  CAENVME_ReadCycle(BHandle, 0x32131328, &data, Am, Dw16);
  Cont_input[15] = data;

  // read unique event label (acquisition counter)
  CAENVME_ReadCycle(BHandle, 0x32132006, &data, Am, Dw16) ;
  *evtId = data ;
}
```

```
static void usage(char* s)
{
  (void)fprintf(stdout,
  "\nUsage: %s [-hH -F fileName -A acq_time -G gate -n NEvts -N nim_mon -t
    thresh]\n\n",
  basename(s)) ;
}

int main(int argc, char** argv)
{
  int option = 0;
  char fileName[1024] = "outFile" ;
  unsigned int data_taq    = 300;
  unsigned int out_0    = 0;
  unsigned int out_1    = 16;
  unsigned int data_tlatch = 30;
  unsigned int data_nim    = 1; // 8-lemos
  char* vThreshold = 0 ;
  int nEvents = -1 ;
  int Amplitute = 0;
  char *v_USB = "1";

  while ((option = getopt(argc, argv, "hHA:F:G:N:n:t:v:o:O:U:")) != -1) {
    switch (option) {
      case 'h':
      case 'H':
      usage(argv[0]) ;
      exit(0) ;
      case 'A' :
      data_taq = atoi(optarg) ;
      break;
      case 'F' :
      (void)strcpy(fileName, optarg) ;
      break;
      case 'G' :
      data_tlatch = atoi(optarg) ;
      break;
      case 'n' :
      nEvents = atoi(optarg) ;
      break;
      case 'N' :
```

```
    data_nim = atoi(optarg) ;
    break;
    case 't' :
    vThreshold = calloc(4, 1) ;
    (void)strncpy(vThreshold, optarg, 3) ;
    vThreshold[3] = '\0' ;
    break;
    case 'v' :
    Amplitute = atoi(optarg);
    break;
    case 'o' :
    out_0 = atoi(optarg);
    break;
    case 'O' :
    out_1 = atoi(optarg);
    break;
    case 'U' :
    v_USB = (optarg);
    break;
    default:
    usage(argv[0]) ;
    exit(1) ;
  }
}

// create time string
time_t myTime = time(0) ;
struct tm* now_time ;
now_time = localtime(&myTime) ;
char timeString[20] ;
strftime(timeString, sizeof(timeString), "%d-%m-%Y_%H:%M:%S", now_time) ;

// open log file
char screenDump[1024] ;
(void)sprintf(screenDump, "data/%s_%s", "screenDump", timeString) ;
FILE* fp = fopen(screenDump, "w+") ;
if ( !fp )
perror(screenDump) ;

// open data file
char outFile[1024] ;
```

```
(void)sprintf(outFile, "data/%s_%s", fileName, timeString) ;
FILE* fpOut = fopen(outFile, "w+") ;
if ( !fp )
perror(outFile) ;

(void)fprintf(stdout, "ACQ. Time = %d s\n", data_taq) ;
(void)fprintf(stdout, "Gate Duration = %d x 25ns\n", data_tlatch) ;
(void)fprintf(stdout, "Group %d -> NIM monitor\n", data_nim) ;
(void)fprintf(stdout, "Log File '%s'\n", screenDump) ;
(void)fprintf(stdout, "Data File '%s'\n", outFile) ;
if ( vThreshold )
(void)fprintf(stdout, "Common Threshold = '%s'\n", vThreshold) ;
(void)fprintf(stdout, "-------------------------------------------------\n"
 ) ;

// configure electronics via Arduino
myArduino(vThreshold,v_USB) ;

// ------------------------------------------------------------------

CVBoardTypes VMEBoard;
short Link,Device;
long BHandle;

unsigned short data;

CVAddressModifier Am      = cvA32_U_DATA;
CVDataWidth       Dw16    = 0x02;
CVDataWidth       Dw32    = 0x04;

VMEBoard = cvV1718;
Link = 0;

unsigned short Cont_input[ninput];

float time_latch = (float)data_taq; //in secs

if (CAENVME_Init(VMEBoard, Device, Link, &BHandle) != cvSuccess)
{
  printf("Error\n");
  return 0;
```

```
  } else {
  printf(" starting ..... \n");
  printf(" time acq = %f sec : time freq = 1 sec", time_latch);
}

int k,i;
int itt=0;
unsigned short evtId = 0 ;

// give reset sequence as instructed by Ulisses
unsigned int data_reset = 1;
CAENVME_WriteCycle(BHandle, 0x32132000, &data_reset, Am, Dw16);

sleep(2) ;

CAENVME_WriteCycle(BHandle, 0x3213200A, &out_0, Am, Dw16);
CAENVME_WriteCycle(BHandle, 0x3213200C, &out_1, Am, Dw16);
CAENVME_WriteCycle(BHandle, 0x32132000, &data_taq, Am, Dw16);
CAENVME_WriteCycle(BHandle, 0x32132002, &data_tlatch, Am, Dw16);
CAENVME_WriteCycle(BHandle, 0x32132004, &data_nim, Am, Dw16);

int nCycles = 0 ;

while (k = 1)
{
  readVME(
  BHandle, Am, Dw16,
  Cont_input,
  &evtId
  ) ;


  // data changed
  int newData = anythingChanged(Cont_input) ;
  // unique event Id (acquisition counter) changed
  int newEvt = newEvent(evtId) ;

  // error : we have new data values, but old evt Id
  if ( newData && (!newEvt) )
  (void)fprintf(stderr,
  " ############################## Error: data changed!\n") ;
```

```
    // warning: we had a new acquisition with same data as previous one
    if ( (!newData) && newEvt )
    (void)fprintf(stderr,
    " >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Warning: no new data\n") ;

    if ( newEvt ) {
      nCycles++ ;
      printValues(
      stdout, itt, evtId,
      Cont_input
      ) ;
          printValues(
      fp, itt, evtId,
      Cont_input
      ) ;
      writeFile(
      fpOut, itt, evtId,
      Cont_input, vThreshold, cont_event, Amplitute, nEvents, out_0, out_1,
   data_taq
      ) ;
    }
    sleep(2);
    itt++;

    if ( nEvents != -1 && nCycles >= nEvents )
    break ;
  }

  CAENVME_End(BHandle);

  if ( fp ) (void)fclose(fp) ;
  if ( fpOut ) (void)fclose(fpOut) ;

  return 0;

}
```

Script LINUX para *threshold scan* do CREAT1.

```
#################################################################
# Script de varredura do threshold
# Author: Leonardo Chaves Ruiz Guedes (leocrg@cbpf.br)
```

```
#################################################################

#!/bin/sh

THR_LIST='  10   20   30   40   50   60   70   80   90   100 110 120 130 140 150 '

THR_LIST_MAX=15
THR_LIST_MIN=1
THR_STEP=1

ARDUINO_PORT=0

######### Settings File #########
CH=CH_all
Amplitute=ruido #10_mv
Width=real
Freq_In=real

FILE_BASENAME=RUN_TestScan_Width_${Width}_freq_${Freq_In}_Amp_${Amplitute}_${CH}_Ruido_v
SaveDataAq=1
AddThr_file=1

t=$THR_LIST_MIN

Amplitute_neg=-${Amplitute}

if [ $SaveDataAq -eq 1 ]; then

#for t in $THR_LIST ; do
while [ $t -le $THR_LIST_MAX ]

do
  vThr=`awk ' END { printf "%03d", '"$t"' + 0 }' /dev/null`
  outFile=${FILE_BASENAME}_T${vThr}
  ./SaveDataAq_v2.a -n 10 -A 1 -t $vThr -G 1000 -F $outFile -v $Amplitute_neg -O 0026 -c
  sleep 3
  ./vthr_leo.sh read > Log/${outFile}_`date '+%H:%M:%S_%d-%m-%Y'`.thr_log
  t=`expr $t + $THR_STEP`
done

fi
```

```
outFile=RUN_TestScan_ger_${Amplitute}_${CH}_v3.dat

#/bin/rm -f fwv25.dat

THR_LIST_MAX=190
THR_LIST_MIN=1
THR_STEP=1

t=$THR_LIST_MIN

if [ $AddThr_file -eq 1 ]; then

echo '    Data          UN_ID    N_ev   Amp(mV)  Thr(mV)  T_aq(s)  Count0   Count1   Count2   Cou

#for FILE_BASENAME in RUN_TestScan_Width_5ns_freq_1khz_Amp_5mv_${CH}_v1    RUN_TestScan_
#do        #1

filnam=data/${FILE_BASENAME}_T

#for t in 10 20
#for t in $THR_LIST ;
while [ $t -le $THR_LIST_MAX ]
do
  myThr=`echo $t | awk ' END { printf "%03d", '"$t"' } '`
# myThr=`awk ' END { printf "%03d", '"$t"' } '`
  infil=`ls ${filnam}${myThr}*`
    if ls ${filnam}${myThr}* 1> /dev/null 2>&1; then
      awk ' { printf "%s\n", $0 } ' $infil
    fi
  t=`expr $t + $THR_STEP`

#done        #1
#t=$THR_LIST_MIN   #1

done >> $outFile
fi

exit 0
```

Apêndice F

Código Principal da Firmware de Aquisição e Controle do Experimento CREAT2

```
-- Descrio do projeto:
-- ******************************************************************************
-- Company:              CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:              CRE@AT - Cosmic Ray Experiment at Antartida
-- Model:                V1495 -  Multipurpose Programmable Trigger Unit
-- FPGA Proj. Name:      Test_FrontEnd_SciTile
-- Device:               ALTERA EP1C4F400C6
-- Author:               Leonardo Chaves Ruiz Guedes (leocrgster@gmail.com)
-- Date:                 28-01-2015
-- ------------------------------------------------------------------------------
-- Baseado na firmware do CREAT1 (Ulisses Carneiro)
-- Baseado na firmware demo version (Luca Colombini) disponível para download em:
-- CAEN SpA at url http://www.caen.it/csite/CaenProd.jsp?idmod=484&parent=11
--
-- ------------------------------------------------------------------------------
-- ******************************************************************************
--


LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
```

```
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;        -- (Para utilizar a funo OR_REDUCE)

USE work.v1495pkg.all;              -- Contm o mapa de registradores do BUS VME

ENTITY Test_LogicAnalyzer IS
  PORT(
    -- ORIGINAL DO CODIGO DA CAEN (NAO MODIFICAR!!!):
    nLBRES      : IN     std_logic;                      -- Async Reset (act L)
    LCLK        : IN     std_logic;                      -- Local Bus Clock
    --***********************************************
      -- REGISTER INTERFACE
    --***********************************************
    REG_WREN    : IN     std_logic;                      -- Write pulse (act H)
    REG_RDEN    : IN     std_logic;                      -- Read  pulse (act H)
    REG_ADDR    : IN     std_logic_vector (15 DOWNTO 0);-- Register address
    REG_DIN     : IN     std_logic_vector (15 DOWNTO 0);-- Data from Local Bus
    REG_DOUT    : OUT    std_logic_vector (15 DOWNTO 0);-- Data to   Local Bus
    USR_ACCESS  : IN     std_logic;                      -- Current reg. access
    -- at user address space
      --***********************************************
      -- V1495 Front Panel Ports (PORT A,B,C,G)
    --***********************************************
    A_DIN       : IN     std_logic_vector (31 DOWNTO 0);-- In A (32 x LVDS/ECL)
    B_DIN       : IN     std_logic_vector (31 DOWNTO 0);-- In B (32 x LVDS/ECL)
    C_DOUT      : OUT    std_logic_vector (31 DOWNTO 0);-- Out C (32 x LVDS)
    G_LEV       : OUT    std_logic;                      -- Output Level(NIM/TTL)
    G_DIR       : OUT    std_logic;                      -- Output Enable
    G_DOUT      : OUT    std_logic_vector (1 DOWNTO 0); -- Out G - (2 x NIM/TTL)
    G_DIN       : IN     std_logic_vector (1 DOWNTO 0); -- In G - (2 x NIM/TTL)
    --***********************************************
      -- A395x MEZZANINES INTERFACES (PORT D,E,F)
      --***********************************************
      -- Expansion Mezzanine Identifier:
      -- x_IDCODE :
      -- 000 : A395A (32 x IN LVDS/ECL)
      -- 001 : A395B (32 x OUT LVDS)
      -- 010 : A395C (32 x OUT ECL)
    -- 011 : A395D (8  x IN/OUT NIM/TTL)

    -- Expansion Mezzanine Port Signal Standard Select
```

```
  -- x_LEV :
--     0=>TTL,1=>NIM

-- Expansion Mezzanine Port Direction
  -- x_DIR :
--     0=>OUT,1=>IN

-- In/Out D (I/O Expansion)
D_IDCODE     : IN     std_logic_vector ( 2 DOWNTO 0);-- D slot mezzanine ID
D_LEV        : OUT    std_logic;                     -- D slot Port Level Sel
D_DIR        : OUT    std_logic;                     -- D slot Port Direction
D_DIN        : IN     std_logic_vector (31 DOWNTO 0);-- D slot Data In  Bus
D_DOUT       : OUT    std_logic_vector (31 DOWNTO 0);-- D slot Data Out Bus
-- In/Out E (I/O Expansion)
E_IDCODE     : IN     std_logic_vector ( 2 DOWNTO 0);-- E slot mezzanine ID
E_LEV        : OUT    std_logic;                     -- E slot Port Level Sel
E_DIR        : OUT    std_logic;                     -- E slot Port Direction
E_DIN        : IN     std_logic_vector (31 DOWNTO 0);-- E slot Data In  Bus
E_DOUT       : OUT    std_logic_vector (31 DOWNTO 0);-- E slot Data Out Bus
-- In/Out F (I/O Expansion)
F_IDCODE     : IN     std_logic_vector ( 2 DOWNTO 0);-- F slot mezzanine ID
F_LEV        : OUT    std_logic;                     -- F slot Port Level Sel
F_DIR        : OUT    std_logic;                     -- F slot Port Direction
F_DIN        : IN     std_logic_vector (31 DOWNTO 0);-- F slot Data In  Bus
F_DOUT       : OUT    std_logic_vector (31 DOWNTO 0);-- F slot Data Out Bus
--***********************************************
  -- DELAY LINES
  --***********************************************
  -- PDL = Programmable Delay Lines  (Step = 0.25ns / FSR = 64ns)
  -- DLO = Delay Line Oscillator     (Half Period ~ 10 ns)
-- 3D3428 PDL (PROGRAMMABLE DELAY LINE) CONFIGURATION
PDL_WR       : OUT    std_logic;                     -- Write Enable
PDL_SEL      : OUT    std_logic;                     -- PDL Sel. (PDL0/PDL1)
PDL_READ     : IN     std_logic_vector ( 7 DOWNTO 0);-- Read Data
PDL_WRITE    : OUT    std_logic_vector ( 7 DOWNTO 0);-- Write Data
PDL_DIR      : OUT    std_logic;                     -- Direction (0>W 1>R)
-- DELAY I/O
PDL0_OUT     : IN     std_logic;                     -- Signal from PDL0 Out
PDL1_OUT     : IN     std_logic;                     -- Signal from PDL1 Out
DLO0_OUT     : IN     std_logic;                     -- Signal from DLO0 Out
DLO1_OUT     : IN     std_logic;                     -- Signal from DLO1 Out
```

```
    PDL0_IN      : OUT    std_logic;                    -- Signal to   PDL0 In
    PDL1_IN      : OUT    std_logic;                    -- Signal to   PDL1 In
    DLO0_GATE    : OUT    std_logic;                    -- DLO0 Gate (active H)
    DLO1_GATE    : OUT    std_logic;                    -- DLO1 Gate (active H)
    ----------------------------------------------
      -- SPARE PORTS
    ----------------------------------------------
    SPARE_OUT    : OUT    std_logic_vector(11 downto 0); -- SPARE Data Out
    SPARE_IN     : IN     std_logic_vector(11 downto 0); -- SPARE Data In
    SPARE_DIR    : OUT    std_logic_vector(11 downto 0); -- SPARE Dir(0>OUT 1>IN)
    ----------------------------------------------
      -- LED
    ----------------------------------------------
    RED_PULSE       : OUT    std_logic;                 -- RED led  (active H)
    GREEN_PULSE     : OUT    std_logic                  -- GREEN led(active H)
  );

  -- Declarations

END Test_LogicAnalyzer ;

ARCHITECTURE rtl OF Test_LogicAnalyzer IS

  -- CONSTANTES:

  type in_ch_array   is array(0 to 2, 0 to 4) of integer range 0 to 100;

  type coin_map_type   is array(0 to 13) of integer range 0 to 15;

  constant IN_PATCH :in_ch_array := ((15-00, 15-03, 15-04, 15-10, 15-12),
                    (15-01, 15-05, 15-08, 15-11, 15-13),
                    (15-02, 15-06, 15-09, 15-15, 15-14));

  -- ////00//01/////07//08//
  -- //////02/////////09////
  -- ////03//04/////10//11//
  -- ////05//06/////12//13//

  constant Coin_Map:coin_map_type := (14,10,12,2,1,9,5,7,6,13,4,0,3,11);
```

```
constant N_NIB        :integer := 4;   -- Preciso dos contadores. 1nibble = 4bits
constant N_CANAIS     :integer := 40;   -- Define qte. de canais + coincidencias
constant N_DATASEND   :integer := (N_CANAIS + 1)*4 + N_CANAIS + 2;
constant N_INPUTS     :integer := 16;   -- Define qte. de entradas por canal
constant C_GATE       :unsigned(N_NIB*4-1 downto 0):= conv_unsigned(5, N_NIB*4);
signal   N_GATE       :unsigned(N_NIB*4-1 downto 0):= C_GATE;
-- TEMPO DE AQUISICAO:
constant N_T_1SEG     :unsigned:= conv_unsigned(40000000, 32);
-- 1 unidade de N_T_1SEG equivale a 25ns.
-- Valor tipico: 40000000  = 1seg
constant C_TAQUIS     :unsigned:= conv_unsigned(00030, 16);

constant C_out_signal_0     :std_logic_vector(15 downto 0):= x"0026";
constant C_out_signal_1     :std_logic_vector(15 downto 0):= x"0026";

-- 1 unidade de N_TAQUIS equivale a N_T_1SEG*25ns (IDEALMENTE 1s).
-- Valores tipicos: 00010  = 10seg
--                  00020  = 20seg
--                  00120  =  2min
--                  01800  = 30min
--                  03600  = 60min
signal   N_TAQUIS     :unsigned(15 DOWNTO 0);--:=C_TAQUIS;

signal   N_out_signal_0     : std_logic_vector(15 downto 0):=C_out_signal_0;
signal   N_out_signal_1     : std_logic_vector(15 downto 0):=C_out_signal_1;

constant N_GEN_TTOTAL:integer := 400000;     -- Intervalo entre pulsos do gerador 40000
constant N_GEN_LPULSO:integer := 2;          -- Largura de pulso do gerador
constant C_CTL_ANAL  :std_logic_vector(15 downto 0):= x"2012";

-- EXEMPLO DE CALCULO DA FREQUENCIA DO GERADOR:
-- fLCLK = 40 MHz   >>>   fGEN = 40MHz / (N_GEN_TTOTAL)
-- N_GEN_TTOTAL = fLCLK / fGEN = 40x10^6 / 10
-- Para fGEN = 10 Hz temos que N_GEN_TTOTAL = 4x10^6 = 4000000


type unsigned_array  is array(N_CANAIS-1 downto 0, N_INPUTS-1 downto 0)
of unsigned(N_NIB*4-1 downto 0);
type unsigned_vector is array(N_CANAIS-1 downto 0                      )
of unsigned(N_NIB*4-1 downto 0);
type vector_array  is array(         N_CANAIS-1         downto 0)
```

```
of std_logic_vector(N_INPUTS-1 downto 0);

-------------------------------------------------------------------------------

signal T_gate    :integer:=conv_integer(N_GATE(N_NIB*4-1 downto 0));

type integer_array    is array(N_CANAIS -1 downto 0)
of integer range 0 to T_gate+1;

type std_logic_array is array(N_CANAIS-1 downto 0)
of unsigned(N_NIB*4-1 downto 0);
type std_logic_DataSend is array(N_DATASEND-1 downto 0)
of unsigned(N_NIB*2-1 downto 0);


signal n_event    :std_logic_array;
signal n_event_aux    :std_logic_array;
signal DataSend    :std_logic_DataSend;

-- signal INPUT      :std_logic_vector(15 downto 0):=x"0000";
signal INPUT      :std_logic_vector(N_CANAIS-1 downto 0):=x"0000000000";


-- signal CANAL      :std_logic_vector(15 downto 0);
signal CANAL      :std_logic_vector(N_CANAIS-1 downto 0);

signal FLAG_AQ     :std_logic :='0';
signal FLAG_COPY     :std_logic :='0';
signal AUX_COPY    :std_logic_vector(N_CANAIS-1 downto 0) :=x"0000000000";

signal FLAG_DET    :std_logic_vector(N_CANAIS-1 downto 0):=x"0000000000";
signal DET_CANAL   :std_logic_vector(N_CANAIS-1 downto 0):=x"0000000000";
signal DET_CANAL_OLD :std_logic_vector(N_CANAIS-1 downto 0):=x"FFFFFFFFFF";
signal DET_XOR :std_logic_vector(N_CANAIS-1 downto 0);
signal NUM_GATE      :std_logic_array;
signal RESET       :std_logic := '0';

-------------------------------------------------------------------------------

signal LATCH_C2           : unsigned_array;
signal U_ID                : unsigned(N_NIB*4-1 downto 0):=conv_unsigned(0, N_NIB*4);
```

```
-- SINAIS DE CONTROLE:

-- Processamento da contagem:
signal LED_DEBUG1 : std_logic;  -- LED_DEBUG
signal LED_DEBUG2 : std_logic;  -- LED_DEBUG

-- Gerador de sinais:
signal CLK_STATE  : std_logic;        -- Clock da Maquina de Estados do Gerador
signal GO_GEN     : std_logic;        -- Pulso de disparo do gerador
signal GEN_FLAG1  : std_logic:='1'; -- Elemento da maquina de estados do gerador
signal GEN_FLAG2  : std_logic:='0'; -- Elemento da maquina de estados do gerador
signal GEN1       : std_logic;       -- Canal 1a da saida do gerador.

signal GEN2       : std_logic;       -- Canal 2a da saida do gerador.
signal GEN3       : std_logic;       -- Canal 3a da saida do gerador.
signal GEN1a      : std_logic;       -- Canal 1a da saida do gerador.
signal GEN2a      : std_logic;       -- Canal 2a da saida do gerador.
signal GEN3a      : std_logic;       -- Canal 3a da saida do gerador.
signal GEN1b      : std_logic;       -- Canal 1b da saida do gerador.
signal GEN2b      : std_logic;       -- Canal 2b da saida do gerador.
signal GEN3b      : std_logic;       -- Canal 3b da saida do gerador.
signal GEN1c      : std_logic;       -- Canal 1c da saida do gerador.
signal GEN2c      : std_logic;       -- Canal 2c da saida do gerador.
signal GEN3c      : std_logic;       -- Canal 3c da saida do gerador.

signal GEN1d      : std_logic;       -- Canal 1d da saida do gerador.
signal GEN2d      : std_logic;       -- Canal 2d da saida do gerador.
signal GEN3d      : std_logic;       -- Canal 3d da saida do gerador.
signal GEN1e      : std_logic;       -- Canal 1e da saida do gerador.
signal GEN2e      : std_logic;       -- Canal 2e da saida do gerador.
signal GEN3e      : std_logic;       -- Canal 3e da saida do gerador.
signal CONT_GEN_POSICAO : integer range 1 to 1000000001:=1; -- Perodo do gerador

-- Definicao de tipo de estados para gerador de sinais:
type STATE_type is
(S0, S1, S2);
signal STATE: STATE_type;

-- DIVISORES DE CLOCK:
signal CONT_T_1SEG      : unsigned(31 DOWNTO 0):=conv_unsigned(0, 32); -- Tempo de aqu
```

```
  signal CONT_TAQUIS      : unsigned(15 DOWNTO 0):=conv_unsigned(0, 16); -- Tempo de aqu
  signal CONT_GEN_TTOTAL : integer range 1 to 1000000001:=1; -- Perodo do gerador
  signal CONT_GEN_LPULSO : integer range 0 to 1000000000:=0; -- Larg. pulso gerador

  signal GO_1SEG          : std_logic:='1';  -- Pulso de disparo do contador de T. Aquisi

  signal PATCH_NIM        : integer range 0 to N_CANAIS-1 := 0;
  signal CTL_ANALISE      : std_logic_vector(15 downto 0):=C_CTL_ANAL;

  signal outBuff : std_logic_vector (9 downto 0);  -- envia do zero ao 7 do byte

  signal ByteIn : std_logic_vector(7 downto 0);
  signal ByteAux : std_logic_vector(7 downto 0);

  signal Busy : std_logic:= '0';
  signal Rx_Out : std_logic;
  signal Go : std_logic:='0';
  signal Go_Antes : std_logic:='0';
  signal cont : integer range 0 to 4200:=0;     -- observar o limite ao mudar o clock /
  signal bits_Sent : integer range 0 to 10:=0;

  type Estados is (Idle, Sending);
  signal Estado : Estados;

  signal Byte_em_Espera : boolean := false;

  signal aux_serial : integer range 0 to N_DATASEND:=0;

  signal envia_serial  :std_logic := '0';
  signal aux_event  :std_logic := '0';
  signal flag_serial  :std_logic:='1';

  signal n : integer range -1 to N_NIB:= N_NIB-1;

  ------------------------


begin  -- incio da arquitetura

  --  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  --  XXXXXXXX            CONFIGURACOES DA V1495          XXXXXXXXXXXXXXXXXX
```

```
--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


--*********************************************

-- PORT Levels: '0'=TTL '1'=NIM
D_LEV <= '1';
E_LEV <= '1';
F_LEV <= '1';
G_LEV <= '0';

--*********************************************
-- PORT Directions: '0'=Output '1'=Input
D_DIR  <= '0';
E_DIR  <= '0';  ----ATENCAO!!!!!!!
F_DIR  <= '1';
G_DIR  <= '0';

--

--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
--  XXXXXXXX              DEBUG REENTRY SECTION            XXXXXXXXXXXXXXXXXX
--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

RED_PULSE <= LED_DEBUG1;
GREEN_PULSE <= LED_DEBUG2;

--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
--  XXXXXXXX         CONEXOES ELETRICAS ESTATICAS         XXXXXXXXXXXXXXXXXX
--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX



CANAL(15 - 0) <= A_DIN(0);     -- Entrada 1a
CANAL(15 - 1) <= A_DIN(1);     -- Entrada 2a
CANAL(15 - 2) <= A_DIN(2);     -- Entrada 3a
CANAL(15 - 3) <= A_DIN(3);     -- Entrada 1b
CANAL(15 - 4) <= A_DIN(4);     -- Entrada 2b
CANAL(15 - 5) <= A_DIN(5);     -- Entrada 3b
CANAL(15 - 6) <= A_DIN(6);     -- Entrada 1c
```

```
CANAL(15 - 7) <= A_DIN(7);     -- Entrada 2c
CANAL(15 - 8) <= A_DIN(8);     -- Entrada 3c
CANAL(15 - 9) <= A_DIN(9);    -- Entrada 1d
CANAL(15 - 10) <= A_DIN(10);    -- Entrada 2d
CANAL(15 - 11) <= A_DIN(11);    -- Entrada 3d
CANAL(15 - 12) <= A_DIN(12);    -- Entrada 1e
CANAL(15 - 13) <= A_DIN(13);    -- Entrada 2e
CANAL(15 - 14) <= A_DIN(14);    -- Entrada 3e
CANAL(15 - 15) <= A_DIN(15);    -- Entrada 1f


-- Conexoes eletricas estaticas das saidas do nosso gerador:
C_DOUT(IN_PATCH(0,0)) <=  GEN1;    -- Saida 1 do Gerador
C_DOUT(IN_PATCH(1,0)) <=  GEN2;    -- Saida 2 do Gerador
C_DOUT(IN_PATCH(2,0)) <=  GEN3;    -- Saida 3 do Gerador
C_DOUT(IN_PATCH(0,1)) <=  GEN1;    -- Saida 1 do Gerador
C_DOUT(IN_PATCH(1,1)) <=  GEN2;    -- Saida 2 do Gerador
C_DOUT(IN_PATCH(2,1)) <=  GEN3;    -- Saida 3 do Gerador
C_DOUT(IN_PATCH(0,2)) <=  GEN1;    -- Saida 1 do Gerador
C_DOUT(IN_PATCH(1,2)) <=  GEN2;    -- Saida 2 do Gerador
C_DOUT(IN_PATCH(2,2)) <=  GEN3;    -- Saida 3 do Gerador
C_DOUT(IN_PATCH(0,3)) <=  GEN1;    -- Saida 1 do Gerador
C_DOUT(IN_PATCH(1,3)) <=  GEN2;    -- Saida 2 do Gerador
C_DOUT(IN_PATCH(2,3)) <=  GEN3;    -- Saida 3 do Gerador
C_DOUT(IN_PATCH(0,4)) <=  GEN1;    -- Saida 1 do Gerador
C_DOUT(IN_PATCH(1,4)) <=  GEN2;    -- Saida 2 do Gerador
C_DOUT(IN_PATCH(2,4)) <=  GEN3;    -- Saida 3 do Gerador
C_DOUT(16) <=  GEN1;     -- Saida 1 do Gerador
C_DOUT(17) <=  GEN2;     -- Saida 2 do Gerador
C_DOUT(18) <=  GEN3;     -- Saida 3 do Gerador

E_DOUT(3) <=  GEN1;
E_DOUT(2) <=  GEN2;
E_DOUT(1) <=  GEN3;

----------CODIGO DE TESTE---------

process(N_out_signal_0,N_out_signal_1)
  begin
  case N_out_signal_0 is
    when x"0000" =>
```

```
G_DOUT(0) <= CANAL(0);
when x"0001" =>
G_DOUT(0) <= CANAL(1);
when x"0002" =>
G_DOUT(0) <= CANAL(2);
when x"0003" =>
G_DOUT(0) <= CANAL(3);
when x"0004" =>
G_DOUT(0) <= CANAL(4);
when x"0005" =>
G_DOUT(0) <= CANAL(5);
when x"0006" =>
G_DOUT(0) <= CANAL(6);
when x"0007" =>
G_DOUT(0) <= CANAL(7);
when x"0008" =>
G_DOUT(0) <= CANAL(8);
when x"0009" =>
G_DOUT(0) <= CANAL(9);
when x"000A" =>
G_DOUT(0) <= CANAL(10);
when x"000B" =>
G_DOUT(0) <= CANAL(11);
when x"000C" =>
G_DOUT(0) <= CANAL(12);
when x"000D" =>
G_DOUT(0) <= CANAL(13);
when x"000E" =>
G_DOUT(0) <= CANAL(14);
when x"000F" =>
G_DOUT(0) <= CANAL(15);

when x"0010" =>
G_DOUT(0) <= INPUT(0);
when x"0011" =>
G_DOUT(0) <= INPUT(1);
when x"0012" =>
G_DOUT(0) <= INPUT(2);
when x"0013" =>
G_DOUT(0) <= INPUT(3);
when x"0014" =>
```

```
G_DOUT(0) <= INPUT(4);
when x"0015" =>
G_DOUT(0) <= INPUT(5);
when x"0016" =>
G_DOUT(0) <= INPUT(6);
when x"0017" =>
G_DOUT(0) <= INPUT(7);
when x"0018" =>
G_DOUT(0) <= INPUT(8);
when x"0019" =>
G_DOUT(0) <= INPUT(9);
when x"001A" =>
G_DOUT(0) <= INPUT(10);
when x"001B" =>
G_DOUT(0) <= INPUT(11);
when x"001C" =>
G_DOUT(0) <= INPUT(12);
when x"001D" =>
G_DOUT(0) <= INPUT(13);
when x"001E" =>
G_DOUT(0) <= INPUT(14);
when x"001F" =>
G_DOUT(0) <= INPUT(15);

when x"0020" =>
G_DOUT(0) <= INPUT(30); --GO_1SEG
when x"0021" =>
G_DOUT(0) <= FLAG_AQ;
when x"0022" =>
G_DOUT(0) <= LCLK;
when x"0023" =>
G_DOUT(0) <= DET_CANAL(0);
when x"0024" =>
G_DOUT(0) <= DET_CANAL_OLD(0);
when x"0025" =>
G_DOUT(0) <= DET_XOR(0);
when x"0026" =>
G_DOUT(0) <= Rx_Out;
when x"0027" =>
G_DOUT(0) <= Go;
when x"0028" =>
```

```
    G_DOUT(0) <= Busy;
    when x"0029" =>
    G_DOUT(0) <= INPUT(32);
    when others =>
    G_DOUT(0) <= Rx_Out;
  end case;

  case N_out_signal_1 is
    when x"0000" =>
    G_DOUT(1) <= CANAL(0);
    when x"0001" =>
    G_DOUT(1) <= CANAL(1);
    when x"0002" =>
    G_DOUT(1) <= CANAL(2);
    when x"0003" =>
    G_DOUT(1) <= CANAL(3);
    when x"0004" =>
    G_DOUT(1) <= CANAL(4);
    when x"0005" =>
    G_DOUT(1) <= CANAL(5);
    when x"0006" =>
    G_DOUT(1) <= CANAL(6);
    when x"0007" =>
    G_DOUT(1) <= CANAL(7);
    when x"0008" =>
    G_DOUT(1) <= CANAL(8);
    when x"0009" =>
    G_DOUT(1) <= CANAL(9);
    when x"000A" =>
    G_DOUT(1) <= CANAL(10);
    when x"000B" =>
    G_DOUT(1) <= CANAL(11);
    when x"000C" =>
    G_DOUT(1) <= CANAL(12);
    when x"000D" =>
    G_DOUT(1) <= CANAL(13);
    when x"000E" =>
    G_DOUT(1) <= CANAL(14);
    when x"000F" =>
    G_DOUT(1) <= CANAL(15);
```

```
when x"0010" =>
G_DOUT(1) <= INPUT(0);
when x"0011" =>
G_DOUT(1) <= INPUT(1);
when x"0012" =>
G_DOUT(1) <= INPUT(2);
when x"0013" =>
G_DOUT(1) <= INPUT(3);
when x"0014" =>
G_DOUT(1) <= INPUT(4);
when x"0015" =>
G_DOUT(1) <= INPUT(5);
when x"0016" =>
G_DOUT(1) <= INPUT(6);
when x"0017" =>
G_DOUT(1) <= INPUT(7);
when x"0018" =>
G_DOUT(1) <= INPUT(8);
when x"0019" =>
G_DOUT(1) <= INPUT(9);
when x"001A" =>
G_DOUT(1) <= INPUT(10);
when x"001B" =>
G_DOUT(1) <= INPUT(11);
when x"001C" =>
G_DOUT(1) <= INPUT(12);
when x"001D" =>
G_DOUT(1) <= INPUT(13);
when x"001E" =>
G_DOUT(1) <= INPUT(14);
when x"001F" =>
G_DOUT(1) <= INPUT(15);

when x"0020" =>
G_DOUT(1) <= INPUT(30);--GO_1SEG;
when x"0021" =>
G_DOUT(1) <= FLAG_AQ;
when x"0022" =>
G_DOUT(1) <= LCLK;
when x"0023" =>
G_DOUT(1) <= DET_CANAL(0);
```

```
      when x"0024" =>
      G_DOUT(1) <= DET_CANAL_OLD(0);
      when x"0025" =>
      G_DOUT(1) <= DET_XOR(0);
      when x"0026" =>
      G_DOUT(1) <= Rx_Out;
      when x"0027" =>
      G_DOUT(1) <= Go;
      when x"0028" =>
      G_DOUT(1) <= Busy;
      when x"0029" =>
      G_DOUT(1) <= FLAG_COPY;
      when others =>
      G_DOUT(1) <= Rx_Out;

    end case;

end process;

--G_DOUT(0) <= GO_1SEG;
--G_DOUT(1) <= FLAG_AQ;

--Largura fixa para um sinal assincrono de entrada

Process(CANAL,NUM_GATE,INPUT)
    begin
    for i in 0 to (N_CANAIS - 1) loop
      if (CANAL(i)'event) and (CANAL(i)='1') and ((DET_CANAL(i) xor DET_CANAL_OLD(i)) =
        DET_CANAL(i) <= NOT DET_CANAL(i);
      end if;
    end loop;
end process;

Process(LCLK,DET_CANAL,INPUT)
    begin
    for i in 0 to (N_CANAIS - 1) loop
      if (LCLK'event) and (LCLK='0') then
        if ((DET_CANAL(i) xor DET_CANAL_OLD(i)) = '0') and (INPUT(i)='0') then
          INPUT(i) <= '1';
          NUM_GATE(i) <= conv_unsigned(0, N_NIB*4);
        end if;
```

```
      if (NUM_GATE(i) <= T_gate) and (INPUT(i) = '1') then
      NUM_GATE(i) <= NUM_GATE(i) + 1;
      elsif (NUM_GATE(i) > T_gate) then
        INPUT(i) <= '0';
        NUM_GATE(i) <= conv_unsigned(0, N_NIB*4);
      end if;
    end if;

  end loop;
end process;

Process(INPUT)
  begin
  for i in 0 to (N_CANAIS  - 1) loop
    if (INPUT(i)'event) and (INPUT(i)='1') then
      DET_CANAL_OLD(i)<= NOT DET_CANAL_OLD(i);
    end if;
  end loop;
end process;

--------Coincidencias----------

Process(CANAL,INPUT)
  begin
  -- ////00//01/////07//08//
  -- //////02/////////09////
  -- ////03//04/////10//11//
  -- ////05//06/////12//13//

  CANAL(16)<=INPUT(coin_map(0)) OR INPUT(coin_map(1));
  CANAL(17)<=INPUT(coin_map(3)) OR INPUT(coin_map(4));
  CANAL(18)<=INPUT(coin_map(5)) OR INPUT(coin_map(6));
  CANAL(19)<=INPUT(coin_map(7)) OR INPUT(coin_map(8));
  CANAL(20)<=INPUT(coin_map(10)) OR INPUT(coin_map(11));
  CANAL(21)<=INPUT(coin_map(12)) OR INPUT(coin_map(13));

  --      DET A
  CANAL(22)<=INPUT(16) AND INPUT(coin_map(2)) AND INPUT(17) AND INPUT(18);
  CANAL(23)<=INPUT(16) AND INPUT(17) AND INPUT(18);
  CANAL(24)<=INPUT(16) AND INPUT(coin_map(2)) AND INPUT(17);
  CANAL(25)<=INPUT(16) AND INPUT(coin_map(2)) AND INPUT(18);
```

```
  CANAL(26)<=INPUT(coin_map(2)) AND INPUT(17) AND INPUT(18);

  --     DET B
  CANAL(27)<=INPUT(19) AND INPUT(coin_map(9)) AND INPUT(20) AND INPUT(21);
  CANAL(28)<=INPUT(19) AND INPUT(20) AND INPUT(21);
  CANAL(29)<=INPUT(19) AND INPUT(coin_map(9)) AND INPUT(20);
  CANAL(30)<=INPUT(19) AND INPUT(coin_map(9)) AND INPUT(21);
  CANAL(31)<=INPUT(coin_map(9)) AND INPUT(20) AND INPUT(21);

  CANAL(32)<= INPUT(22) AND INPUT(27);


end process;


--contagem de pulsos

process(LCLK,FLAG_AQ)
  begin
  for i in 0 to (N_CANAIS  - 1) loop
    if (LCLK'event) and (LCLK = '1') then
      if (FLAG_AQ = '1') then
        if  (INPUT(i) = '1') and (FLAG_DET(i) = '0') then
          FLAG_DET(i) <= '1';

          if (n_event_aux (i) < 65535) then
          n_event_aux (i) <= n_event_aux (i) + 1;
          else
            n_event_aux (i) <= conv_unsigned(65535, N_NIB*4);
          end if;

        elsif  (INPUT(i) = '0') and (FLAG_DET(i) = '1') then
          FLAG_DET(i) <= '0';
        end if;
        else
          n_event_aux (i) <= conv_unsigned(0, N_NIB*4);
        end if;

    end if;

    if (FLAG_AQ'event)and (FLAG_AQ = '0') then
```

```
      n_event(i) <= conv_unsigned(n_event_aux(i), N_NIB*4);
    end if;

  end loop;
end process;

--GATE do Tempo de Aquisicao--GATE do Tempo de Aquisicao

process(LCLK)
  begin
  if (LCLK'event) and (LCLK='0') then
    if CONT_T_1SEG <= N_T_1SEG then
      CONT_T_1SEG <= CONT_T_1SEG + 1;
    GO_1SEG <= '1';
    else
      CONT_T_1SEG <= conv_unsigned(0, 32);
      GO_1SEG <= '0';
    end if;
  end if;
end process;

process(GO_1SEG)
  begin
  if (GO_1SEG'event) and (GO_1SEG='0') then
    if CONT_TAQUIS < N_TAQUIS then
      CONT_TAQUIS <= CONT_TAQUIS + 1;
    FLAG_AQ <= '1';
    else
      CONT_TAQUIS <= conv_unsigned(0, 16);
      U_ID <= U_ID + 1;
      FLAG_AQ <= '0';
    end if;
  end if;
end process;




--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
--  XXXXXXXXX                                                 XXXXXXXXXXXXXXXXXX
--  XXXXXXXXX  LEITURA DE REGISTRADORES PELO  BUS VME  XXXXXXXXXXXXXXXXXX
```

```
--  XXXXXXXXX                                                    XXXXXXXXXXXXXXXXXX
--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
--  XXXXXXXXX            INCIO DO CODIGO  VHDL           XXXXXXXXXXXXXXXXXX
--  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
--
P_RREG: process(LCLK, nLBRES)
  begin
  if (nLBRES = '0') then
  REG_DOUT   <= (others => '0');
  elsif LCLK'event and LCLK = '1' then
    if (REG_RDEN = '1') and (USR_ACCESS = '1') then
      case REG_ADDR is

        -- Registradores do Somatorio instantaneo de eventos por entrada:
        when CONT_CH0  => REG_DOUT <= conv_std_logic_vector(n_event(0), 16);
        when CONT_CH1  => REG_DOUT <= conv_std_logic_vector(n_event(1), 16);
        when CONT_CH2  => REG_DOUT <= conv_std_logic_vector(n_event(2), 16);
        when CONT_CH3  => REG_DOUT <= conv_std_logic_vector(n_event(3), 16);
        when CONT_CH4  => REG_DOUT <= conv_std_logic_vector(n_event(4), 16);
        when CONT_CH5  => REG_DOUT <= conv_std_logic_vector(n_event(5), 16);
        when CONT_CH6  => REG_DOUT <= conv_std_logic_vector(n_event(6), 16);
        when CONT_CH7  => REG_DOUT <= conv_std_logic_vector(n_event(7), 16);
        when CONT_CH8  => REG_DOUT <= conv_std_logic_vector(n_event(8), 16);
        when CONT_CH9  => REG_DOUT <= conv_std_logic_vector(n_event(9), 16);
        when CONT_CH10  => REG_DOUT <= conv_std_logic_vector(n_event(10), 16);
        when CONT_CH11  => REG_DOUT <= conv_std_logic_vector(n_event(11), 16);
        when CONT_CH12  => REG_DOUT <= conv_std_logic_vector(n_event(12), 16);
        when CONT_CH13  => REG_DOUT <= conv_std_logic_vector(n_event(13), 16);
        when CONT_CH14  => REG_DOUT <= conv_std_logic_vector(n_event(14), 16);
        when CONT_CH15  => REG_DOUT <= conv_std_logic_vector(n_event(15), 16);

        when COINC0  => REG_DOUT <= conv_std_logic_vector(n_event(16), 16);
        when COINC1  => REG_DOUT <= conv_std_logic_vector(n_event(17), 16);
        when COINC2  => REG_DOUT <= conv_std_logic_vector(n_event(18), 16);
        when COINC3  => REG_DOUT <= conv_std_logic_vector(n_event(19), 16);
        when COINC4  => REG_DOUT <= conv_std_logic_vector(n_event(20), 16);
        when COINC5  => REG_DOUT <= conv_std_logic_vector(n_event(21), 16);
        when COINC6  => REG_DOUT <= conv_std_logic_vector(n_event(22), 16);
        when COINC7  => REG_DOUT <= conv_std_logic_vector(n_event(23), 16);
        when COINC8  => REG_DOUT <= conv_std_logic_vector(n_event(24), 16);
```

```
            when COINC9   => REG_DOUT <= conv_std_logic_vector(n_event(25), 16);
            when COINC10  => REG_DOUT <= conv_std_logic_vector(n_event(26), 16);
            when COINC11  => REG_DOUT <= conv_std_logic_vector(n_event(27), 16);
            when COINC12  => REG_DOUT <= conv_std_logic_vector(n_event(28), 16);
            when COINC13  => REG_DOUT <= conv_std_logic_vector(n_event(29), 16);
            when COINC14  => REG_DOUT <= conv_std_logic_vector(n_event(30), 16);
            when COINC15  => REG_DOUT <= conv_std_logic_vector(n_event(31), 16);
            when COINC16  => REG_DOUT <= conv_std_logic_vector(n_event(32), 16);
            when COINC17  => REG_DOUT <= conv_std_logic_vector(n_event(33), 16);
            when COINC18  => REG_DOUT <= conv_std_logic_vector(n_event(34), 16);
            when COINC19  => REG_DOUT <= conv_std_logic_vector(n_event(35), 16);
            when COINC20  => REG_DOUT <= conv_std_logic_vector(n_event(36), 16);

            -- Registradores de leitura e escrita:
            when A_TAQUIS      => REG_DOUT <= conv_std_logic_vector(N_TAQUIS ,
16);
            when A_TGATE       => REG_DOUT <= conv_std_logic_vector(N_GATE    ,
16);
            when A_PATCH_NIM   => REG_DOUT <= conv_std_logic_vector(PATCH_NIM,
16);
            when A_U_ID        => REG_DOUT <= conv_std_logic_vector(U_ID      ,
16);
            when A_CTL_ANAL    => REG_DOUT <= CTL_ANALISE;
            when out_signal_0 => REG_DOUT <= N_out_signal_0;
            when out_signal_1 => REG_DOUT <= N_out_signal_1;
            -- Outros registradores:
            when others        => REG_DOUT <= (others => '0');
          end case;
        end if;
      end if;
    end process;

-- WRITE REGISTERS
    P_WREG : process(LCLK, nLBRES)
      begin
      if (nLBRES = '0') then
        N_TAQUIS      <= C_TAQUIS; -- Valor default para o tempo de aquisicao
        N_GATE        <= C_GATE;   -- Valor default para o tempo de aquisicao
        --    N_out_signal_0 <= C_out_signal_0;
        --    N_out_signal_1 <= C_out_signal_1;
        PATCH_NIM     <= 0;
```

```
  CTL_ANALISE  <= C_CTL_ANAL;
  elsif LCLK'event and LCLK = '1' then
    if (REG_WREN = '1') and (USR_ACCESS = '1') then
      case REG_ADDR is
        when A_TAQUIS     => N_TAQUIS     <= unsigned(REG_DIN);
        when A_TGATE      => N_GATE       <= unsigned(REG_DIN);
        when A_PATCH_NIM => PATCH_NIM    <= conv_integer(REG_DIN);
        when A_CTL_ANAL   => CTL_ANALISE <= REG_DIN;
        when out_signal_0  => N_out_signal_0 <= REG_DIN;
        when out_signal_1  => N_out_signal_1 <= REG_DIN;
        when others       => null;
      end case;
    end if;
  end if;
end process;



-- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-- XXXXXXXXX                                                 XXXXXXXXXXXXXXXXXX
-- XXXXXXXXX  LEITURA DE REGISTRADORES PELO  BUS VME  XXXXXXXXXXXXXXXXXX
-- XXXXXXXXX                                                 XXXXXXXXXXXXXXXXXX
-- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-- XXXXXXXXX              FIM DO CODIGO  VHDL              XXXXXXXXXXXXXXXXXX
-- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX



-- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-- XXXXXXXXX                                                 XXXXXXXXXXXXXXXXXX
-- XXXXXXXXX              COMUNICACAO SERIAL                 XXXXXXXXXXXXXXXXXX
-- XXXXXXXXX                                                 XXXXXXXXXXXXXXXXXX
-- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-- XXXXXXXXX            INICIO DO CODIGO  VHDL             XXXXXXXXXXXXXXXXXX
-- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX



process (FLAG_AQ)
  begin


  if (FLAG_AQ'event)and (FLAG_AQ = '1') then
```

```
 -- ASCII "{" (Inicio da TX)
DataSend(0)<= conv_unsigned(x"7B", N_NIB*2);
-- ASCII "}" (Fim do BYTE)
DataSend(N_DATASEND-1)<= conv_unsigned(x"7D", N_NIB*2);
for n in (N_NIB-1) downto 0 loop
  case U_ID(((n*4)+3) downto (n*4)) is
    when "0000" => DataSend(3-n+1) <= "00110000";  -- ASCII "0"
    when "0001" => DataSend(3-n+1) <= "00110001";  -- ASCII "1"
    when "0010" => DataSend(3-n+1) <= "00110010";  -- ASCII "2"
    when "0011" => DataSend(3-n+1) <= "00110011";  -- ASCII "3"
    when "0100" => DataSend(3-n+1) <= "00110100";  -- ASCII "4"
    when "0101" => DataSend(3-n+1) <= "00110101";  -- ASCII "5"
    when "0110" => DataSend(3-n+1) <= "00110110";  -- ASCII "6"
    when "0111" => DataSend(3-n+1) <= "00110111";  -- ASCII "7"
    when "1000" => DataSend(3-n+1) <= "00111000";  -- ASCII "8"
    when "1001" => DataSend(3-n+1) <= "00111001";  -- ASCII "9"
    when "1010" => DataSend(3-n+1) <= "01000001";  -- ASCII "A"
    when "1011" => DataSend(3-n+1) <= "01000010";  -- ASCII "B"
    when "1100" => DataSend(3-n+1) <= "01000011";  -- ASCII "C"
    when "1101" => DataSend(3-n+1) <= "01000100";  -- ASCII "D"
    when "1110" => DataSend(3-n+1) <= "01000101";  -- ASCII "E"
    when "1111" => DataSend(3-n+1) <= "01000110";  -- ASCII "F"
    when others => DataSend(3-n+1) <= "00000000";  -- ASCII null
  end case;
end loop;

for i in 0 to (N_CANAIS - 1) loop
  -- ASCII " " (separa canais)
  DataSend(i*5+5)<= conv_unsigned(x"20", N_NIB*2);

  for n in (N_NIB-1) downto 0 loop
    case n_event(i)(((n*4)+3) downto (n*4)) is
      when "0000" => DataSend(i*5+6+3-n) <= "00110000";  -- ASCII "0"
      when "0001" => DataSend(i*5+6+3-n) <= "00110001";  -- ASCII "1"
      when "0010" => DataSend(i*5+6+3-n) <= "00110010";  -- ASCII "2"
      when "0011" => DataSend(i*5+6+3-n) <= "00110011";  -- ASCII "3"
      when "0100" => DataSend(i*5+6+3-n) <= "00110100";  -- ASCII "4"
      when "0101" => DataSend(i*5+6+3-n) <= "00110101";  -- ASCII "5"
      when "0110" => DataSend(i*5+6+3-n) <= "00110110";  -- ASCII "6"
      when "0111" => DataSend(i*5+6+3-n) <= "00110111";  -- ASCII "7"
      when "1000" => DataSend(i*5+6+3-n) <= "00111000";  -- ASCII "8"
```

```
            when "1001" => DataSend(i*5+6+3-n) <= "00111001";  -- ASCII "9"
            when "1010" => DataSend(i*5+6+3-n) <= "01000001";  -- ASCII "A"
            when "1011" => DataSend(i*5+6+3-n) <= "01000010";  -- ASCII "B"
            when "1100" => DataSend(i*5+6+3-n) <= "01000011";  -- ASCII "C"
            when "1101" => DataSend(i*5+6+3-n) <= "01000100";  -- ASCII "D"
            when "1110" => DataSend(i*5+6+3-n) <= "01000101";  -- ASCII "E"
            when "1111" => DataSend(i*5+6+3-n) <= "01000110";  -- ASCII "F"
            when others => DataSend(i*5+6+3-n) <= "00000000";  -- ASCII null
          end case;
        end loop;
      end loop;
    end if;

end process;



process (Busy,Go,aux_serial,ByteIn,FLAG_AQ,aux_serial)
  begin

  If (FLAG_AQ'event) and (FLAG_AQ= '0') then
    envia_serial <= NOT envia_serial;
  end if;

  If (Busy'event) and (Busy = '0') then
    if (aux_serial < N_DATASEND) then
    aux_serial <= aux_serial+1;
    else
      aux_serial<=0;
      flag_serial <= NOT flag_serial;
    end if;
  end if;

  ByteIn <= conv_std_logic_vector(DataSend(aux_serial),8);

  if (ByteIn = conv_std_logic_vector(DataSend(aux_serial),8))
  and ((envia_serial XOR flag_serial )= '0' ) and (Busy ='0') then
    Go<='1';

  else
    Go<='0';
```

```
  end if;

end process;

process (LCLK)
  begin
  if rising_edge(LCLK) then
    Go_Antes <= Go;

    case Estado is
      ------------------------------------------------------------------
      when Idle =>
      Rx_Out <= '1';
      if (((Go = '1') and(Go_Antes = '0')) or Byte_em_Espera) then
        Byte_em_Espera <= false;
        outBuff <= '1' & ByteIn (7 downto 0) & '0'; -- stop + byte + start
        cont <= 0;
        bits_Sent <= 0;
        Estado <= Sending;
        Busy <= '1';
      end if;
      ------------------------------------------------------------------
      when Sending =>
      Rx_Out <= outBuff(0);

      if ((Go = '1')and(Go_Antes = '0')) then Byte_em_Espera <= true;
      end if;
      if (bits_Sent<10) then
        if (cont<4167) then -- 40(MHZ) / 9600(bps)
        cont<=cont+1;
        else
          cont <= 0;
          bits_Sent <= bits_Sent+1;
          outBuff <= '0' & outBuff(9 downto 1);
        end if;
        else
          Estado <= Idle;
          Busy <= '0';
        end if;
        ------------------------------------------------------------------
        when others =>
```

```
            Estado <= Idle;
        end case;
      end if;
    end process;

    -- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    -- XXXXXXXXX                                                   XXXXXXXXXXXXXXXXXX
    -- XXXXXXXXX                COMUNICACAO SERIAL                 XXXXXXXXXXXXXXXXXX
    -- XXXXXXXXX                                                   XXXXXXXXXXXXXXXXXX
    -- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    -- XXXXXXXXX                FIM DO CODIGO  VHDL                XXXXXXXXXXXXXXXXXX
    -- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
END rtl;
```

Apêndice G

Firmware do Microcontrolador do Sistema de Armazenamento e Envio dos Dados Via *TCP/IP*

```
////////////////////////////////////////////////////////////////
// Código do Sistema de Armazenamento e Envio dos Dados Via TCP/IP
// Author: Leonardo Chaves Ruiz Guedes (leocrg@cbpf.br)
////////////////////////////////////////////////////////////////


#include <DS1307RTC.h>
#include <Wire.h>
#include <Time.h>
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#include <SFE_BMP180.h>
#define FileNameStore "Fil_nam.dat"
#define FileNameRegisters "Reg_data.dat"
#define TesteEnable true
#define DataPrintEnable false
int DataToSend[40]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
    16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
    34, 35, 36, 37, 38, 39};
int stepSendData=1;
int maxDataSend=10000;
```

```
unsigned long currentMillis;
unsigned long previousMillis = 0;
unsigned long interval_tryToConnect = 10000;  //Espera 10s antes de conectar
    a primeira vez
unsigned long currentMillis2;
unsigned long previousMillis2 = 0;
unsigned long interval_tryToConnect2 = 5000;  //Tenta conectar de 5 em 5s
   enquanto cntrlsenddata estiver LOW

boolean DataSaveEnable = false;
boolean DataSaveToSend = false;
String inputString = "";
String inputString2 = "";
String DataFile = "";
String old_DataFile = "";
String old_DataFile2 = "";
String DataRegister = "";
boolean stringComplete = false;  // whether the string is complete
boolean stringComplete2 = false;
boolean enviaserial = false;
boolean Data_done=true;
boolean Bit_savedata=LOW;
boolean FirstFile=false;
//Strings do Datalogger e inteiros para configuração do RTC
String hora;
int int_hora;
String minuto;
int int_minuto;
String segundo;
int int_segundo;
String dia;
int int_dia;
String mes;
int int_mes;
String ano;
int int_ano;
int countString = 0;
int IndinChar1=0,
//Declarar variável de configuração do RTC
tmElements_t tm;
//Recebe a string de tempo do Datalogger
```

```
void serialEvent2() {
  while (Serial2.available()) {
    // get the new byte:
    char inChar = Serial2.read();
    //Estrutura para associar as variáveis de tempo com a string recebida
    if (inChar == ':' && countString == 0) {
      hora = inputString2;
      int_hora = hora.toInt();
      inputString2 = "";
      countString = 1;
    }
    else if (inChar == ':' && countString == 1) {
      minuto = inputString2;
      int_minuto = minuto.toInt();
      inputString2 = "";
      countString = 2;
    }
    else if (inChar == ' ') {
      segundo = inputString2;
      int_segundo = segundo.toInt();
      inputString2 = "";
      countString = 3;
    }
    else if (inChar == '/' && countString == 3) {
      dia = inputString2;
      int_dia = dia.toInt();
      inputString2 = "";
      countString = 4;
    }
    else if (inChar == '/' && countString == 4) {
      mes = inputString2;
      int_mes = mes.toInt();
      inputString2 = "";
      countString = 4;
    }
    else if (inChar == '\n') {
      ano = inputString2;
      int_ano = ano.toInt()+2000;
      inputString2 = "";
      countString = 0;
      stringComplete2 = true;
```

```
    }
    else {
      // add it to the inputString2:
      inputString2 += inChar;
    }
  }
}

//BMP180 settings//////////////////
SFE_BMP180 pressure;
#define ALTITUDE 1270.0 // Altitude of Criosfera1 in meters
boolean BMP180_ok = false;
void config_BMP180(void)
{
  if (pressure.begin()){
    BMP180_ok = true;
  }
  else
  {
    if (TesteEnable) Serial.println("BMP180 init fail\n\n");
  }
}
float read_temp(void){
  char status;
  double T=0;
  status = pressure.startTemperature();
  if (status != 0)
  {
    delay(status);
    status = pressure.getTemperature(T);
    if ((status = 0)&&(TesteEnable)) Serial.println("error retrieving
   temperature measurement\n");
  }
  else if (TesteEnable) Serial.println("error starting temperature
   measurement\n");
  return T;
}
double read_pressure(double T){
  char status;
  double P=0;
  status = pressure.startPressure(3);
```

```
  if (status != 0)
  {
    delay(status);
    status = pressure.getPressure(P,T);
    if ((status = 0)&&(TesteEnable)) Serial.println("error retrieving
    pressure measurement\n");
  }
  else if (TesteEnable) Serial.println("error starting pressure measurement\
    n");
  return P;
}
//MAG3110 settings////////////////
#define MAG_ADDR  0x0E //7-bit address for the MAG3110
void config_Mag3110(void)
{
  Wire.beginTransmission(MAG_ADDR); // transmit to device 0x0E
  Wire.write(0x11);              // cntrl register2
  Wire.write(0x80);              // write 0x80, enable auto resets
  Wire.endTransmission();        // stop transmitting
  delay(15);
  Wire.beginTransmission(MAG_ADDR); // transmit to device 0x0E
  Wire.write(0x10);              // cntrl register1
  Wire.write(1);                 // write 0x01, active mode
  Wire.endTransmission();        // stop transmitting
}
int mag_read_register(int reg)
{
  int reg_val;
  Wire.beginTransmission(MAG_ADDR); // transmit to device 0x0E
  Wire.write(reg);               // x MSB reg
  Wire.endTransmission();        // stop transmitting
  delayMicroseconds(2); //needs at least 1.3us free time between start and
    stop
  Wire.requestFrom(MAG_ADDR, 1); // request 1 byte
  while(Wire.available())    // slave may write less than requested
  {
    reg_val = Wire.read(); // read the byte
  }
  return reg_val;
}
int mag_read_value(int msb_reg, int lsb_reg)
```

```
{
  int val_low, val_high;  //define the MSB and LSB
  val_high = mag_read_register(msb_reg);
  delayMicroseconds(2); //needs at least 1.3us free time between start and
   stop
  val_low = mag_read_register(lsb_reg);
  int out = (val_low|(val_high << 8)); //concatenate the MSB and LSB
  return out;
}
int read_x(void)
{
  return mag_read_value(0x01, 0x02);
}
int read_y(void)
{
  return mag_read_value(0x03, 0x04);
}
int read_z(void)
{
  return mag_read_value(0x05, 0x06);
}
////////////Ethernet//////////////
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0F, 0x61, 0x3B };
/************ ETHERNET STUFF ************/
IPAddress server(###,###,###,###);
const int TCPport = #### ;
///////////////IRIDIUM///////////////
IPAddress ip(###,###,###,###);        //Define o endereco IP
IPAddress gateway(###,###,###,###);  //Define o gateway
IPAddress subnet(255, 255, 255, 0);     //Define a máscara de rede
EthernetClient client; //define 'client' as object
String data;
String data_log;
boolean FlagClientConnect = false;
//
  ------------------------------------------------------------------------


void(* resetFunc)(void) = 0;
char c;
/************ SDCARD STUFF ************/
Sd2Card card;
```

```
SdVolume volume;
SdFile root;
SdFile file;
// store error strings in flash to save RAM
#define error(s) error_P(PSTR(s))
void error_P(const char* str) {
  PgmPrint("error: ");
  SerialPrintln_P(str);
  if (card.errorCode()) {
    PgmPrint("SD error: ");
    if (TesteEnable) Serial.print(card.errorCode(), HEX);
    if (TesteEnable) Serial.print(',');
    if (TesteEnable) Serial.println(card.errorData(), HEX);
  }
  while (1);
}
File myFile;
boolean test_dife;
int contr_senddata = 3;
boolean sendDataDebug = false ;
int flag_savedata = 6;
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(9600);
  Wire.begin();
  pinMode(flag_savedata,OUTPUT);
  digitalWrite(flag_savedata, HIGH);
  pinMode(contr_senddata,INPUT_PULLUP);
  if (TesteEnable) Serial.println("Initializing SD card...");
  // On the Ethernet Shield, CS is pin 4. It's set as an output by default.
  // Note that even if it's not used as the CS pin, the hardware SS pin
  // (10 on most Arduino boards, 53 on the Mega) must be left as an output
  // or the SD library functions will not work.
  pinMode(4, OUTPUT);
  delay(100);
  int t = 0;
  int SDbegin;
  do {
    SDbegin = SD.begin(4);
    t = t + 1;
```

```
  if (TesteEnable) Serial.println(t);
  delay(100);
}while ((SDbegin == 0) && (t < 3));
if (!SDbegin) {
  if (TesteEnable) Serial.println("initialization failed!");
  } else {
  if (TesteEnable) Serial.println("initialization done.");
}
if(SD.exists(FileNameStore)){
  t = 0;
  do {
    myFile = SD.open(FileNameStore);
    t = t + 1;
    if (TesteEnable) Serial.println(t);
  }
  while ((myFile == 0) && (t < 3));
  if (myFile) {
    while (myFile.available()) {
      char IncharFile = myFile.read();
      if (IncharFile == ' ') {
        if (DataFile != "") {
          if (DataFile != old_DataFile2) {
            old_DataFile2 = DataFile;
          }
        }
        DataFile = "";
      }
      else if ((IncharFile != '\n') && ((int)IncharFile > 0)) {
        DataFile += IncharFile;
      }
    }
    myFile.close();
  }
  else {
    if (TesteEnable) Serial.println("error opening Fil_nam.dat");
  }
  delay(1000);
}
delay(500);
config_Mag3110();
delay(500);
```

```
  config_BMP180();
  DataSaveEnable = true;
}
void loop() {
  //Recebe string de tempo do Datalogger e configura o RTC
  if (stringComplete2) {
    String string_timeDatalogger = hora+":"+minuto+":"+segundo+" "+dia+"/"+
   mes+"/"+ano;
    tm.Hour = int_hora;
    tm.Minute = int_minuto;
    tm.Second = int_segundo;
    tm.Day = int_dia;
    tm.Month = int_mes;
    tm.Year = CalendarYrToTm(int_ano);
    RTC.write(tm);
    inputString2 = "";
    stringComplete2 = false;
    DataSaveToSend = true;
  }
  if((digitalRead(contr_senddata)==LOW)||(sendDataDebug)){
    currentMillis = millis();
    if((currentMillis - previousMillis) >= interval_tryToConnect) {
      previousMillis = currentMillis;
      while(((digitalRead(contr_senddata)==LOW)||(sendDataDebug))&& !
   Data_done) SendTCP();      //Contador inicial
    }
  }
  else{
    Data_done=false;
  }
}

void SendTCP(){
  if (TesteEnable) Serial.println("Trying to configure Ethernet using DHCP")
   ;
  if (Ethernet.begin(mac) == 0) {
    if (TesteEnable) Serial.println("Failed to configure Ethernet using DHCP
   ");
    delay(10000);
    }else{
    delay(1000);
```

```
if((digitalRead(contr_senddata)==LOW)||sendDataDebug){
  File myFile;
  ////le file_antigo///////
  int t = 0;
  do {
    myFile = SD.open(FileNameStore);
    t = t + 1;
    if (TesteEnable) Serial.println(t);
  }while ((myFile == 0) && (t < 10));

  if (myFile) {
    while (myFile.available()) {
      char IncharFile = myFile.read();
      if (IncharFile == ' ') {
        if (DataFile != "") {
          if (DataFile != old_DataFile2) {
            old_DataFile2 = DataFile;
          }
        }
        DataFile = "";
      }
      else if ((IncharFile != '\n') && ((int)IncharFile > 0)) {
        DataFile += IncharFile;
      }
    }
    myFile.close();
  }
  else {
    client.println("error opening Fil_nam.dat");
  }
  delay(1000);
  ////////////////////////

  t = 0;
  if (TesteEnable) Serial.println(old_DataFile2);
  char filename[30];
  strncpy(filename,old_DataFile2.c_str(),12);
  filename[12]='\0';
  if (TesteEnable) Serial.println(filename);
  do {
    myFile = SD.open(filename);
```

```
    t = t + 1;
    if (TesteEnable) Serial.println(t);
  }while ((myFile == 0) && (t < 10));
  t=0;
  int cont_Send_columms = 0;
  int cont_data_columms = 0;
  int cont_data_line = stepSendData-1;
  int cont_data_send = 0;

  boolean noStop = true;

  if (myFile) {
    data_log = "";
    while (myFile.available()&& ((digitalRead(contr_senddata)==LOW)||(
sendDataDebug))&&(cont_data_send < maxDataSend)&&(t<100)) {
      c = myFile.read();
      if ((c == '\t')||(c == ' ')) {
        if (cont_data_columms==DataToSend[cont_Send_columms])
cont_Send_columms++;
        cont_data_columms++;
      }
      if (cont_data_columms==DataToSend[cont_Send_columms]){
        if (( c > 0)) data_log += c;
      }
      if (c == '\n'){
        cont_data_columms=0;
        cont_Send_columms = 0;
        cont_data_line++;
        c=NULL;
        if (cont_data_line==stepSendData){
          cont_data_send++;
          t=0;
          noStop = true;
          while (digitalRead(contr_senddata)==LOW){
            currentMillis2 = millis();
            if(((currentMillis2 - previousMillis2) >=
interval_tryToConnect2)||noStop) {
              previousMillis2 = currentMillis2;
              noStop = false;
              if(SendTCPMessage(server,TCPport,data_log)) { break;}
            }
```

```
                serialEvent1();
              }
              cont_data_line=0;
            }
            data_log = "";
          }
        }
        c = ' ';
        Data_done=true;
        myFile.close();
        } else {
        if (TesteEnable) Serial.print("error opening ");
        if (TesteEnable) Serial.println(old_DataFile2);
      }
    }
  }
  sendDataDebug = false ;
}
// Funcao de ennvio de dados via TCP
boolean SendTCPMessage(IPAddress IPToSend, unsigned int PortToCommunicate,
    String  MessageToSend){
  boolean TCPStatus=false ;
  EthernetClient client;
  if(client.connected()){
    client.println(MessageToSend);
    TCPStatus = true ;
    } else if (client.connect(IPToSend, PortToCommunicate)) {
    client.println(MessageToSend);
    if (TesteEnable) Serial.println(MessageToSend);
    TCPStatus = true ;
  }else if (TesteEnable) Serial.println("Can't connect to TCP!");
  client.stop();
  return TCPStatus;
}
void serialEvent() { //Debug
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    switch(inChar){
      case '&':
      sendDataDebug = !sendDataDebug ;
      break ;
```

```
    }
  }

}
////////////Serial to SD//////////////
void serialEvent1() {
  double T,P;
  int Bx,By,Bz;
  String Time_str;
  String Date_str;
  String str_Day;
  String str_Month;
  String str_Year;
  String str_Second;
  String str_Minute;
  String str_Hour;
  int t=0;
  if (DataSaveEnable){
    while (Serial1.available()) {
      char inChar1 = (char)Serial1.read();
      if ((DataPrintEnable)&&((enviaserial)||(inChar1=='{'))&&(inChar1>0)){
        if (TesteEnable) Serial.print(inChar1);
      }
      if (inChar1 == ' ') inChar1 = '\t';
      if (inChar1 == '}') {
        old_DataFile = DataFile;
        DataFile = "";
        RTC.read(tm);
        if (tm.Day<10)str_Day = "0" + String(tm.Day);
        else str_Day = String(tm.Day);
        if (tm.Month<10)str_Month = "0" + String(tm.Month);
        else str_Month = String(tm.Month);
        if ((tmYearToCalendar(tm.Year)-2000)<10)str_Year = "0" + String(
   tmYearToCalendar(tm.Year)-2000);
        else str_Year = String(tmYearToCalendar(tm.Year)-2000);
        if (tm.Hour<10)str_Hour = "0" + String(tm.Hour);
        else str_Hour = String(tm.Hour);
        if (tm.Minute<10)str_Minute = "0" + String(tm.Minute);
        else str_Minute = String(tm.Minute);
        if (tm.Second<10)str_Second = "0" + String(tm.Second);
        else str_Second = String(tm.Second);
```

```
    Time_str = str_Hour + ":" + str_Minute + ":" + str_Second;
    Date_str = str_Day + "_" + str_Month + "_" + str_Year;
    DataFile =  Date_str + ".dat";

    if (DataFile.length()>12) DataFile = "bugdata.dat";

    if (TesteEnable) Serial.print("\n");

    if ( !compare_str(old_DataFile.c_str(),DataFile.c_str(),12)&&
DataSaveToSend) {
      t = 0;
      FirstFile=true;
      do {
        myFile = SD.open(FileNameStore, FILE_WRITE);

        t++;
      } while ((myFile == 0) && (t < 3));

      if (myFile) {
        myFile.print(" ");
        myFile.println(DataFile.c_str());
        myFile.close();
      }
    }
    if (TesteEnable) {
      t = 0;
      do {
        myFile = SD.open(FileNameStore);
        t = t + 1;
      } while ((myFile == 0) && (t < 3));

      if (myFile) {
        while (myFile.available()) {
          Serial.write(myFile.read());
        }
        myFile.close();
        } else {
        if (TesteEnable) Serial.print("error opening " );
        if (TesteEnable) Serial.println(DataFile.c_str());
      }
```

```
    }

    delay(20);
    T=read_temp();
    if (TesteEnable) Serial.print("temperatura: ");
    if (TesteEnable) Serial.println(T);
    delay(20);
    P=read_pressure(T);
    if (TesteEnable) Serial.print("Pressao: ");
    if (TesteEnable) Serial.println(P);
    delay(20);
    Bx=read_x();
    if (TesteEnable) Serial.print("Campo x: ");
    if (TesteEnable) Serial.println(Bx);
    delay(20);
    By=read_y();
    if (TesteEnable) Serial.print("Campo y: ");
    if (TesteEnable) Serial.println(By);
    delay(20);
    Bz=read_z();
    if (TesteEnable) Serial.print("Campo z: ");
    if (TesteEnable) Serial.println(Bz);
    inputString = Date_str + "_" + Time_str + "\t" + T + "\t" + P + "\t"
 + Bx + "\t" + By + "\t" + Bz + "\t" + inputString;
    if (DataSaveToSend){
      t=0;
      do {
        myFile = SD.open(DataFile.c_str(), FILE_WRITE);
        t++;
        if (TesteEnable) Serial.println(t);
      } while ((myFile == 0) && (t < 3));
      if (myFile) {
        if (FirstFile) {
          FirstFile = false;
          myFile.println("Date\tTemp\tPressure\tBx\tBy\tBz\tUNID\tCH0\
tCH1\tCH2\tCH3\tCH4\tCH5\tCH6\tCH7\tCH8\tCH9\tCH10\tCH11\tCH12\tCH13\
tCH14\tCH15\tA_0_1\tA_3_4\tA_5_6\tB_0_1\tB_3_4\tB_5_6\tA_0_1&A_2&A_3_4&
A_5_6\tA_0_1&A_3_4&A_5_6\tA_0_1&A_2&A_3_4\tA_0_1&A_2&A_5_6\tA_2&A_3_4&
A_5_6\tB_0_1&B_2&B_3_4&B_5_6\tB_0_1&B_3_4&B_5_6\tB_0_1&B_2&B_3_4\tB_0_1&
B_2&B_5_6\tB_2&B_3_4&B_5_6\tTOTAL");
        }
```

```
      if (TesteEnable) Serial.println(inputString);
      myFile.println(inputString);
      myFile.close();
      } else {
      // if the file didn't open, print an error:
      if (TesteEnable) Serial.print("error opening " );
      if (TesteEnable) Serial.println(DataFile.c_str());
    }
  }

  t=0;
  do {
    t++;
    myFile = SD.open("raw_data.dat", FILE_WRITE);
    //            if (TesteEnable) Serial.println(t);
  } while ((myFile == 0) && (t < 3));

  if (myFile) {
    myFile.println(inputString);
    Bit_savedata = !Bit_savedata;
    digitalWrite(flag_savedata, Bit_savedata);
    myFile.close();
    } else {
    // if the file didn't open, print an error:
    if (TesteEnable) Serial.print("error opening " );
    if (TesteEnable) Serial.println("raw_data.dat");
  }

  if (TesteEnable) Serial.print("\r\n");
  enviaserial = false;
  IndinChar1=0;
}
if (enviaserial == true) {
  inputString += inChar1;
  IndinChar1++;
}
if (inChar1 == '{') {
  inputString = "";
  enviaserial = true;
}
if (inChar1 == '\n') {
```

```
        stringComplete = true;
      }
    }
  }
}
boolean compare_str(const char a[], const char b[],int n_Ind){
  boolean match = false;
  if ( a[0] != NULL )
  match = true;
  for ( int k = 0; k < n_Ind; k++ ) {
    if ( a[k] != b[k] )
    match = false;
  }
  if ( match ) {
    return true;
  }
  else  {
    return false;
  }
}
String HexToBin(char HexChar){

  switch(HexChar){
    case '0': return "0000"; break;
    case '1': return "0001"; break;
    case '2': return "0010"; break;
    case '3': return "0011"; break;
    case '4': return "0100"; break;
    case '5': return "0101"; break;
    case '6': return "0110"; break;
    case '7': return "0111"; break;
    case '8': return "1000"; break;
    case '9': return "1001"; break;
    case 'A': return "1010"; break;
    case 'B': return "1011"; break;
    case 'C': return "1100"; break;
    case 'D': return "1101"; break;
    case 'E': return "1110"; break;
    case 'F': return "1111"; break;
    default: return ""; break;
  }
```

```
}
```

Apêndice H

Firmware de Controle e Aquisição de Dados do Detector CREAT3

A firmware completa é baseada na original da *Evaluation Board OMEGA MAROC3*. Neste Apêndice serão apresentados os blocos criados exclusivamente para o detector CREAT3.

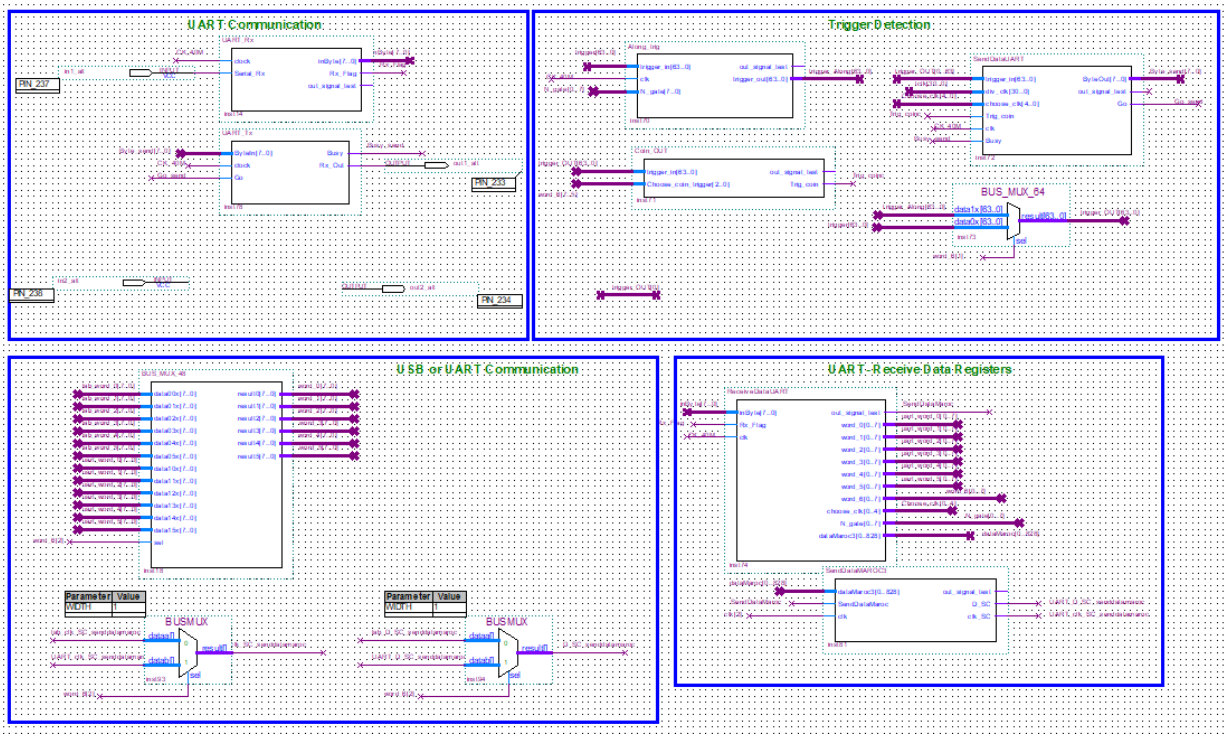


Figura H.1: Diagrama de blocos principal da firmware

Código VHDL referente ao bloco de comunicação UART_RX.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.numeric_std.ALL;

entity UART_Rx is

  Port ( clock : in  STD_LOGIC;
    Serial_Rx : in  STD_LOGIC;
    inByte : out  STD_LOGIC_VECTOR (7 downto 0);
    Rx_Flag : out  STD_LOGIC;
  out_signal_test : out  STD_LOGIC);

end UART_Rx;

architecture Behavioral of UART_Rx is
  constant ClockValue : integer:=40000000;
  constant BaudRate : integer:=115200;

  signal cont : integer range 0 to ((ClockValue/BaudRate)+(ClockValue/BaudRate)/2):= 0;
--6250 16000 observar o limite ao mudar o clock / baud rate
  signal bitsRecebidos : integer range 0 to 8:=0;

  signal inBits : std_logic_vector  (7 downto 0);
  signal Estado : std_logic_vector  (1 downto 0):="00";

  signal flag_Rx_1: std_logic:='0';
  signal flag_Rx_2: std_logic:='0';


begin


  process (Serial_Rx)
    begin
    if (falling_edge(Serial_Rx) and ((flag_Rx_1 XOR flag_Rx_2)='0'))then
      flag_Rx_1<= Not flag_Rx_1;
    end if;
  end process;
```

```
UART_Rx : process (clock)
  begin
  if rising_edge(clock) then
    case Estado is
      ------------------------------------------------------------------
      when "00" =>
      if ((flag_Rx_1 XOR flag_Rx_2)='1') then
        Estado <= "01";
        cont<=0;
      end if;
      ------------------------------------------------------------------
      when "01" =>
      -- 6250 ex: 15417 = Clock(100Mhz) / BaudRate * 1.5
      if (cont<((ClockValue/BaudRate)+(ClockValue/BaudRate)/2)) then
      cont<=cont+1;
      else
        bitsRecebidos<=0;
        Estado <= "10";
      end if;
      ------------------------------------------------------------------
      when "10" =>
      if (bitsRecebidos<8) then
        --4167 ex: 10417 = Clock(100MHz) / BaudRate
        if (cont<ClockValue/BaudRate) then
        cont <= cont+1;
        else
          cont <=0;
          bitsRecebidos <= bitsRecebidos+1;
          inBits <= Serial_Rx & inBits(7 downto 1);
        end if;
        else
          inByte <= inBits;
          Rx_Flag <= '1';
          Estado <= "11";
        end if;
        ------------------------------------------------------------------
        when "11" =>
        -- 4167 ex: 10417 = Clock(100MHz) / BaudRate
        if (cont<ClockValue/BaudRate) then
```

```
            cont<=cont+1;
            else
              flag_Rx_2 <= NOT flag_Rx_2;
              cont<=0;
              Rx_Flag <= '0';
              Estado <= "00";
            end if;
            ------------------------------------------------------------------

            when others =>
            Estado <= "00";
        end case;
      end if;
    end process;

end Behavioral;
```

Código VHDL referente ao bloco de comunicação UART_TX.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity UART_Tx is
Port ( ByteIn : in  STD_LOGIC_VECTOR (7 downto 0);
     clock : in  STD_LOGIC;
     Go : in  STD_LOGIC;
     Busy : out  STD_LOGIC;
   Rx_Out : out  STD_LOGIC);
end UART_Tx;

architecture Behavioral of UART_Tx is
  constant ClockValue : integer:=40000000;
  constant BaudRate : integer:=115200;--9600;
  signal outBuff : std_logic_vector (9 downto 0);  -- envia do zero ao 7 do byte
  signal Go_Antes : std_logic:='0';
  -- 4167 observar o limite ao mudar o clock / baud rate
  signal cont : integer range 0 to (ClockValue/BaudRate):=0;
  signal bits_Sent : integer range 0 to 10:=0;
  signal Estado : std_logic:='0';
signal Byte_em_Espera : boolean := false;
begin

  UART_Tx : process (clock)
```

```
    begin
    if rising_edge(clock) then
      Go_Antes <= Go;

      case Estado is
        ------------------------------------------------------------------
        when '0' =>
        Rx_Out <= '1';
        if ((Go = '1')and(Go_Antes = '0')) or Byte_em_Espera  then
          Byte_em_Espera <= false;
          outBuff <= '1' & ByteIn (7 downto 0) & '0'; -- stop + byte + start
          cont <= 0;
          bits_Sent <= 0;
          Estado <= '1';
          Busy <= '1';
        end if;
        ------------------------------------------------------------------
        when '1' =>
        Rx_Out <= outBuff(0);
        if ((Go = '1')and(Go_Antes = '0')) then Byte_em_Espera <= true;
        end if;
        if (bits_Sent<10) then
          if (cont<(ClockValue/BaudRate)) then --4167 40Mhz / 9600
          cont<=cont+1;
          else
            cont <= 0;
            bits_Sent <= bits_Sent+1;
            outBuff <= '0' & outBuff(9 downto 1);
          end if;
          else
            Estado <= '0';
            Busy <= '0';
          end if;
          ------------------------------------------------------------------
          when others =>
          Estado <= '0';
      end case;
    end if;
  end process;

end Behavioral;
```

Código VHDL referente ao bloco de alongamento dos sinais de *trigger*.

```
----------------------------------------------------------------------------------
-- Bloco de Coincidencia e tratamento dos triggers

----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;

entity Along_trig is
Port ( trigger_in : in  STD_LOGIC_VECTOR (63 downto 0);
    clk :  IN  STD_LOGIC;
    N_gate : in  STD_LOGIC_VECTOR (7 downto 0);
    out_signal_test : out  STD_LOGIC;
    trigger_out : OUT  STD_LOGIC_VECTOR (63 downto 0));
end Along_trig;

architecture Behavioral of Along_trig is
  constant N_CH    :integer := 64;
  constant N_NIB   :integer := 4;
  type std_logic_array is array(N_CH-1 downto 0)  of integer range 0 to 255;
  signal flag_trig_1     :std_logic_vector(N_CH-1 downto 0):=(others=>'0');
  signal flag_trig_2     :std_logic_vector(N_CH-1 downto 0):=(others=>'0');
  signal estado          :std_logic_vector(N_CH-1 downto 0):=(others=>'0');
  signal cont     :std_logic_array;
  signal T_gate   :integer range 0 to 255;

begin
  T_gate <= conv_integer(N_gate);

  ---------Alongamento do sinal dos triggers
  Process(trigger_in)
    begin
    for i in 0 to (N_CH - 1) loop
      if (rising_edge(trigger_in(i)) and ((flag_trig_1(i) XOR flag_trig_2(i))='0'))
      then flag_trig_1(i)<= Not flag_trig_1(i);
      end if;
    end loop;
  end process;
```

```
  Process(clk)
    begin
    if rising_edge(clk)then
      for i in 0 to (N_CH  - 1) loop
        case estado(i) is
          when '0' =>
          if ((flag_trig_1(i) XOR flag_trig_2(i))='1') then
            estado(i)<='1';
          trigger_out(i)<='1';
          else
            trigger_out(i)<='0';
          end if;

          when '1' =>
          if cont(i)<T_gate-1 then
            cont(i)<=cont(i)+1;
          trigger_out(i)<='1';
          else
            cont(i)<=0;
            estado(i)<='0';
            trigger_out(i)<='0';
            flag_trig_2(i)<= Not flag_trig_2(i);
          end if;
          when others =>
          Estado(i) <= '0';
        end case;
      end loop;
    end if;
  end process;

  ------------------

end Behavioral;
```

Código VHDL referente ao bloco de coincidência.

```
----------------------------------------------------------------------------------
-- Bloco de Coincidencia e tratamento dos triggers
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_unsigned.all;
```

```
USE ieee.std_logic_arith.all;

LIBRARY lpm;
USE lpm.lpm_components.all;

entity Coin_OUT is
Port ( trigger_in : in  STD_LOGIC_VECTOR (63 downto 0);
     out_signal_test : out  STD_LOGIC;
     Trig_coin : out  STD_LOGIC;
   Choose_coin_trigger : in STD_LOGIC_VECTOR (2 downto 0));


end Coin_OUT;

architecture Behavioral of Coin_OUT is
  constant N_CH     :integer := 64;

  ------------------------------------linha----col
  type in_ch_array     is array(0 to 7, 0 to 7)  of integer range 0 to 64;

  constant Coin_Map     :in_ch_array := (  ( 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 ),
                       ( 8 , 9 , 10  , 11  , 12  , 13  , 14  , 15  ),

                       ( 16  , 17  , 18  , 19  , 20  , 21  , 22  , 23  ),
                       ( 24  , 25  , 26  , 27  , 28  , 29  , 30  , 31  ),

                       ( 32  , 33  , 34  , 35  , 36  , 37  , 38  , 39  ),
                       ( 40  , 41  , 42  , 43  , 44  , 45  , 46  , 47  ),

                       ( 48  , 49  , 50  , 51  , 52  , 53  , 54  , 55  ),
                       ( 56  , 57  , 58  , 59  , 60  , 61  , 62  , 63  ));

  signal COIN       :std_logic_vector(N_CH-1 downto 0);

begin

  Process(trigger_in ,COIN, Choose_coin_trigger)
    begin
    for i in 0 to 3 loop

      ----------------Lógica coincidencia eixo xyz--------------------
```

```
  COIN(i)<=    ((trigger_in(coin_map(i*2,0)))
    OR  (trigger_in(coin_map(i*2,1)))
    OR  (trigger_in(coin_map(i*2,2)))
    OR  (trigger_in(coin_map(i*2,3)))
    OR  (trigger_in(coin_map(i*2,4)))
    OR  (trigger_in(coin_map(i*2,5)))
    OR  (trigger_in(coin_map(i*2,6)))
    OR  (trigger_in(coin_map(i*2,7))))
    AND((trigger_in(coin_map(i*2+1,0)))
    OR  (trigger_in(coin_map(i*2+1,1)))
    OR  (trigger_in(coin_map(i*2+1,2)))
    OR  (trigger_in(coin_map(i*2+1,3)))
    OR  (trigger_in(coin_map(i*2+1,4)))
    OR  (trigger_in(coin_map(i*2+1,5)))
    OR  (trigger_in(coin_map(i*2+1,6)))
    OR  (trigger_in(coin_map(i*2+1,7))));

end loop;

case Choose_coin_trigger is
  ------------------------------------------------------------------

  when "000" =>
  Trig_coin <= trigger_in(0) OR trigger_in(1) OR trigger_in(2) OR trigger_in(3) OR
  trigger_in(4) OR trigger_in(5) OR trigger_in(6) OR trigger_in(7)OR
  trigger_in(8) OR trigger_in(9) OR trigger_in(10) OR trigger_in(11)OR
  trigger_in(12)OR trigger_in(13)OR trigger_in(14)OR trigger_in(15)OR
  trigger_in(16)OR trigger_in(17)OR trigger_in(18)OR trigger_in(19)OR
  trigger_in(20)OR trigger_in(21)OR trigger_in(22)OR trigger_in(23)OR
  trigger_in(24) OR trigger_in(25)OR trigger_in(26)OR trigger_in(27)OR
  trigger_in(28)OR trigger_in(29)OR trigger_in(30)OR trigger_in(31)OR
  trigger_in(32) OR trigger_in(33)OR trigger_in(34)OR trigger_in(35)OR
  trigger_in(36)OR trigger_in(37)OR trigger_in(38)OR trigger_in(39)OR
  trigger_in(40)OR trigger_in(41)OR trigger_in(42)OR trigger_in(43)OR
  trigger_in(44)OR trigger_in(45)OR trigger_in(46)OR trigger_in(47)OR
  trigger_in(48)OR trigger_in(49)OR trigger_in(50)OR trigger_in(51)OR
  trigger_in(52)OR trigger_in(53)OR trigger_in(54)OR trigger_in(55)OR
  trigger_in(56)OR trigger_in(57)OR trigger_in(58)OR trigger_in(59)OR
  trigger_in(60)OR trigger_in(61)OR trigger_in(62)OR trigger_in(63);

  when "100" =>
  Trig_coin <= COIN(0) OR COIN(1) OR COIN(2) OR COIN(3);
```

```
        when "010" =>
        Trig_coin <= (COIN(0) AND COIN(1))
        OR  (COIN(0) AND COIN(2))
        OR (COIN(0) AND COIN(3))
        OR (COIN(1) AND COIN(2))
        OR (COIN(1) AND COIN(3))
        OR (COIN(2) AND COIN(3));

        when "110"=>
        Trig_coin <= (COIN(0) AND COIN(1) AND COIN(2))
        OR  (COIN(0) AND COIN(1)AND COIN(3))
        OR  (COIN(1) AND COIN(2)AND COIN(3));

        when others =>
        Trig_coin <= COIN(0) AND COIN(1) AND COIN(2) AND COIN(3);
    end case;
  end process;
end Behavioral;
```

Código VHDL referente ao bloco de recepção de mensagem via UART.

```
----------------------------------------------------------------------------------
-- Bloco de Coincidencia e tratamento dos triggers

----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;

entity ReceiveDataUART is
Port ( inByte : in  STD_LOGIC_VECTOR (7 downto 0);
    Rx_Flag : in  STD_LOGIC;
    clk :  IN  STD_LOGIC;
    out_signal_test : out  STD_LOGIC;
    word_0  : OUT STD_LOGIC_VECTOR (0 TO 7);
    word_1    : OUT STD_LOGIC_VECTOR (0 TO 7);
    word_2    : OUT STD_LOGIC_VECTOR (0 TO 7);
    word_3    : OUT STD_LOGIC_VECTOR (0 TO 7);
    word_4    : OUT STD_LOGIC_VECTOR (0 TO 7);
    word_5    : OUT STD_LOGIC_VECTOR (0 TO 7);
```

```
    word_6     : OUT STD_LOGIC_VECTOR (0 TO 7);
    choose_clk: OUT STD_LOGIC_VECTOR (0 TO 4);
    N_gate     : OUT STD_LOGIC_VECTOR (0 TO 7);
    dataMaroc3 : OUT  STD_LOGIC_VECTOR (0 to 828));


end ReceiveDataUART;

architecture Behavioral of ReceiveDataUART is
  constant N_CH     :integer := 64;
  constant N_NIB    :integer := 4;
  constant N_DATARECEIVE :integer := 48;
  constant N_DATASENDMAROC :integer := 829;
  constant N_Registers_Out :integer := 85;
  constant AD_MAROC_DATA :integer := 8;
  constant AD_MAROC_Ctest :integer := 8;
  constant AD_MAROC_GAIN :integer := 16;
  constant AD_MAROC_DAC0 :integer := 80;
  constant AD_MAROC_DAC1 :integer := 82;
  constant AD_MAROC_OTHERS :integer := 84;

  type std_logic_array is array(0 to N_Registers_Out-1)
  of STD_LOGIC_VECTOR ( 0 to N_NIB*2-1);

  signal n :integer range 0 to 2:= 2;
  signal i :integer range 0 to N_Registers_Out:= N_Registers_Out;
  signal cont :integer range 0 to 10:= 0;
  signal Registers_Out : std_logic_array :=
  (("10101000"),  -- word_0      ad0
  ("00111011"), -- word_1        ad1
  ("11000110"), -- word_2        ad2
  ("00100011"), -- word_3        ad3
  ("11011001"), -- word_4        ad4
  ("00000010"), -- word_5      ad5
  ("11000010"), -- word_6 NEW    ad6
  ("00001010"), -- N_gate NEW    ad7
  ("10000000"), -- Ctest_ch0-7     ad8
  ("10000000"),   -- Ctest_ch8-15    ad9
  ("01000000"),   -- Ctest_ch16-23   ad10
  ("01000000"),   -- Ctest_ch24-31   ad11
  ("00100000"),   -- Ctest_ch32-39   ad12
```

```
("00100000"),    -- Ctest_ch40-47   ad13
("00010000"),    -- Ctest_ch48-55   ad14
("00010000"),    -- Ctest_ch56-63   ad15
("01000000"), --  GAIN0      ad16
("01000000"), --  GAIN1      ad17
("01000000"), --  GAIN2      ad18
("01000000"), --  GAIN3      ad19
("01000000"), --  GAIN4      ad20
("01000000"), --  GAIN5      ad21
("01000000"), --  GAIN6      ad22
("01000000"), --  GAIN7      ad23
("01000000"), --  GAIN8      ad24
("01000000"), --  GAIN9      ad25
("01000000"), --  GAIN10       ad26
("01000000"), --  GAIN11       ad27
("01000000"), --  GAIN12       ad28
("01000000"), --  GAIN13       ad29
("01000000"), --  GAIN14       ad30
("01000000"), --  GAIN15       ad31
("01000000"), --  GAIN16       ad32
("01000000"), --  GAIN17       ad33
("01000000"), --  GAIN18       ad34
("01000000"), --  GAIN19       ad35
("01000000"), --  GAIN20       ad36
("01000000"), --  GAIN21       ad37
("01000000"), --  GAIN22       ad38
("01000000"), --  GAIN23       ad39
("01000000"), --  GAIN24       ad40
("01000000"), --  GAIN25       ad41
("01000000"), --  GAIN26       ad42
("01000000"), --  GAIN27       ad43
("01000000"), --  GAIN28       ad44
("01000000"), --  GAIN29       ad45
("01000000"), --  GAIN30       ad46
("01000000"), --  GAIN31       ad47
("01000000"), --  GAIN32       ad48
("01000000"), --  GAIN33       ad49
("01000000"), --  GAIN34       ad50
("01000000"), --  GAIN35       ad51
("01000000"), --  GAIN36       ad52
("01000000"), --  GAIN37       ad53
```

```
  ("01000000"), --  GAIN38       ad54
  ("01000000"), --  GAIN39       ad55
  ("01000000"), --  GAIN40       ad56
  ("01000000"), --  GAIN41       ad57
  ("01000000"), --  GAIN42       ad58
  ("01000000"), --  GAIN43       ad59
  ("01000000"), --  GAIN44       ad60
  ("01000000"), --  GAIN45       ad61
  ("01000000"), --  GAIN46       ad62
  ("01000000"), --  GAIN47       ad63
  ("01000000"), --  GAIN48       ad64
  ("01000000"), --  GAIN49       ad65
  ("01000000"), --  GAIN50       ad66
  ("01000000"), --  GAIN51       ad67
  ("01000000"), --  GAIN52       ad68
  ("01000000"), --  GAIN53       ad69
  ("01000000"), --  GAIN54       ad70
  ("01000000"), --  GAIN55       ad71
  ("01000000"), --  GAIN56       ad72
  ("01000000"), --  GAIN57       ad73
  ("01000000"), --  GAIN58       ad74
  ("01000000"), --  GAIN59       ad75
  ("01000000"), --  GAIN60       ad76
  ("01000000"), --  GAIN61       ad77
  ("01000000"), --  GAIN62       ad78
  ("01000000"), --  GAIN63       ad79
  ("00000000"), --  DAC0 MSB     ad80
  ("11111010"), --  DAC0 LSB     ad81
  ("00000000"), --  DAC1 MSB     ad82
  ("00000000"), --  DAC1 LSB     ad83
  ("00000000"));  --  OTHERS      ad84    [X X X X X X X "small_dac[7]"]

  signal dataMaroc3_Buff :std_LOGIC_VECTOR(0 to N_DATASENDMAROC-1):='1'&
   x"8003E8000000000000000000000000000000030B068872010080402010080402010
   0804020100804020100804020100804020100804020100804020100804020100804020
   0100804020100804020100804020100804020100804020100804F000000000000001";
  signal inByte_buff : STD_LOGIC_VECTOR (7 downto 0):="00000000";
  signal out_signal_test_buff :  STD_LOGIC:='0';
  signal Estado : std_logic_vector  (2 downto 0):="000";

begin
```

```
word_0(0 TO 7) <= Registers_Out(0)(0 TO 7);
word_1(0 TO 7) <= Registers_Out(1)(0 TO 7);
word_2(0 TO 7) <= Registers_Out(2)(0 TO 7);
word_3(0 TO 7) <= Registers_Out(3)(0 TO 7);
word_4(0 TO 7) <= Registers_Out(4)(0 TO 7);
word_5(0 TO 7) <= Registers_Out(5)(0 TO 7);
word_6(0 TO 7) <= Registers_Out(6)(0 TO 7);
N_gate(0 TO 7) <= Registers_Out(7)(0 TO 7);

dataMaroc3<=dataMaroc3_Buff;
out_signal_test<=out_signal_test_buff;

process (Rx_Flag)
  begin
  if (falling_edge(Rx_Flag)) then
    inByte_buff<=inByte;
  end if;
end process;


process (clk)
  begin
  if (falling_edge(clk))then
    case estado is
      when "000" =>
      if (Rx_Flag='1') then
        estado <= "001";
      end if;

      when "001" =>
      if (Rx_Flag='0') then
        estado <= "010";
      end if;

      when "010" =>
      case inByte is
        when "01100001" => i<=0; n<=0; estado <= "000";  -- ASCII "a" word_0
        when "01100010" => i<=1; n<=0; estado <= "000";  -- ASCII "b" word_1
        when "01100011" => i<=2; n<=0; estado <= "000";  -- ASCII "c" word_2
        when "01100100" => i<=3; n<=0; estado <= "000";  -- ASCII "d" word_3
```

```
when "01100101" => i<=4; n<=0; estado <= "000";  -- ASCII "e" word_4
when "01100110" => i<=5; n<=0; estado <= "000";  -- ASCII "f" word_5
when "01100111" => i<=6; n<=0; estado <= "000";  -- ASCII "g" word_6
when "01101000" => i<=7; n<=0; estado <= "000";  -- ASCII "h" N_Gate
when "01101001" => i<=AD_MAROC_GAIN;  n<=0; estado <= "000";
-- ASCII "i"  GAIN
when "01101010" => i<=AD_MAROC_DAC0;  n<=0; estado <= "000";
-- ASCII "j"  DAC0
when "01101011" => i<=AD_MAROC_DAC1;  n<=0; estado <= "000";
-- ASCII "k"  DAC1
when "01101100" => i<=AD_MAROC_OTHERS; n<=0; estado <= "000";
-- ASCII "l"  OTHERS
when "01111011" => i<=AD_MAROC_Ctest;   n<=0; estado <= "000";
-- ASCII "{" C_test
when "01111101" =>                                  -- ASCII "}" send to Maroc
i<=N_Registers_Out; estado <= "100";
------------------- Ctest Register ------------------------------
for j in 0 to 7 loop
  for k in 0 to 7 loop
    dataMaroc3_Buff(828-j*8-k) <= Registers_Out(j+AD_MAROC_Ctest)(k);
  end loop;
end loop;
---------------------- DAC0 Register ------------------------------
for j in 0 to 7 loop
  dataMaroc3_Buff(22-j) <= Registers_Out(AD_MAROC_DAC0+1)(7-j);
end loop;
for j in 0 to 1 loop
  dataMaroc3_Buff(14-j) <= Registers_Out(AD_MAROC_DAC0)(7-j);
end loop;
---------------------- DAC1 Register ------------------------------
for j in 0 to 7 loop
  dataMaroc3_Buff(12-j) <= Registers_Out(AD_MAROC_DAC1+1)(7-j);
end loop;
for j in 0 to 1 loop
  dataMaroc3_Buff(4-j) <= Registers_Out(AD_MAROC_DAC1)(7-j);
end loop;
---------------------- OTHERS Register ----------------------------
dataMaroc3_Buff(2) <= Registers_Out(AD_MAROC_OTHERS)(7); -- small_dac
for j in 0 to 4 loop
  choose_clk(4-j) <= Registers_Out(AD_MAROC_OTHERS)(j);
end loop;
```

```
            when "01011000" => estado <= "011";  -- ASCII "X" not change register
            when "00110000" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "0000"; end if; estado <= "011";
-- ASCII "0"
            when "00110001" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "0001"; end if; estado <= "011";
-- ASCII "1"
            when "00110010" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "0010"; end if; estado <= "011";
-- ASCII "2"
            when "00110011" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "0011"; end if; estado <= "011";
-- ASCII "3"
            when "00110100" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "0100"; end if; estado <= "011";
-- ASCII "4"
            when "00110101" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "0101"; end if; estado <= "011";
-- ASCII "5"
            when "00110110" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "0110"; end if; estado <= "011";
-- ASCII "6"
            when "00110111" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "0111"; end if; estado <= "011";
-- ASCII "7"
            when "00111000" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "1000"; end if; estado <= "011";
-- ASCII "8"
            when "00111001" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "1001"; end if; estado <= "011";
-- ASCII "9"
            when "01000001" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "1010"; end if; estado <= "011";
-- ASCII "A"
            when "01000010" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "1011"; end if; estado <= "011";
-- ASCII "B"
            when "01000011" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "1100"; end if; estado <= "011";
-- ASCII "C"
```

```
            when "01000100" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "1101"; end if; estado <= "011";
-- ASCII "D"
            when "01000101" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "1110"; end if; estado <= "011";
-- ASCII "E"
            when "01000110" => if (n<2) and (i<N_Registers_Out) then
             Registers_Out(i)((n*4) TO ((n*4)+3)) <= "1111"; end if; estado <= "011";
-- ASCII "F"
            when others => estado <= "000"; -- ASCII null
          end case;

          when "011" =>
          if (n<1) then
          n<=n+1;
          elsif (i>AD_MAROC_DATA-1) AND (i<N_Registers_Out) then
            i<=i+1;
            n<=0;
          end if;
          estado <= "000";

          when "100" =>
          if cont < 10 then
          cont <= cont+1;
          else
            cont <=0;
            estado <= "101";
          end if;

          when "101" =>
          if cont < 10 then
            out_signal_test_buff <= '1';
          cont <= cont+1;
          else
            cont <=0;
            out_signal_test_buff <= '0';
            estado <= "000";
          end if;

          when others => estado <= "000";
        end case;
```

```
    end if;
  end process;

end Behavioral;
```

Código VHDL referente ao bloco de envio de mensagem via UART_TX.

```
----------------------------------------------------------------------------------
-- Bloco de Coincidencia e tratamento dos triggers

----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;

entity SendDataMAROC3 is
Port ( dataMaroc3 : in  STD_LOGIC_VECTOR (0 to 828);
    SendDataMaroc : in  STD_LOGIC;
    clk :  IN  STD_LOGIC;
    out_signal_test : out  STD_LOGIC;
    D_SC : out  STD_LOGIC;
    clk_SC :  OUT  STD_LOGIC);


end SendDataMAROC3;

architecture Behavioral of SendDataMAROC3 is
  constant N_DATASENDMAROC :integer := 829;
  constant teste1 :std_LOGIC_VECTOR(0 to 827):=x"8003E8000000000000000
  00000000000000000030B06887201008040201008040201008040201008040201008
  040201008040201008040201008040201008040201008040201008040201008040201
  10080402010080402010080402010080402010080402010080402010080402010080
  10080402010080402010080402010080F000000000000001";
  constant teste2 :std_LOGIC:='1';
  signal dataMaroc3_Buff      :std_logic_vector(0 to N_DATASENDMAROC-1);
  signal i :integer range 0 to N_DATASENDMAROC:= 0;
  signal clk_SC_Buff: std_LOGIC :='1';
  signal flag_send1: std_LOGIC :='0';
  signal flag_send2: std_LOGIC :='0';
  signal estado: std_LOGIC_vector(0 to 1):="00";
begin
  clk_SC<=clk_SC_Buff;
```

```
dataMaroc3_Buff<=dataMaroc3;

process(sendDataMaroc)
  begin
  if falling_edge(sendDataMaroc) then
    flag_send1<=NOT flag_send1;
  end if;
end process;

process (clk)
  begin
  if (falling_edge(clk))then
    if ((flag_send1 XOR flag_send2) = '1') then
    clk_SC_Buff<= not clk_SC_Buff;
    else
      clk_SC_Buff<='1';
    end if;

    case estado is

      when "00" =>
      if ((flag_send1 XOR flag_send2) = '1') then
        D_SC<=dataMaroc3_Buff(i);
        estado<="01";
      end if;

      when "01" =>
      if ((flag_send1 XOR flag_send2) = '1') then
        if i<N_DATASENDMAROC-1 then
          i<=i+1;
        estado<="00";
        else
          i<=0;
          flag_send2<=NOT flag_send2;
          estado<="10";
        end if;

      end if;
      when "10" =>
      D_SC<='0';
      estado<="00";
```

```
        when others => estado<="00";
      end case;

    end if;

  end process;


end Behavioral;
```

Apêndice I

Código de Aquisição de Dados do Detector CREAT3

```
////////////////////////////////////////////////////////
// Código de Aquisição de Dados do Detector CREAT3
// Author: Leonardo Chaves Ruiz Guedes (leocrg@cbpf.br)
////////////////////////////////////////////////////////

#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <unistd.h>
#include <time.h>
#include <math.h>
#include <fcntl.h>      //Incluido para utilizar ctes O_RDWR | O_NOCTTY |
   O_NDELAY
#include <termios.h>    //Esta é a biblioteca de comunicação serial
#include <string.h>
#include <libgen.h>
#include <errno.h>       // Error number definitions
#include <stdbool.h>

char Reg_DAC0[] = "01C2";
char Reg_DAC1[] = "0000";
char USB_Port[30] = "0";
char *Reg_Ctest = "8080202008080202";
char Reg_N_gate[30] = "0A";
```

```
char Reg_word_0[30] = "A8";
char Reg_word_1[30] = "3B";
char Reg_word_2[30] = "C6";
char Reg_word_3[30] = "23";
char Reg_word_4[30] = "D9";
char Reg_word_5[30] = "02";
char binReg_word_6[30] = "11100000";
char Reg_word_6[30] = "";
char binReg_OTHERS[30] = "00000000";
char Reg_OTHERS[30] = "";
char DataSend[30]="000000000000000000";
int IndinChar1=0;
double z0[4] = {0,0,0,0};
// number of noise hits
int Noise_ena = 0;
int Crosstalk_ena = 0;// Map position 8x8
int Coin_Map[8][8];    //axi y
double width_Tile=0;  //5cm
double length_Tile=0; //40cm (need to be a square)
double height_Tile=0;
//  model: ModelProd
int ModelProd = 0;       // 0 = cos^2(theta), 1 = cos(2*theta)
double xprod[2]  = {0,0};      // (reference origin x = 0)
double yprod[2]  = {0,0};      // (reference origin y = 0)///UART
int open_port(char* USB_Port){
    int fd_UART;
    (void)sprintf(USB_Port,"/dev/ttyACM%c",*USB_Port);
    fd_UART = open(USB_Port, O_RDWR | O_NOCTTY);
    if (fd_UART == -1){
        fprintf(stderr, "open_port: Unable to open UART Communication (%s) %
   s\n",USB_Port,strerror(errno));
        exit(EXIT_FAILURE);
    }
    return fd_UART;
}
char* Hex_to_bin(char* HexValue){
    long int i=8;
    char BinValue[100]="";
    printf("b");
    while(i<strlen(HexValue)){
        switch(HexValue[i]){
```

```
            case '0': printf("0000"); break;
            case '1': printf("0001"); break;
            case '2': printf("0010"); break;
            case '3': printf("0011"); break;
            case '4': printf("0100"); break;
            case '5': printf("0101"); break;
            case '6': printf("0110"); break;
            case '7': printf("0111"); break;
            case '8': printf("1000"); break;
            case '9': printf("1001"); break;
            case 'A': printf("1010"); break;
            case 'B': printf("1011"); break;
            case 'C': printf("1100"); break;
            case 'D': printf("1101"); break;
            case 'E': printf("1110"); break;
            case 'F': printf("1111"); break;
            case 'a': printf("1010"); break;
            case 'b': printf("1011"); break;
            case 'c': printf("1100"); break;
            case 'd': printf("1101"); break;
            case 'e': printf("1110"); break;
            case 'f': printf("1111"); break;
            default:  ;
        }
        i++;
    }
    printf("\r\n");
    char* cp = BinValue;
    return cp;
}
char* Save_Hex_to_bin(char* HexValue, FILE* fpOut, long int i, long int
    maxInd){
    char BinValue[100]="";
    while(i<maxInd){
        switch(HexValue[i]){
            case '0': fprintf(fpOut,"0000"); break;
            case '1': fprintf(fpOut,"0001"); break;
            case '2': fprintf(fpOut,"0010"); break;
            case '3': fprintf(fpOut,"0011"); break;
            case '4': fprintf(fpOut,"0100"); break;
            case '5': fprintf(fpOut,"0101"); break;
```

```
            case '6': fprintf(fpOut,"0110"); break;
            case '7': fprintf(fpOut,"0111"); break;
            case '8': fprintf(fpOut,"1000"); break;
            case '9': fprintf(fpOut,"1001"); break;
            case 'A': fprintf(fpOut,"1010"); break;
            case 'B': fprintf(fpOut,"1011"); break;
            case 'C': fprintf(fpOut,"1100"); break;
            case 'D': fprintf(fpOut,"1101"); break;
            case 'E': fprintf(fpOut,"1110"); break;
            case 'F': fprintf(fpOut,"1111"); break;
            case 'a': fprintf(fpOut,"1010"); break;
            case 'b': fprintf(fpOut,"1011"); break;
            case 'c': fprintf(fpOut,"1100"); break;
            case 'd': fprintf(fpOut,"1101"); break;
            case 'e': fprintf(fpOut,"1110"); break;
            case 'f': fprintf(fpOut,"1111"); break;
            default:  ;
        }
        i++;
    }
    char* cp = BinValue;
    return cp;
}
char* Save_Hex_to_Dec(char* HexValue, FILE* fpOut, long int minIn, long int
    maxInd){
    char BinValue[100]="";
    int i;
    int x;
    for (i = minIn; i <= maxInd; i++){
        BinValue[i]= HexValue[i];
    }
    sscanf(BinValue, "%x", &x);
    fprintf(fpOut,"%d", x);
    char* cp = BinValue;
    return cp;
}
void ReadSetupInfo(char* namedata ){
    FILE* in = fopen(namedata, "r");
    if ( !in ){
        printf("\n");
        printf("***[WARMING] file %s not exist***\n",namedata);
```

```
}
else{
    int line=0;
    char CharIn[400];
    char c;
    while (line<19) {
        fscanf(in, "%c", &c);
        if((c!='\n')&&(c>0)){
            (void)sprintf(CharIn,"%s%c",CharIn,c);
        }
        if (c=='\n') {
            if(line==1){
                sscanf(CharIn,"%d %d %d %d %d %d %d %d",
                &Coin_Map[0][0],&Coin_Map[0][1],&Coin_Map[0][2],&
Coin_Map[0][3],
                &Coin_Map[0][4],&Coin_Map[0][5],&Coin_Map[0][6],&
Coin_Map[0][7]);
            }
            if(line==2){
                sscanf(CharIn,"%d %d %d %d %d %d %d %d",
                &Coin_Map[1][0],&Coin_Map[1][1],&Coin_Map[1][2],&
Coin_Map[1][3],
                &Coin_Map[1][4],&Coin_Map[1][5],&Coin_Map[1][6],&
Coin_Map[1][7]);
            }
            if(line==3){
                sscanf(CharIn,"%d %d %d %d %d %d %d %d",
                &Coin_Map[2][0],&Coin_Map[2][1],&Coin_Map[2][2],&
Coin_Map[2][3],
                &Coin_Map[2][4],&Coin_Map[2][5],&Coin_Map[2][6],&
Coin_Map[2][7]);
            }
            if(line==4){
                sscanf(CharIn,"%d %d %d %d %d %d %d %d",
                &Coin_Map[3][0],&Coin_Map[3][1],&Coin_Map[3][2],&
Coin_Map[3][3],
                &Coin_Map[3][4],&Coin_Map[3][5],&Coin_Map[3][6],&
Coin_Map[3][7]);
            }
            if(line==5){
                sscanf(CharIn,"%d %d %d %d %d %d %d %d",
```

```
                    &Coin_Map[4][0],&Coin_Map[4][1],&Coin_Map[4][2],&
Coin_Map[4][3],
                    &Coin_Map[4][4],&Coin_Map[4][5],&Coin_Map[4][6],&
Coin_Map[4][7]);
                }
                if(line==6){
                    sscanf(CharIn,"%d %d %d %d %d %d %d %d",
                    &Coin_Map[5][0],&Coin_Map[5][1],&Coin_Map[5][2],&
Coin_Map[5][3],
                    &Coin_Map[5][4],&Coin_Map[5][5],&Coin_Map[5][6],&
Coin_Map[5][7]);
                }
                if(line==7){
                    sscanf(CharIn,"%d %d %d %d %d %d %d %d",
                    &Coin_Map[6][0],&Coin_Map[6][1],&Coin_Map[6][2],&
Coin_Map[6][3],
                    &Coin_Map[6][4],&Coin_Map[6][5],&Coin_Map[6][6],&
Coin_Map[6][7]);
                }
                if(line==8){
                    sscanf(CharIn,"%d %d %d %d %d %d %d %d",
                    &Coin_Map[7][0],&Coin_Map[7][1],&Coin_Map[7][2],&
Coin_Map[7][3],
                    &Coin_Map[7][4],&Coin_Map[7][5],&Coin_Map[7][6],&
Coin_Map[7][7]);
                }
                if(line==9){
                    sscanf(CharIn,"%*s %lf",&width_Tile);
                }
                if(line==11){
                    sscanf(CharIn,"%*s %lf",&height_Tile);
                }
                if(line==12){
                    sscanf(CharIn,"%*s %lf %lf %lf %lf",&z0[0],&z0[1],&z0
[2],&z0[3]);
                }
                (void)sprintf(CharIn,"");
                line++;
            }
        }
        length_Tile = 8*width_Tile;
```

```
        fclose(in);
    }
}
void SaveSetupInfo(char* namedata){
    FILE* fpOut = fopen(namedata, "w+");
    if ( !fpOut )
    perror(namedata) ;
    int a, b;
    fprintf(fpOut,"Map_position_8x8:\n");
    for(a=0;a<8;a++){
        for(b=0;b<8;b++){
            fprintf(fpOut,"%d\t",Coin_Map[a][b]);
        }
        fprintf(fpOut,"\n");
    }
    fprintf(fpOut,"Tile_width=\t%lf\n",width_Tile);
    fprintf(fpOut,"Tile_length=\t%lf\n",length_Tile);
    fprintf(fpOut,"Tile_height=\t%lf\n",height_Tile);
    fprintf(fpOut,"vertical_distance_from_origin=\t%lf\t%lf\t%lf\t%lf\n",z0
    [0],z0[1],z0[2],z0[3]);
    fprintf(fpOut,"\n");
    fprintf(fpOut,"----Simulation----\n");
    fprintf(fpOut,"Modelprod=\t%d\t//0 = cos^2(theta), 1 = cos(2*theta), 2 =
     Uniform\n",ModelProd);
    fprintf(fpOut,"xprod=\t%lf\t%lf\t//(reference origin x = 0)\n",xprod[0],
    xprod[1]);
    fprintf(fpOut,"yprod=\t%lf\t%lf\t//(reference origin y = 0)\n",yprod[0],
    yprod[1]);
    fprintf(fpOut,"Noise_Enable=\t%d\t%d\n",Noise_ena,Crosstalk_ena);
    fprintf(fpOut,"\n");
    fprintf(fpOut,"----Measurement_Registers----\n");
    fprintf(fpOut,"Reg_word_0\t%s\n",Reg_word_0);
    fprintf(fpOut,"Reg_word_1\t%s\n",Reg_word_1);
    fprintf(fpOut,"Reg_word_2\t%s\n",Reg_word_2);
    fprintf(fpOut,"Reg_word_3\t%s\n",Reg_word_3);
    fprintf(fpOut,"Reg_word_4\t%s\n",Reg_word_4);
    fprintf(fpOut,"Reg_word_5\t%s\n",Reg_word_5);
    fprintf(fpOut,"Reg_word_6\t%s\n",Reg_word_6);
    fprintf(fpOut,"Reg_N_gate\t%s\n",Reg_N_gate);
    fprintf(fpOut,"Reg_Ctest\t%s\n",Reg_Ctest);
    fprintf(fpOut,"Reg_DAC0\t%s\n",Reg_DAC0);
```

```
    fprintf(fpOut,"Reg_DAC1\t%s\n",Reg_DAC1);
    fprintf(fpOut,"Reg_OTHERS\t%s\n",Reg_OTHERS);
    if ( fpOut ) (void)fclose(fpOut) ;
}
int PrintValues(char *DataReceive, char *timeString){
    int i;
    printf ("%s\t",timeString);
    printf ("0x");
    for (i = 0; i < strlen(DataReceive); i++){
        printf ("%c",DataReceive[i]);
        if (i==3) printf ("\t0x");
        if (i==7) printf ("\t0x");
    }
    printf ("\t");
    (void)Hex_to_bin(DataReceive);
}
int SaveFile(FILE* fpOut, char *DataReceive, char *timeString){
    int i;
    (void)fprintf(fpOut,"%s\t",timeString);
    for (i = 0; i < 8; i++){
        (void)fprintf(fpOut,"%c",DataReceive[i]);
        if ((i==7)||(i==3)) (void)fprintf(fpOut,"\t");
    }
    (void)Save_Hex_to_bin(DataReceive,fpOut,8,strlen(DataReceive));
    fprintf(fpOut,"\t0\t0\r\n");
}
char RX_UART(int fd_UART, FILE* fpOut, char*PrintEna){
    char  DataReceive[300]="";
    char    aux[1];
    int i;
    bool stringComplete=false;
    bool enviaserial=false;
    char timeString[20] ;
    while (!stringComplete){
        if (read(fd_UART,aux,1)>0){
            if (aux[0] == ']') {
                if ((enviaserial)&&(IndinChar1==180)) {
                    fprintf(fpOut,"%s\r\n",DataReceive);
                }
                enviaserial = false;
            }
```

```
            if (aux[0] == '}') {
                if ((enviaserial)&&(IndinChar1==24)) { //96
                    SaveFile(fpOut,DataReceive,timeString);
                    if (PrintEna[0]=='1') PrintValues(DataReceive,timeString
    );
                    stringComplete = true;
                }
                enviaserial = false;
            }
            if (enviaserial) {
                (void)sprintf(DataReceive, "%s%c",DataReceive,aux[0]);
                IndinChar1++;
            }

            if ((aux[0] == '{')||(aux[0] == '[')) {
                time_t myTime = time(0) ;
                struct tm* now_time ;
                now_time = localtime(&myTime) ;
                strftime(timeString, sizeof(timeString), "%d/%m/%y_%H:%M:%S"
    , now_time) ;
                (void)sprintf(DataReceive, "");
                enviaserial = true;
                IndinChar1=0;
            }
        }
    }
}

int TX_UART(int fd_UART){
    int i;
    printf("\r\f");
    sleep(2);
    char dataRegister[55]="";
    (void)sprintf(dataRegister,"a%sb%sc%sd%se%sf%sg%sh%sj%sk%sl%s{%s}",
    Reg_word_0,Reg_word_1,Reg_word_2,Reg_word_3,Reg_word_4,Reg_word_5,
    Reg_word_6,Reg_N_gate,Reg_DAC0,Reg_DAC1,Reg_OTHERS,Reg_Ctest);
    printf("%s",dataRegister);
    write(fd_UART,dataRegister,strlen(dataRegister));
    sleep(2);
    if (tcflush(fd_UART, TCIOFLUSH) != 0) perror("tcflush error");
    sleep(1);
```

```
    printf("\r\n");
}

void ReadRegisters(int fd_UART, char*PrintEna){
    char  DataReceive[300]="";
    char    aux[1];
    int i;
    bool stringComplete=false;
    bool enviaserial=false;
    char timeString[20] ;

    while (!stringComplete){
        if (read(fd_UART,aux,1)>0){
            if (aux[0] == ']') {
                stringComplete = true;
                enviaserial = false;
            }
            if (enviaserial) {
                (void)sprintf(DataReceive, "%s%c",DataReceive,aux[0]);
                IndinChar1++;
            }
            if ((aux[0] == '[')) {
                (void)sprintf(DataReceive, "");
                enviaserial = true;
                IndinChar1=0;
            }
        }
    }
    if (IndinChar1==180){
        if (DataSend[10]=='0') for ( i=0; i<2; i++)  Reg_word_0[i] =
    DataReceive[11+i];
        if (DataSend[11]=='0') for ( i=0; i<2; i++)  Reg_word_1[i] =
    DataReceive[25+i];
        if (DataSend[12]=='0') for ( i=0; i<2; i++)  Reg_word_2[i] =
    DataReceive[39+i];
        if (DataSend[13]=='0') for ( i=0; i<2; i++)  Reg_word_3[i] =
    DataReceive[53+i];
        if (DataSend[14]=='0') for ( i=0; i<2; i++)  Reg_word_4[i] =
    DataReceive[67+i];
        if (DataSend[15]=='0') for ( i=0; i<2; i++)  Reg_word_5[i] =
    DataReceive[81+i];
```

```
        if (DataSend[16]=='0') for ( i=0; i<2; i++)  Reg_word_6[i] =
    DataReceive[95+i];
        if (DataSend[5]=='0') for ( i=0; i<2; i++)  Reg_N_gate[i] =
    DataReceive[109+i];
        if (DataSend[6]=='0') for ( i=0; i<4; i++)  Reg_DAC0[i] =
    DataReceive[148+i];
        if (DataSend[7]=='0') for ( i=0; i<4; i++)  Reg_DAC1[i] =
    DataReceive[162+i];
        if (DataSend[17]=='0') for ( i=0; i<2; i++)  Reg_OTHERS[i] =
    DataReceive[178+i];
        if (DataSend[8]=='0') for ( i=0; i<16; i++) Reg_Ctest[i] =
    DataReceive[122+i];
    }
}

int main(int argc, char** argv){
    struct termios   options;    // Terminal options
    int rc;          // Return value
    cfsetispeed(&options, B115200);         // Set the baud rates to 115200
    cfsetospeed(&options, B115200);         // Set the baud rates to 115200
    cfmakeraw(&options);
    options.c_cflag |= (CLOCAL | CREAD);    // Enable the receiver and set
    local mode
    options.c_cflag &= ~CSTOPB;             // 1 stop bit
    options.c_cflag &= ~CRTSCTS;            // Disable hardware flow control
    options.c_cc[VMIN]  = 1;
    options.c_cc[VTIME] = 2;

    int c, d;
    int k=0;
    int option = 0;
    char fileName[1024] = "Acquisition/DefautFile.dat" ;
    char fileaux[1024];
    int nEvents = 1;
    char *DataReceive;
    char* pEnd;
    char PrintEna[1] = "1" ;
    char DefaultReg[1] = "0" ;
    char fileNamecfg[1024] = "" ;
    (void)sprintf(Reg_word_6,"%X",(int)strtol (binReg_word_6,&pEnd,2));
    (void)sprintf(Reg_OTHERS,"0%X",(int)strtol (binReg_OTHERS,&pEnd,2));
```

```
 while ((option = getopt(argc, argv, "HPhF:N:T:g:t:k:c:a:b:i:d:e:f:w:y:l:
m:")) != -1) {
     switch (option) {
         case 'H':
         printf("\r\n") ;
         printf("-H\t--help\r\n") ;
         printf("-F\t\"File to save (str)\"\r\n") ;
         printf("-T\t\"TX_UART USB (dec)\t(COM0)\"\r\n") ;
         printf("-N\t\"number of events (dec)\"\r\n") ;
         printf("-P\t\"--Disable print values\"\r\n") ;
         printf("-h\t\"--Use registeres default values\"\r\n") ;
         printf("-g\t\"Time window (dec)\"\r\n") ;
         printf("-t\t\"Reg_DAC0 Value (dec)\"\r\n") ;
         printf("-k\t\"Reg_DAC1 Value (dec)\"\r\n") ;
         printf("-c\t\"Reg_Ctest Channels (hex)\"\r\n") ;
         printf("-a\t\"Reg_word_0 (hex)\"\r\n") ;
         printf("-b\t\"Reg_word_1 (hex)\"\r\n") ;
         printf("-i\t\"Reg_word_2 (hex)\"\r\n") ;
         printf("-d\t\"Reg_word_3 (hex)\"\r\n") ;
         printf("-e\t\"Reg_word_4 (hex)\"\r\n") ;
         printf("-f\t\"Reg_word_5 (hex)\"\r\n") ;
         printf("-w\t\"along trigger Enable (bin)\"\r\n") ;
         printf("-y\t\"LabView Enable (bin)\"\r\n") ;
         printf("-l\t\"small_dac Enable (bin)\"\r\n") ;
         printf("-m\t\"time Resolution (dec) (2^n/1000 sec)\"\r\n") ;
         printf("\r\n") ;
         exit(1) ;
         DataSend[0]='1';
         break;
         case 'P' :
         PrintEna[0]='0';
         break;
         case 'h' :
         DefaultReg[0]='1';
         (void)sprintf(DataSend,"1111111111111111");
         break;
         case 'F' :
         (void)strcpy(fileaux, optarg) ;
         (void)sprintf(fileName,"%s",fileaux);
         DataSend[1]='1';
         break;
```

```
            case 'N' :
            nEvents = atoi(optarg) ;
            DataSend[2]='1';
            break;
            case 'T' :
            (void)strcpy(USB_Port , optarg) ;
            DataSend[3]='1';
            break;
            case 'g' :
            if(atoi(optarg)<256){
                if (atoi(optarg)<16) (void)sprintf(Reg_N_gate,"0%X",atoi(
optarg));
                else (void)sprintf(Reg_N_gate,"%X",atoi(optarg));
                printf("Reg_N_gate changed to:\t0x%s\t%s(dec)\r\n",
Reg_N_gate,optarg);
                }else{
                printf("Error - Reg_N_gate was not changed\r\n");
            }

            DataSend[5]='1';
            break;
            case 't' :
            if(atoi(optarg)<1024){
                if (atoi(optarg)<16) (void)sprintf(Reg_DAC0,"000%X",atoi(
optarg));
                else if (atoi(optarg)<256) (void)sprintf(Reg_DAC0,"00%X",
atoi(optarg));
                else (void)sprintf(Reg_DAC0,"0%X",atoi(optarg));
                printf("Reg_DAC0 changed to:\t0x%s\t%s(dec)\r\n",Reg_DAC0,
optarg);
                }else{
                printf("Error - Reg_DAC0 was not changed\r\n");
            }
            DataSend[6]='1';
            break;
            case 'k' :
            if(atoi(optarg)<1024){
                if (atoi(optarg)<16) (void)sprintf(Reg_DAC1,"000%X",atoi(
optarg));
                else if (atoi(optarg)<256) (void)sprintf(Reg_DAC1,"00%X",
atoi(optarg));
```

```
            else (void)sprintf(Reg_DAC1,"0%X",atoi(optarg));
            printf("Reg_DAC1 changed to:\t0x%s\t%s(dec)\r\n",Reg_DAC1,
optarg);
            }else{
            printf("Error - Reg_DAC1 was not changed\r\n");
        }
        DataSend[7]='1';
        break;
        case 'c' :
        if(strlen(optarg)==16){
            Reg_Ctest=optarg;
            printf("Reg_Ctest changed to:\t0x%s\r\n",Reg_Ctest);
            }else{
            printf("Error - Reg_Ctest was not changed\r\n");
        }
        DataSend[8]='1';
        break;
        case 'a' :
        if (strlen(optarg)==1) {
            (void)sprintf(Reg_word_0,"0%s",optarg);
            printf("Reg_word_0 changed to:\t0x%s\r\n",Reg_word_0);
        }
        else if (strlen(optarg)==2){
            (void)sprintf(Reg_word_0,"%s",optarg);
            printf("Reg_word_0 changed to:\t0x%s\r\n",Reg_word_0);
            }else{
            printf("Error - Reg_word_0 was not changed\r\n");
        }

        DataSend[9]='1';
        break;
        case 'b' :
        if (strlen(optarg)==1) {
            (void)sprintf(Reg_word_1,"0%s",optarg);
            printf("Reg_word_1 changed to:\t0x%s\r\n",Reg_word_1);
        }
        else if (strlen(optarg)==2){
            (void)sprintf(Reg_word_1,"%s",optarg);
            printf("Reg_word_1 changed to:\t0x%s\r\n",Reg_word_1);
            }else{
            printf("Error - Reg_word_1 was not changed\r\n");
```

```
}

DataSend[10]='1';
break;
case 'i' :
if (strlen(optarg)==1) {
    (void)sprintf(Reg_word_2,"0%s",optarg);
    printf("Reg_word_2 changed to:\t0x%s\r\n",Reg_word_2);
}
else if (strlen(optarg)==2){
    (void)sprintf(Reg_word_2,"%s",optarg);
    printf("Reg_word_2 changed to:\t0x%s\r\n",Reg_word_2);
    }else{
    printf("Error - Reg_word_2 was not changed\r\n");
}

DataSend[11]='1';
break;
case 'd' :
if (strlen(optarg)==1) {
    (void)sprintf(Reg_word_3,"0%s",optarg);
    printf("Reg_word_3 changed to:\t0x%s\r\n",Reg_word_3);
}
else if (strlen(optarg)==2){
    (void)sprintf(Reg_word_3,"%s",optarg);
    printf("Reg_word_3 changed to:\t0x%s\r\n",Reg_word_3);
    }else{
    printf("Error - Reg_word_3 was not changed\r\n");
}

DataSend[12]='1';
break;
case 'e' :
if (strlen(optarg)==1) {
    (void)sprintf(Reg_word_4,"0%s",optarg);
    printf("Reg_word_4 changed to:\t0x%s\r\n",Reg_word_4);
}
else if (strlen(optarg)==2){
    (void)sprintf(Reg_word_4,"%s",optarg);
    printf("Reg_word_4 changed to:\t0x%s\r\n",Reg_word_4);
    }else{
```

```
            printf("Error - Reg_word_4 was not changed\r\n");
        }

        DataSend[13]='1';
        break;
        case 'f' :
        if (strlen(optarg)==1) {
            (void)sprintf(Reg_word_5,"0%s",optarg);
            printf("Reg_word_5 changed to:\t0x%s\r\n",Reg_word_5);
        }
        else if (strlen(optarg)==2){
            (void)sprintf(Reg_word_5,"%s",optarg);
            printf("Reg_word_5 changed to:\t0x%s\r\n",Reg_word_5);
            }else{
            printf("Error - Reg_word_5 was not changed\r\n");
        }

        DataSend[14]='1';
        break;
        case 'w' :
        if (optarg[0]=='0') {
            binReg_word_6[1]='0';
            (void)sprintf(Reg_word_6,"%X",(int)strtol (binReg_word_6,&
pEnd,2));
            printf("Along Trigger\t\t(Disable)\r\n",Reg_word_6);
        }
        else if (optarg[0]=='1') {
            binReg_word_6[1]='1';
            (void)sprintf(Reg_word_6,"%X",(int)strtol (binReg_word_6,&
pEnd,2));
            printf("Along Trigger\t\t(Enable)\r\n",Reg_word_6);
            }else{
            printf("Error - Reg_word_6 was not changed\r\n");
        }
        DataSend[15]='1';
        break;
        case 'y' :
        if (optarg[0]=='0') {
            binReg_word_6[2]='1';
            (void)sprintf(Reg_word_6,"%X",(int)strtol (binReg_word_6,&
pEnd,2));
```

```
            printf("LabView\t\t\t(Disable)\r\n",Reg_word_6);
        }
        else if (optarg[0]=='1') {
            binReg_word_6[2]='0';
            (void)sprintf(Reg_word_6,"%X",(int)strtol (binReg_word_6,&
pEnd,2));
            printf("LabView\t\t\t(Enable)\r\n",Reg_word_6);
            }else{
            printf("Error - Reg_word_6 was not changed\r\n");
        }
        DataSend[16]='1';
        break;

        case 'l' :
        if (optarg[0]=='0') {
            binReg_OTHERS[7]='0';
            (void)sprintf(Reg_OTHERS,"0%X",(int)strtol (binReg_OTHERS,&
pEnd,2));
            printf("Small_dac\t\t(Disable)\r\n",Reg_OTHERS);
        }
        else if (optarg[0]=='1') {
            binReg_OTHERS[7]='1';
            (void)sprintf(Reg_OTHERS,"0%X",(int)strtol (binReg_OTHERS,&
pEnd,2));
            printf("Small_dac\t\t(Enable)\r\n",Reg_OTHERS);
            }else{
            printf("Error - Reg_OTHERS was not changed\r\n");
        }
        DataSend[17]='1';
        break;
        case 'm' :
        if(atoi(optarg)<32){
            for (c = 4; c >= 0; c--)
            {
                d = atoi(optarg) >> c;
                if (d & 1) binReg_OTHERS[4-c]='1';
                else binReg_OTHERS[4-c]='0';
            }
            if ((int)strtol(binReg_OTHERS,&pEnd,2)<16) (void)sprintf(
Reg_OTHERS,"0%X",(int)strtol(binReg_OTHERS,&pEnd,2));
            else (void)sprintf(Reg_OTHERS,"%X",(int)strtol(binReg_OTHERS
```

```
,&pEnd,2));
                printf("Reg_OTHERS changed to:\t0x%s\tTime Resulution = %lf(
sec)\r\n",Reg_OTHERS,(double)pow(2,(double)atoi(optarg))/1000.);
                }else{
                printf("Error - Reg_OTHERS was not changed\r\n");
            }
            DataSend[17]='1';
            break;
            default:
            exit(1) ;
        }
    }
    int fd_UART = open_port(USB_Port);
    if((rc = tcgetattr(fd_UART, &options)) < 0){
        fprintf(stderr, "failed to get attr: %d, %s\n", fd_UART, strerror(
errno));
        exit(EXIT_FAILURE);
    }
    if((rc = tcgetattr(fd_UART, &options)) < 0){
        fprintf(stderr, "failed to get attr: %d, %s\n", fd_UART, strerror(
errno));
        exit(EXIT_FAILURE);
    }
    strncpy(fileNamecfg,fileName,strlen(fileName)-3);
    sprintf(fileNamecfg,"%scfg",fileNamecfg);
    ReadSetupInfo(fileNamecfg);
    sleep(1);
    if (DefaultReg[0]=='0') ReadRegisters(fd_UART,PrintEna);
    SaveSetupInfo(fileNamecfg);
    sleep(1);
    TX_UART(fd_UART);
    FILE* fpOut = fopen(fileName, "w+");
    if ( !fpOut )
    perror(fileName) ;
    printf ("Start Acquisition...\r\n");
    printf ("\r\n");
    if (PrintEna[0]=='1') printf ("Date\t\t\t1msEv\tUN_AD\tData Receive(HEX)
\tData Receive(BIN)\r\n");
    (void)fprintf(fpOut, "Date\t\t\t1msEv\tUN_AD\tCH_trigger\t\t\t\t\t\t\t\
    ttheta_gen\tphi_gen\r\n") ;
    (void)fprintf(fpOut,"Reg_word_0   %s    Reg_word_1    %s    Reg_word_2
```

```
    %s    Reg_word_3    %s    Reg_word_4    %s    Reg_word_5    %s
   Reg_word_6    %s    Reg_N_gate    %s    Reg_Ctest    %s    Reg_DAC0    %s
       Reg_DAC1    %s    Reg_OTHERS    %s    \r\n",
    Reg_word_0,Reg_word_1,Reg_word_2,Reg_word_3,Reg_word_4,Reg_word_5,
   Reg_word_6,Reg_N_gate,Reg_Ctest,Reg_DAC0,Reg_DAC1,Reg_OTHERS);
    while (k < nEvents)
    {
        k++;
        RX_UART(fd_UART, fpOut,PrintEna);
    }
    close(fd_UART);
    if ( fpOut ) (void)fclose(fpOut) ;
    printf ("Acquisition Finished!\r\n");
    printf ("\r\n");
    return 0;
}
```

Apêndice J

Código de Simulação de Traços do Detector CREAT3

```
///// Fast simulation:
///// muons cosmics passing through set of detectors positioned one on the
    top of each other
///// Scintillating tiles detector with 64 channels organized in 4 planes of
     8 axis x and 8 axis y
///// author: André Massafferri
///// Modificated by: Leonardo Guedes
///// date: july 2016
/////////////// configured parameters ////////////////////////////
///// all dimensions in cm !!!!!!!!!!!!
//--------------------------------------------
#include <stdio.h>
#include "TFile.h"
#include "TTree.h"
#include "TRandom.h"
#include "TCanvas.h"
#include <time.h>
#include "TMath.h"

// vertical distance from origin
Double_t z[4] = {0,20,70,90};
// number of noise hits
Int_t Noise_ena = 0;
```

```
Int_t Crosstalk_ena = 1;

// Map position 8x8
Int_t Coin_Map[8][8]=  {{ 0  , 1  , 2  , 3  , 4  , 5  , 6  , 7  }, //axi x
            { 8  , 9  , 10 , 11 , 12 , 13 , 14 , 15 }, //axi y

            { 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 }, //axi x
            { 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 }, //axi y

            { 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 }, //axi x
            { 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 }, //axi y

            { 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 }, //axi x
            { 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 }}; //axi y

Double_t Noise_Map[8][8]=  {{0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 ,
     0.001 , 0.001},
              {0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 ,
   0.001},
              {0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 ,
   0.001},
              {0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 ,
   0.001},
              {0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 ,
   0.001},
              {0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 ,
   0.001},
              {0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 ,
   0.001},
              {0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 , 0.001 ,
   0.001}};

Double_t Crosstalk_Map[3][3]= { {0.01,0.05, 0.01},
                {0.05,1   , 0.05},
                {0.01,0.05, 0.01}};

Int_t InitNum = 11110000;
Int_t InitNumNoise = 11110000;
Int_t InitNumCrosstalk = 11110000;
/// geometries of detectors with rectangles:
Double_t width_Tile=5;  //5cm
```

```
Double_t length_Tile=8*width_Tile; //40cm (need to be a square)
Double_t height_Tile=1;
//-----------------------------------------------
/// Muon production parameters
/// geometry: defining rectangle in z=0 and area with vertices (0.0,0.0) (
    xprod,yprod)
//  model: ModelProd
Int_t ModelProd = 0;       // 0 = cos^2(theta), 1 = cos(2*theta), 2 = uniform
Double_t xprod[2]  = {0,40};      // (reference origin x = 0)
Double_t yprod[2]  = {0,40};      // (reference origin y = 0)
TRandom3 *r1=new TRandom3();
r1->SetSeed(InitNum);
TRandom3 *r2=new TRandom3();
r2->SetSeed(InitNumNoise);
TRandom3 *r3=new TRandom3();
r3->SetSeed(InitNumCrosstalk);
//-----------------------------------------------
/// global variables
Double_t pi     = atan(1)*4;
Int_t linewidth = 3; // line width of histogram
Char_t name1[20];
Int_t chx[4],chy[4]; //output channels
Int_t OutAcceptance[4];
Int_t InAcceptance[4];
Double_t Acceptance[4];
Int_t ntracks;
Double_t x_gen,y_gen,theta_gen,phi_gen;
Int_t vector_ev[64] =
    {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

Int_t run=0;
TFile *hist_gen;
//-----------------------------------------------
// histograms
TH2F *hch[4];
TH2F *hxy[4];
TH1F *hxprod;
TH1F *hyprod;
TH1F *hphiprod_5;
TH1F *hthetaprod_5;
TH1F *hphiprod_10;
```

```
TH1F *hthetaprod_10;
TH1F *hphiprod_15;
TH1F *hthetaprod_15;
TH1F *hphiprod_20;
TH1F *hthetaprod_20;
TH1F *hphiprod_40;
TH1F *hthetaprod_40;
TH1F *hphiprod_60;
TH1F *hthetaprod_60;
TH1F *hphiprod_80;
TH1F *hthetaprod_80;
TH1F *hphiprod_100;
TH1F *hthetaprod_100;
//-----------------------------------------------
// comparison empirical results
Double_t Acc_Lab[4] = {1,0.4,0.1};
char rootfile[1024]="";
char auxrootfile[1024]="";
Loop_Run(){
  xprod[0] = -20;
  xprod[1] = 60;
  yprod[0] = -20;
  yprod[1] = 60;
  ModelProd =0;
  Noise_ena=0;
  Crosstalk_ena =0;
  char filename[1024];
  char fileconfig[1024];
  char auxfileconfig[1024];
  Double_t auxz[100][4]={ {0,40,110,150},
    {0,50,100,150},
    {0,40,80,150},
    {0,20,130,150},
    {0,20,130,150},
    {0,20,130,150},
    {0,20,130,150},
    {0,20,130,150},
  {0,20,130,150}};

  for (Int_t i=0;i<3;i++){
    for (Int_t j=0;j<4;j++){
```

```
      z[j]=auxz[i][j];
    }
    sprintf(filename,"Acquisition/Simul_teste_1_11059200_loop_%d.dat",i);
    strncpy(auxfileconfig,filename,strlen(filename)-3);
    sprintf(fileconfig,"%scfg",auxfileconfig);
    SaveSetupInfo(fileconfig);
    Run(11059200,filename);
  }
}

void ReadSetupInfo(char* namedata ){
  ifstream in;
  in.open(namedata);
  Int_t line=0;
  Int_t x=0;

  if(!in.is_open( ))  // * verifica se o programa conseguiu abrir o arquivo
   de nome informado pelo usuario *
  {
    cout<<"\n***[WARMING] file \"" << namedata << "\" not exist!" << endl;
    cout<<"\n***[WARMING] Creating file \"" << namedata << "\" with default
   parameters..." << endl;
    in.clear( ); //reseta o objeto leitura
    x=1;
  }

  if(x==0){
    char CharIn[400];
    char c;
    while (line<19) {
      in.get(c);
      if((c!='\n')&&(c>0)) (void)sprintf(CharIn,"%s%c",CharIn,c);
      if (c=='\n') {
        if(line==1){
          sscanf(CharIn,"%d %d %d %d %d %d %d %d",
          &Coin_Map[0][0],&Coin_Map[0][1],&Coin_Map[0][2],&Coin_Map[0][3],
          &Coin_Map[0][4],&Coin_Map[0][5],&Coin_Map[0][6],&Coin_Map[0][7]);
        }
        if(line==2){
          sscanf(CharIn,"%d %d %d %d %d %d %d %d",
          &Coin_Map[1][0],&Coin_Map[1][1],&Coin_Map[1][2],&Coin_Map[1][3],
```

```
  &Coin_Map[1][4],&Coin_Map[1][5],&Coin_Map[1][6],&Coin_Map[1][7]);
}
if(line==3){
  sscanf(CharIn,"%d %d %d %d %d %d %d %d",
  &Coin_Map[2][0],&Coin_Map[2][1],&Coin_Map[2][2],&Coin_Map[2][3],
  &Coin_Map[2][4],&Coin_Map[2][5],&Coin_Map[2][6],&Coin_Map[2][7]);
}
if(line==4){
  sscanf(CharIn,"%d %d %d %d %d %d %d %d",
  &Coin_Map[3][0],&Coin_Map[3][1],&Coin_Map[3][2],&Coin_Map[3][3],
  &Coin_Map[3][4],&Coin_Map[3][5],&Coin_Map[3][6],&Coin_Map[3][7]);
}
if(line==5){
  sscanf(CharIn,"%d %d %d %d %d %d %d %d",
  &Coin_Map[4][0],&Coin_Map[4][1],&Coin_Map[4][2],&Coin_Map[4][3],
  &Coin_Map[4][4],&Coin_Map[4][5],&Coin_Map[4][6],&Coin_Map[4][7]);
}
if(line==6){
  sscanf(CharIn,"%d %d %d %d %d %d %d %d",
  &Coin_Map[5][0],&Coin_Map[5][1],&Coin_Map[5][2],&Coin_Map[5][3],
  &Coin_Map[5][4],&Coin_Map[5][5],&Coin_Map[5][6],&Coin_Map[5][7]);
}
if(line==7){
  sscanf(CharIn,"%d %d %d %d %d %d %d %d",
  &Coin_Map[6][0],&Coin_Map[6][1],&Coin_Map[6][2],&Coin_Map[6][3],
  &Coin_Map[6][4],&Coin_Map[6][5],&Coin_Map[6][6],&Coin_Map[6][7]);
}
if(line==8){
  sscanf(CharIn,"%d %d %d %d %d %d %d %d",
  &Coin_Map[7][0],&Coin_Map[7][1],&Coin_Map[7][2],&Coin_Map[7][3],
  &Coin_Map[7][4],&Coin_Map[7][5],&Coin_Map[7][6],&Coin_Map[7][7]);
}
if(line==9){
  sscanf(CharIn,"%*s %lf",&width_Tile);
}
if(line==11){
  sscanf(CharIn,"%*s %lf",&height_Tile);
}
if(line==12){
  sscanf(CharIn,"%*s %lf %lf %lf %lf",&z[0],&z[1],&z[2],&z[3]);
}
```

```
        if(line==15){
          sscanf(CharIn,"%*s %d",&ModelProd);
        }
        if(line==16){
          sscanf(CharIn,"%*s %lf %lf",&xprod[0], &xprod[1]);
        }
        if(line==17){
          sscanf(CharIn,"%*s %lf %lf",&yprod[0], &yprod[1]);
        }
        if(line==18){
          sscanf(CharIn,"%*s %d %d",&Noise_ena, &Crosstalk_ena);
        }
        (void)sprintf(CharIn,"");
        line++;
      }
    }
    length_Tile = 8*width_Tile;
    in.close();
  }
}
//-----------------------------------------------
// main function
Int_t Run(Int_t nEvents = 100, char* namefile = "Acquisition/DefautFile.dat"
    ) //Event -> Poissonian number of muons in prod area per sec
{
  strncpy(auxrootfile,namefile,strlen(namefile)-4);
  sprintf(rootfile,"%s_gen.root",auxrootfile);
  hist_gen = new TFile(rootfile,"RECREATE");
  char namefilecfg[1024]="";
  char auxnamefilecfg[1024]="";
  strncpy(auxnamefilecfg,namefile,strlen(namefile)-3);
  sprintf(namefilecfg,"%scfg",auxnamefilecfg);
  ReadSetupInfo(namefilecfg);
  Initialization();
  hxprod->Reset();
  hyprod->Reset();
  hphiprod_5->Reset();
  hthetaprod_5->Reset();
  hphiprod_10->Reset();
  hthetaprod_10->Reset();
  hphiprod_15->Reset();
```

```
hthetaprod_15->Reset();
hphiprod_20->Reset();
hthetaprod_20->Reset();
hphiprod_40->Reset();
hthetaprod_40->Reset();
hphiprod_60->Reset();
hthetaprod_60->Reset();
hphiprod_80->Reset();
hthetaprod_80->Reset();
hphiprod_100->Reset();
hthetaprod_100->Reset();
SaveSetupInfo(namefilecfg);
ofstream fpIn;
fpIn.open(namefile);

char timeString[20];
time_t myTime = time(0) ;
struct tm* now_time ;
now_time = localtime(&myTime) ;
strftime(timeString, sizeof(timeString), "%d/%m/%y_%H:%M:%S", now_time) ;
fpIn << "Data      1msEv UN_AD CH_trigger                    theta_gen phi_gen"
  << endl;
fpIn << "Reg_word_0 00   Reg_word_1   00   Reg_word_2   00   Reg_word_3   00
 Reg_word_4   00   Reg_word_5   00   Reg_word_6   00   Reg_N_gate   00   Reg_Ctest
  0000000000000000   Reg_DAC0   0000   Reg_DAC1   0000   Reg_OTHERS   00" <<
 endl;

for (Int_t i = 0; i < nEvents; i++)
{
  Int_t nmuons = 1;
  for  (Int_t j = 0; j < nmuons; j++)
  {
    /// production of one muon given by the Model -> function of theta,
 the angle wrt vertical axis
    /// in a rectangle in z=0 varying uniformly 0->xprod and 0->yprod
    GenerateMuon(ModelProd);// output : x_gen, y_gen, theta_gen, phi_gen
    hxprod->Fill(x_gen);
    hyprod->Fill(y_gen);
    hthetaprod_5->Fill(theta_gen*180/pi);
    hphiprod_5->Fill(phi_gen*180/pi);
    hthetaprod_10->Fill(theta_gen*180/pi);
```

```
    hphiprod_10->Fill(phi_gen*180/pi);
    hthetaprod_15->Fill(theta_gen*180/pi);
    hphiprod_15->Fill(phi_gen*180/pi);
    hthetaprod_20->Fill(theta_gen*180/pi);
    hphiprod_20->Fill(phi_gen*180/pi);
    hthetaprod_40->Fill(theta_gen*180/pi);
    hphiprod_40->Fill(phi_gen*180/pi);
    hthetaprod_60->Fill(theta_gen*180/pi);
    hphiprod_60->Fill(phi_gen*180/pi);
    hthetaprod_80->Fill(theta_gen*180/pi);
    hphiprod_80->Fill(phi_gen*180/pi);
    hthetaprod_100->Fill(theta_gen*180/pi);
    hphiprod_100->Fill(phi_gen*180/pi);
    ntracks++;
    for (Int_t idet = 0; idet<4; idet++)
    {
      /// propagation of the muon track in each detector
      /// identify the cells chx and chy in which the muon hit (100%
efficiently, without noise)
      /// output: chx, chy (if no hit is identified -> chx or chy = 1000)
      MuonInDetector(idet,chx[idet],chy[idet]);
      if ((chx[idet]==1000) || (chy[idet]==1000))
      {
        OutAcceptance[idet]++;
        } else {
        makevector_ev(idet);
        if ( Crosstalk_ena == 1 ) CrossTalkInDetector(idet);
        hch[idet]->Fill(chx[idet],chy[idet]);
        InAcceptance[idet]++;
      }
    }
    if ( Noise_ena == 1 ) NoiseInDetector();
    if (FlagCondition()==1){
      fpIn << timeString << "\t0\t0\t";
      for (Int_t a =0; a <64; a++){
        fpIn << vector_ev[a];
      }
      fpIn << "\t" << theta_gen << "\t" << phi_gen << "\r\n";
    }
    for (Int_t ii = 0; ii<64; ii++) vector_ev[ii]= 0;
  }
```

```
  }
  fpIn.close();
  Results();
  hist_gen->Write();
  return 0;
}
Int_t FlagCondition(){
  Int_t Flagcond = 0;
  for (Int_t a =0; a <64; a++){
    if (vector_ev[a]==1) Flagcond = 1;
  }
  return Flagcond;
}
void SaveSetupInfo(char* namedata ){
  ofstream fpOut;
  fpOut.open(namedata);
  fpOut << "Map_position_8x8:" << endl;
  for(Int_t a=0;a<8;a++){
    for(Int_t b=0;b<8;b++){
      fpOut << Coin_Map[a][b] << "\t";
    }
    fpOut << "\r\n";
  }
  fpOut << "Tile_width=\t" << width_Tile << endl;
  fpOut << "Tile_length=\t" << length_Tile << endl;
  fpOut << "Tile_height=\t" << height_Tile << endl;
  fpOut << "vertical_distance_from_origin=\t" << z[0] << "\t" << z[1] << "\t
   " << z[2] << "\t" << z[3] << endl;
  fpOut << " " <<endl;
  fpOut << "----Simulation----" <<endl;
  fpOut << "Modelprod=  " << ModelProd << " //0 = cos^2(theta), 1 = cos(2*
   theta), 2 = Uniform" <<endl;
  fpOut << "xprod=  " << xprod[0] << "\t" << xprod[1] << "  //(reference
   origin x = 0)" <<endl;
  fpOut << "yprod=  " << yprod[0] << "\t" << yprod[1] << "  //(reference
   origin y = 0)" <<endl;
  fpOut << "Noise/Crosstalk_Enable= " << Noise_ena << "\t" << Crosstalk_ena
   <<endl;
  fpOut << " " <<endl;
  fpOut << "----Measurement_Registers----" <<endl;
  fpOut << "Reg_word_0  00" <<endl;
```

```
  fpOut << "Reg_word_1   00" <<endl;
  fpOut << "Reg_word_2   00" <<endl;
  fpOut << "Reg_word_3   00" <<endl;
  fpOut << "Reg_word_4   00" <<endl;
  fpOut << "Reg_word_5   00" <<endl;
  fpOut << "Reg_word_6   00" <<endl;
  fpOut << "Reg_N_gate   00" <<endl;
  fpOut << "Reg_Ctest 0000000000000000" <<endl;
  fpOut << "Reg_DAC0   0000" <<endl;
  fpOut << "Reg_DAC1   0000" <<endl;
  fpOut << "Reg_OTHERS   00" <<endl;
  fpOut.close();
}
//put the hits in a vector
void makevector_ev(Int_t idet){
  vector_ev[Coin_Map[((idet)*2)][chx[idet]]]=1;
  vector_ev[Coin_Map[((idet)*2+1)][chy[idet]]]=1;
}

void Initialization(){
  hxprod = new TH1F("hxprod","h x prod",100,xprod[0],xprod[1]);
  hyprod = new TH1F("hyprod","h y prod",100,yprod[0],yprod[1]);
  hphiprod_5 = new TH1F("hphiprod_5","h phi sim prod",5,-180,180);
  hthetaprod_5 = new TH1F("hthetaprod_5","h theta sim prod",5,0,90);
  hphiprod_10 = new TH1F("hphiprod_10","h phi sim prod",10,-180,180);
  hthetaprod_10 = new TH1F("hthetaprod_10","h theta sim prod",10,0,90);
  hphiprod_15 = new TH1F("hphiprod_15","h phi sim prod",15,-180,180);
  hthetaprod_15 = new TH1F("hthetaprod_15","h theta sim prod",15,0,90);
  hphiprod_20 = new TH1F("hphiprod_20","h phi sim prod",20,-180,180);
  hthetaprod_20 = new TH1F("hthetaprod_20","h theta sim prod",20,0,90);
  hphiprod_40 = new TH1F("hphiprod_40","h phi sim prod",40,-180,180);
  hthetaprod_40 = new TH1F("hthetaprod_40","h theta sim prod",40,0,90);
  hphiprod_60 = new TH1F("hphiprod_60","h phi sim prod",60,-180,180);
  hthetaprod_60 = new TH1F("hthetaprod_60","h theta sim prod",60,0,90);
  hphiprod_80 = new TH1F("hphiprod_80","h phi sim prod",80,-180,180);
  hthetaprod_80 = new TH1F("hthetaprod_80","h theta sim prod",80,0,90);
  hphiprod_100 = new TH1F("hphiprod_100","h phi sim prod",100,-180,180);
  hthetaprod_100 = new TH1F("hthetaprod_100","h theta sim prod",100,0,90);
  if (ModelProd==0) cout << " Production model: (costheta)^2 " << endl;
  if (ModelProd==1) cout << " Production model: cos(2*theta)  " << endl;
  if (ModelProd==2) cout << " Production model: Uniform  " << endl;
```

```
  for (Int_t idet=0; idet<4; idet++)
  {
    cout << " DETECTOR " << idet << endl;
    cout << " type: creat " << endl;
    cout << "      location x (nchannels = 8) = 0 " << 8*width_Tile << endl;
    cout << "      location y (nchannels = 8) = 0 " << 8*width_Tile << endl;
    cout << "      location z = " << z[idet] << endl;
    if (((Double_t)(8.*width_Tile) > xprod[1]) || ( xprod[0]<0 )) cout << "
   WARNING: [x] detector larger than production area " << endl;
    if (((Double_t)(8.*width_Tile) > yprod[1]) || ( yprod[0]<0 )) cout << "
   WARNING: [y] detector larger than production area " << endl;
    cout << " --------- " << endl;
    OutAcceptance[idet] = 0;
    InAcceptance[idet]  = 0;
    ntracks             = 0;
    sprintf(name1,"hch_%d",idet);
    hch[idet] = new TH2F(name1,name1,8,0,8.,8,0,8.);
  }
}
void Results(){

  TCanvas *c1 = new TCanvas("c1","Muon Production details",200,10,700,500);
  c1->SetFillColor(0);
  c1->GetFrame()->SetFillColor(6);
  c1->GetFrame()->SetBorderSize(10);
  c1->GetFrame()->SetBorderMode(3);
  c1->Divide(2,2);
  hxprod->SetLineWidth(linewidth);
  hyprod->SetLineWidth(linewidth);
  hthetaprod_100->SetLineWidth(linewidth);
  hphiprod_100->SetLineWidth(linewidth);
  TF1 *gcosthetasim;
  switch(ModelProd){
    case 0: gcosthetasim = new TF1("gcosthetasim","[0]*pow(cos(x*pi/180)
    ,[1])*sin(x*pi/180)*cos(x*pi/180)",0,90); break;
    case 1: gcosthetasim = new TF1("gcosthetasim","[0]*cos([1]*x*pi/180)"
    ,0,90); break;
    case 2: gcosthetasim = new TF1("gcosthetasim","pol0",0,90); break;
  }
  if (ModelProd!=2) gcosthetasim->SetParameters(1,2);
```

```
c1->cd(1);
hxprod->SetMinimum(0);
hxprod->Draw();
c1->cd(2);
hyprod->SetMinimum(0);
hyprod->Draw();
c1->cd(3);
hthetaprod_100->SetMinimum(0);
hthetaprod_100->Fit("gcosthetasim","R");
c1->cd(4);
hphiprod_100->SetMinimum(0);
hphiprod_100->Draw();
TCanvas *c2 = new TCanvas("c2","Detector details CH",200,10,700,500);
c2->SetFillColor(0);
c2->GetFrame()->SetFillColor(6);
c2->GetFrame()->SetBorderSize(10);
c2->GetFrame()->SetBorderMode(3);
Double_t Temp = TMath::Sqrt(4);
Int_t num1 = Temp;
Int_t num2 = Temp+1;
c2->Divide(num1,num2);
for (Int_t idet=0; idet<4; idet++)
{
  c2->cd(idet+1);
  hch[idet]->SetLineWidth(linewidth);
  hch[idet]->SetMinimum(0);
  hch[idet]->Draw("LEGO");
  Acceptance[idet] = Double_t(InAcceptance[idet])/Double_t(ntracks);
  cout << "Eff geom (det=" << idet << ") over production plane = " <<
 Acceptance[idet] << endl;
}
cout << " ntracks produced = " << ntracks << endl;
TCanvas *c3 = new TCanvas("c3","flux in detector over production plane"
 ,200,10,700,500);
c3->SetFillColor(42);
TGraph *g = new TGraph(4,z,Acceptance);
g->SetLineColor(2);
g->SetLineWidth(4);
g->SetMarkerColor(4);
g->SetMarkerStyle(21);
g->SetMinimum(0);
```

```
  g->SetTitle("Acceptance x Distance");
  g->GetXaxis()->SetTitle("distance(cm)");
  g->GetYaxis()->SetTitle("acceptance");
  g->Draw("AL*");
  TGraph *h = new TGraph(4,z,Acc_Lab);
  h->SetLineColor(5);
  h->SetLineWidth(4);
  h->SetMarkerColor(5);
  h->SetMarkerStyle(23);
  h->Draw("Same");

}
void MuonInDetector(Int_t detector, Int_t& channel_x, Int_t& channel_y){
  /// linear propagation from area of production to the detector

  Double_t radii = tan(theta_gen)*z[detector];
  Double_t x     = radii*cos(phi_gen) + x_gen;
  Double_t y     = radii*sin(phi_gen) + y_gen;
  channel_x = SearchChannelX(x);
  channel_y = SearchChannelY(y);
}
void CrossTalkInDetector(Int_t idet){
  Int_t lin;
  Int_t col;
  Int_t prob;
  lin=(Coin_Map[((idet)*2)][chx[idet]])/8;
  col=((Coin_Map[((idet)*2)][chx[idet]])%8);
  for (Int_t a=lin-1; a<(lin+2); a++){
    for (Int_t b=col-1; b<(col+2); b++){
      if (Crosstalk_Map[a-(lin-1)][b-(col-1)]==0) prob = 0;
      else prob = r3 -> Uniform(1,1/Crosstalk_Map[a-(lin-1)][b-(col-1)]);
      if ((prob == 1)&&(a>=0)&&(b>=0)&&(a<8)&&(b<8)) vector_ev[8*a+b]=1;
    }
  }
  lin=(Coin_Map[((idet)*2+1)][chy[idet]])/8;
  col=((Coin_Map[((idet)*2+1)][chy[idet]])%8);
  for (Int_t a=lin-1; a<(lin+2); a++){
    for (Int_t b=col-1; b<(col+2); b++){
      if (Crosstalk_Map[a-(lin-1)][b-(col-1)]==0) prob = 0;
      else prob = r3 -> Uniform(1,1/Crosstalk_Map[a-(lin-1)][b-(col-1)]);
      if ((prob == 1)&&(a>=0)&&(b>=0)&&(a<8)&&(b<8)) vector_ev[8*a+b]=1;
```

```
    }
  }
}

void NoiseInDetector(){
  Int_t prob;
  for (Int_t a=0; a<8; a++){
    for (Int_t b=0; b<8; b++){
      if (Noise_Map[a][b]!=0){
        if (Noise_Map[a][b]==0) prob = 0; //7-a => a
        else prob = r2 -> Uniform(1,1/Noise_Map[a][b]);
        if (prob == 1) vector_ev[Coin_Map[a][b]]=1;
      }
    }
  }
}

Int_t SearchChannelX(Double_t x){
  /// identifying the cell in x
  Int_t icell = 0;
  if (x>=0){
    while (1)
    {
      if ( ( (x > (icell*width_Tile) ) && (x <= ((icell+1)*width_Tile) ) )
   || (icell==8) )  break;
      icell++;
    }
    if (icell < 8)
    {
      return = icell;
    } else
    {
      return = 1000;
    }
  }return = 1000;
}
Int_t SearchChannelY(Double_t y){
  /// identifying the cell in y
  Int_t icell = 0;
  if (y>=0){
    while (1)
```

```
    {
      if ( ( (y > (icell*width_Tile) ) && (y <= ((icell+1)*width_Tile) ) )
    || (icell==8) )  break;
      icell++;
    }
    if (icell < 8)
    {
      return = icell;
    } else
    {
      return = 1000;
    }
  }return = 1000;
}
void GenerateMuon(Int_t model){
  /// generating muons uniformly in the rectangle
  /// with phi angle also uniform
  x_gen   = r1->Uniform(xprod[0],xprod[1]);
  y_gen   = r1->Uniform(yprod[0],yprod[1]);

  phi_gen = r1 -> Uniform(-1*pi,pi);         //// -pi to pi
  Double_t r    = 1.;
  Double_t prod = 0.;
  /// generating the angle wrt the normal according the model
  /// 0->cos^2(theta), 1->cos(2*theta)
  if (model==2) theta_gen = r1 -> Uniform(0,pi/2);
  else{
    while (r>prod)
    {
      r = r1 -> Uniform(0,1.1);
      theta_gen = r1 -> Uniform(0,pi/2);  //// 0 to pi/2
      if (model==0) prod = pow(cos(theta_gen),2.)*sin(theta_gen)*cos(
   theta_gen);
      if (model==1) prod = cos(2.0*theta_gen);
    }
  }
}
```

Apêndice K

Código de Trajetografia do Detector CREAT3

```
///// Tracking:
///// muons cosmics passing through set of detectors positioned one on the
    top of each other
///// Scintillating tiles detector with 64 channels organized in 4 planes of
     8 axis x and 8 axis y
///// author: Leonardo Guedes
///// date: july 2016
/////////////// configured parameters ////////////////////////////
///// all dimensions in cm !!!!!!!!!!!!
//-------------------------------------------------------------
#include <stdio.h>
#include "TGraph.h"
#include "TAxis.h"
#include "TCanvas.h"
#include "TLine.h"
#include <time.h>
#include <TGraph2D.h>
#include <TRandom.h>
#include <TStyle.h>
#include <TF2.h>
#include <TH1.h>
#include "TMath.h"
#include "TFile.h"
#include "TTree.h"
```

```
Double_t pi = atan(1)*4;
typedef struct {
  Double_t x_det0, y_det0, z_det0, phi, theta, chisquaxz, chisquayz, err_phi
    , err_theta;
  Int_t ncells, ncellsdthetaP, ncellsdthetaM, ncellsdphiP, ncellsdphiM;
} tracktype1;
typedef struct {
  Double_t x_det0, y_det0, z_det0, phi, theta, chisquaxz, chisquayz, err_phi
    , err_theta;
} tracktype2;
Double_t phi_gen;
Double_t theta_gen;
Double_t aux_x1;
Double_t aux_y1;
Double_t aux_z1;
Double_t aux_phi1;
Double_t aux_theta1;
Double_t aux_chisquaxz1=100000;
Double_t aux_chisquayz1=100000;
Double_t aux_err_phi1;
Double_t aux_err_theta1;
Double_t aux_x2;
Double_t aux_y2;
Double_t aux_z2;
Double_t aux_phi2;
Double_t aux_theta2;
Double_t aux_chisquaxz2=100000;
Double_t aux_chisquayz2=100000;
Double_t aux_err_phi2;
Double_t aux_err_theta2;
Double_t aux_x3;
Double_t aux_y3;
Double_t aux_z3;
Double_t aux_phi3;
Double_t aux_theta3;
Double_t aux_chisquaxz3=100000;
Double_t aux_chisquayz3=100000;
Double_t aux_err_phi3;
Double_t aux_err_theta3;
Int_t aux_npoints;
Int_t aux_n_combi_tr;
```

```
Double_t aux_phi_sim;
Double_t aux_theta_sim;
Int_t aux_ndet;
Int_t aux_ncells;
Int_t aux_ncellsdthetaP;
Int_t aux_ncellsdthetaM;
Int_t aux_ncellsdphiP;
Int_t aux_ncellsdphiM;
Int_t over_max_npoints_cont = 0;
Int_t treeOK=0;
Double_t xzp0=0,xzp1=0,yzp0=0,yzp1=0,chisquaxz=0,chisquayz=0;
Double_t Err_xzp0=0,Err_xzp1=0,Err_yzp0=0,Err_yzp1=0;
//Mapa de posição de cada canal
Int_t Coin_Map_pos_xz[4][8]    = {{ 0 , 1  , 2  , 3  , 4  , 5  , 6  , 7  },
  { 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 },
  { 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 },
  { 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 }};
Int_t Coin_Map_pos_yz[4][8]    = {{ 8 , 9  , 10 , 11 , 12 , 13 , 14 , 15 },
  { 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 },
  { 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 },
  { 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 }};
//largura de cada canal
Double_t tile_width_ch;
Double_t tile_height_ch;
Double_t Map_height_z[4];
Int_t Coin_Map_hit_xz[4][8];
Int_t Coin_Map_hit_yz[4][8];
void Loop_maketrack(){
  char filename[1024];
  for (Int_t i=0;i<3;i++){
    sprintf(filename,"Acquisition/Simul_teste_1_11059200_loop_%d.dat",i);
    maketrack(-1,filename,2,50);
  }
}
void maketrack(Int_t nevents=-1, char* input_Datafile = "Acquisition/
   DefautFile.dat", Int_t min_num_detectors=2, Int_t max_npoints = 50){
  char nametreeout[1024]="";
  char auxnametreeout[1024]="";
  char * nchar;
  Int_t intchar;
  intchar = strlen(input_Datafile);
```

```
  nchar = strchr(input_Datafile,'.');
  strncpy(auxnametreeout,input_Datafile,(int(nchar-input_Datafile+1)));
  sprintf(nametreeout,"%sroot",auxnametreeout);
  char SetupInfoFile[1024]="";
  char auxSetupInfoFile[1024]="";
  cout << (int(nchar-input_Datafile+1)) << endl;
  strncpy(auxSetupInfoFile,input_Datafile,(int(nchar-input_Datafile+1)));
  sprintf(SetupInfoFile,"%scfg",auxSetupInfoFile);
  aux_x1=0;
  aux_y1=0;
  aux_z1=0;
  aux_phi1=0;
  aux_theta1=0;
  aux_chisquaxz1=100000;
  aux_chisquayz1=100000;
  aux_x2=0;
  aux_y2=0;
  aux_z2=0;
  aux_phi2=0;
  aux_theta2=0;
  aux_chisquaxz2=100000;
  aux_chisquayz2=100000;
  aux_x3=0;
  aux_y3=0;
  aux_z3=0;
  aux_phi3=0;
  aux_theta3=0;
  aux_chisquaxz3=100000;
  aux_chisquayz3=100000;
  aux_npoints=0;
  aux_n_combi_tr=0;
  aux_phi_sim=0;
  aux_theta_sim=0;
  aux_ndet=0;
  treeOK=0;
  over_max_npoints_cont = 0;
  if (ReadSetupInfo(SetupInfoFile)==0){
    Readinput_file(input_Datafile, max_npoints, min_num_detectors, nevents,
    nametreeout);
  }
}
```

```
Int_t ReadSetupInfo(char* namedata ){
  ifstream in;
  in.open(namedata);
  Int_t line=0;
  Int_t x=0;
  if(!in.is_open( ))  // * verifica se o programa conseguiu abrir o arquivo
   de nome informado pelo usuario *
  {
    cout<<"\n***[ERROR] file \"" << namedata << "\" not exist!" << endl;
    in.clear( ); //reseta o objeto leitura
    x=1;
  }
  if(x==0){
    char CharIn[400];
    Char_t varteste[3];
    char c;
    while (line<13) {
      in.get(c);
      if(c!='\n') (void)sprintf(CharIn,"%s%c",CharIn,c);
      if (c=='\n') {
        if(line==1){
          sscanf(CharIn,"%d %d %d %d %d %d %d %d",
          &Coin_Map_pos_xz[0][0],&Coin_Map_pos_xz[0][1],&Coin_Map_pos_xz
  [0][2],&Coin_Map_pos_xz[0][3],
          &Coin_Map_pos_xz[0][4],&Coin_Map_pos_xz[0][5],&Coin_Map_pos_xz
  [0][6],&Coin_Map_pos_xz[0][7]);
        }
        if(line==2){
          sscanf(CharIn,"%d %d %d %d %d %d %d %d",
          &Coin_Map_pos_yz[0][0],&Coin_Map_pos_yz[0][1],&Coin_Map_pos_yz
  [0][2],&Coin_Map_pos_yz[0][3],
          &Coin_Map_pos_yz[0][4],&Coin_Map_pos_yz[0][5],&Coin_Map_pos_yz
  [0][6],&Coin_Map_pos_yz[0][7]);
        }
        if(line==3){
          sscanf(CharIn,"%d %d %d %d %d %d %d %d",
          &Coin_Map_pos_xz[1][0],&Coin_Map_pos_xz[1][1],&Coin_Map_pos_xz
  [1][2],&Coin_Map_pos_xz[1][3],
          &Coin_Map_pos_xz[1][4],&Coin_Map_pos_xz[1][5],&Coin_Map_pos_xz
  [1][6],&Coin_Map_pos_xz[1][7]);
        }
```

```
    if(line==4){
      sscanf(CharIn,"%d %d %d %d %d %d %d %d",
      &Coin_Map_pos_yz[1][0],&Coin_Map_pos_yz[1][1],&Coin_Map_pos_yz
[1][2],&Coin_Map_pos_yz[1][3],
      &Coin_Map_pos_yz[1][4],&Coin_Map_pos_yz[1][5],&Coin_Map_pos_yz
[1][6],&Coin_Map_pos_yz[1][7]);
    }
    if(line==5){
      sscanf(CharIn,"%d %d %d %d %d %d %d %d",
      &Coin_Map_pos_xz[2][0],&Coin_Map_pos_xz[2][1],&Coin_Map_pos_xz
[2][2],&Coin_Map_pos_xz[2][3],
      &Coin_Map_pos_xz[2][4],&Coin_Map_pos_xz[2][5],&Coin_Map_pos_xz
[2][6],&Coin_Map_pos_xz[2][7]);
    }
    if(line==6){
      sscanf(CharIn,"%d %d %d %d %d %d %d %d",
      &Coin_Map_pos_yz[2][0],&Coin_Map_pos_yz[2][1],&Coin_Map_pos_yz
[2][2],&Coin_Map_pos_yz[2][3],
      &Coin_Map_pos_yz[2][4],&Coin_Map_pos_yz[2][5],&Coin_Map_pos_yz
[2][6],&Coin_Map_pos_yz[2][7]);
    }
    if(line==7){
      sscanf(CharIn,"%d %d %d %d %d %d %d %d",
      &Coin_Map_pos_xz[3][0],&Coin_Map_pos_xz[3][1],&Coin_Map_pos_xz
[3][2],&Coin_Map_pos_xz[3][3],
      &Coin_Map_pos_xz[3][4],&Coin_Map_pos_xz[3][5],&Coin_Map_pos_xz
[3][6],&Coin_Map_pos_xz[3][7]);
    }
    if(line==8){
      sscanf(CharIn,"%d %d %d %d %d %d %d %d",
      &Coin_Map_pos_yz[3][0],&Coin_Map_pos_yz[3][1],&Coin_Map_pos_yz
[3][2],&Coin_Map_pos_yz[3][3],
      &Coin_Map_pos_yz[3][4],&Coin_Map_pos_yz[3][5],&Coin_Map_pos_yz
[3][6],&Coin_Map_pos_yz[3][7]);
    }
    if(line==9){
      sscanf(CharIn,"%*s %lf",&tile_width_ch);
    }
    if(line==11){
      sscanf(CharIn,"%*s %lf",&tile_height_ch);
    }
```

```
        if(line==12){
          sscanf(CharIn,"%*s %lf %lf %lf %lf",&Map_height_z[0],&Map_height_z
    [1],&Map_height_z[2],&Map_height_z[3]);
        }
        (void)sprintf(CharIn,"");
        line++;
      }
    }
    cout << "Tile_width= \t" << tile_width_ch << endl;
    cout << "Height(z)= \t[" << Map_height_z[0] << " " << Map_height_z[1] <<
    " " << Map_height_z[2] << " " << Map_height_z[3]<< "]" << endl;
    cout << "MAP:" << endl;
    for(Int_t a=0;a<4;a++){
      for(Int_t b=0;b<8;b++){
        cout << Coin_Map_pos_xz[a][b] << "\t";
      }
      cout << "\n";
      for(Int_t b=0;b<8;b++){
        cout << Coin_Map_pos_yz[a][b] << "\t";
      }
      cout << "\n";
    }
    in.close();
  }
  return x;
}
void Readinput_file(char* namedata, Int_t max_npoints, Int_t
    min_num_detectors, Int_t nevents, char* nametreeout){
  Int_t Entries_Tree_cont =0;
  static tracktype1 Track1;
  static tracktype2 Track2, Track3;
  Int_t n_tracks, n_points, n_det;
  Int_t hit_vector[64];
  Double_t phi_sim, theta_sim;
  Int_t un_id;
  Char_t date[20];
  Int_t time_register;
  TFile *track = new TFile(nametreeout,"RECREATE");
  TTree *Tracking = new TTree("Tracking","Tracking");
  Tracking->Branch("date",&date,"date/C");
  Tracking->Branch("time_register",&time_register,"time_register/I");
```

```
Tracking->Branch("un_id",&un_id,"un_id/I");
Tracking->Branch("n_tracks",&n_tracks,"n_tracks/I");
Tracking->Branch("n_points",&n_points,"n_points/I");
Tracking->Branch("phi_sim",&phi_sim,"phi_sim/D");
Tracking->Branch("theta_sim",&theta_sim,"theta_sim/D");
Tracking->Branch("n_det",&n_det,"n_det/I");
Tracking->Branch("hit_vector",&hit_vector,"hit_vector[64]/I");
Tracking->Branch("Track1",&Track1,"x_det0/D:y_det0/D:z_det0/D:phi/D:theta/
 D:chisquaxz/D:chisquayz/D:err_phi/D:err_theta/D:ncells/I:ncellsdthetaP/I:
 ncellsdthetaM/I:ncellsdphiP/I:ncellsdphiM/I");
Tracking->Branch("Track2",&Track2,"x_det0/D:y_det0/D:z_det0/D:phi/D:theta/
 D:chisquaxz/D:chisquayz/D:err_phi/D:err_theta/D");
Tracking->Branch("Track3",&Track3,"x_det0/D:y_det0/D:z_det0/D:phi/D:theta/
 D:chisquaxz/D:chisquayz/D:err_phi/D:err_theta/D");
ifstream in;
in.open(namedata);
Int_t EvtId;
Int_t SecEv;
Char_t CH_triggers[64];
Char_t dateAux[20];
char CharIn[360];
char c;
Int_t nrun=0;
cout << "Readinput_file..."<< endl;
while ((nrun<nevents)||(nevents==-1)) {
  in.get(c);
  (void)sprintf(CharIn,"%s%c",CharIn,c);
  if (c=='\n') {
    if((CharIn[0]!='D')&&(CharIn[0]>0)&&(CharIn[0]!='R')){
      nrun++;
      if (nrun%100==0) printf("%d\n",nrun);
      if (!in.good()) break;
      aux_x1=0;
      aux_y1=0;
      aux_z1=0;
      aux_phi1=0;
      aux_theta1=0;
      aux_chisquaxz1=100000;
      aux_chisquayz1=100000;
      aux_x2=0;
      aux_y2=0;
```

```
      aux_z2=0;
      aux_phi2=0;
      aux_theta2=0;
      aux_chisquaxz2=100000;
      aux_chisquayz2=100000;
      aux_x3=0;
      aux_y3=0;
      aux_z3=0;
      aux_phi3=0;
      aux_theta3=0;
      aux_chisquaxz3=100000;
      aux_chisquayz3=100000;
      aux_npoints=0;
      aux_n_combi_tr=0;
      aux_phi_sim=0;
      aux_theta_sim=0;
      aux_ndet=0;
      treeOK=0;
      aux_ncells=0;
      aux_ncellsdthetaP=0;
      aux_ncellsdthetaM=0;
      aux_ncellsdphiP=0;
      aux_ncellsdphiM=0;
      sscanf(CharIn,"%s %x %x %s %lf %lf",dateAux,&SecEv,&EvtId,
CH_triggers,&aux_theta_sim,&aux_phi_sim);
      for(Int_t a=0;a<4;a++){
        for(Int_t b=0;b<8;b++){
          Coin_Map_hit_xz[a][b]=(Int_t)CH_triggers[Coin_Map_pos_xz[a][b
]]-48;
          Coin_Map_hit_yz[a][b]=(Int_t)CH_triggers[Coin_Map_pos_yz[a][b
]]-48;
        }
      }
      MakePoints(max_npoints, min_num_detectors);
      if (treeOK==1){
        sprintf(date,"%s",dateAux);
        time_register=SecEv;
        un_id=EvtId;
        n_tracks = aux_n_combi_tr;
        n_points = aux_npoints;
        phi_sim = aux_phi_sim;
```

```
    theta_sim = aux_theta_sim;
    n_det = aux_ndet;
    find_all_ncells();

    for(a=0;a<64;a++) hit_vector[a] = (Int_t)CH_triggers[a]-48;
    Track1.x_det0=aux_x1;
    Track1.y_det0=aux_y1;
    Track1.z_det0=aux_z1;
    Track1.phi=aux_phi1;
    Track1.theta=aux_theta1;
    Track1.chisquaxz=aux_chisquaxz1;
    Track1.chisquayz=aux_chisquayz1;
    Track1.err_phi=aux_err_phi1;
    Track1.err_theta=aux_err_theta1;
    Track1.ncells=aux_ncells;
    Track1.ncellsdthetaP=aux_ncellsdthetaP;
    Track1.ncellsdthetaM=aux_ncellsdthetaM;
    Track1.ncellsdphiP=aux_ncellsdphiP;
    Track1.ncellsdphiM=aux_ncellsdphiM;
    Track2.x_det0=aux_x2;
    Track2.y_det0=aux_y2;
    Track2.z_det0=aux_z2;
    Track2.phi=aux_phi2;
    Track2.theta=aux_theta2;
    Track2.chisquaxz=aux_chisquaxz2;
    Track2.chisquayz=aux_chisquayz2;
    Track2.err_phi=aux_err_phi2;
    Track2.err_theta=aux_err_theta2;
    Track3.x_det0=aux_x3;
    Track3.y_det0=aux_y3;
    Track3.z_det0=aux_z3;
    Track3.phi=aux_phi3;
    Track3.theta=aux_theta3;
    Track3.chisquaxz=aux_chisquaxz3;
    Track3.chisquayz=aux_chisquayz3;
    Track3.err_phi=aux_err_phi3;
    Track3.err_theta=aux_err_theta3;
    Tracking->Fill();
    Entries_Tree_cont++;
  }
}
```

```
      (void)sprintf(CharIn,"");
    }
    treeOK=0;
  }
  track->Write();
  track->Close();
  delete track;
  cout << "\nEntries of new Tree= " << Entries_Tree_cont << endl;
  cout << "Points over the maximum value (" << max_npoints << ")= " <<
   over_max_npoints_cont << endl;
  in.close();
}
void find_all_ncells(){
  Int_t aux_funcx=1000;
  Int_t aux_funcy=1000;
  for(Int_t i=0;i<4;i++){
    aux_funcx = SearchChannelx(Map_height_z[i],aux_phi1,aux_theta1);
    aux_funcy = SearchChannely(Map_height_z[i],aux_phi1,aux_theta1);
    if ((aux_funcx!=1000)&&(aux_funcy!=1000)){
      if (Coin_Map_hit_xz[i][aux_funcx]==1) aux_ncells++;
      if(Coin_Map_hit_yz[i][aux_funcy]==1) aux_ncells++;
    }
    aux_funcx = SearchChannelx(Map_height_z[i],aux_phi1+aux_err_phi1,
   aux_theta1);
    aux_funcy = SearchChannely(Map_height_z[i],aux_phi1+aux_err_phi1,
   aux_theta1);
    if ((aux_funcx!=1000)&&(aux_funcy!=1000)){
      if(Coin_Map_hit_xz[i][aux_funcx]==1) aux_ncellsdphiP++;
      if(Coin_Map_hit_yz[i][aux_funcy]==1) aux_ncellsdphiP++;
    }
    aux_funcx = SearchChannelx(Map_height_z[i],aux_phi1-aux_err_phi1,
   aux_theta1);
    aux_funcy = SearchChannely(Map_height_z[i],aux_phi1-aux_err_phi1,
   aux_theta1);
    if ((aux_funcx!=1000)&&(aux_funcy!=1000)){
      if(Coin_Map_hit_xz[i][aux_funcx]==1) aux_ncellsdphiM++;
      if(Coin_Map_hit_yz[i][aux_funcy]==1) aux_ncellsdphiM++;
    }
    aux_funcx = SearchChannelx(Map_height_z[i],aux_phi1,aux_theta1+
   aux_err_theta1);
    aux_funcy = SearchChannely(Map_height_z[i],aux_phi1,aux_theta1+
```

```
    aux_err_theta1);
     if ((aux_funcx!=1000)&&(aux_funcy!=1000)){
       if(Coin_Map_hit_xz[i][aux_funcx]==1) aux_ncellsdthetaP++;
       if(Coin_Map_hit_yz[i][aux_funcy]==1) aux_ncellsdthetaP++;
     }
     aux_funcx = SearchChannelx(Map_height_z[i],aux_phi1,aux_theta1-
    aux_err_theta1);
     aux_funcy = SearchChannely(Map_height_z[i],aux_phi1,aux_theta1-
    aux_err_theta1);
     if ((aux_funcx!=1000)&&(aux_funcy!=1000)){
       if(Coin_Map_hit_xz[i][aux_funcx]==1) aux_ncellsdthetaM++;
       if(Coin_Map_hit_yz[i][aux_funcy]==1) aux_ncellsdthetaM++;
     }
   }
}
Int_t SearchChannelx(Double_t axi_z,Double_t v_phi, Double_t v_theta){
   Double_t radii = tan(v_theta)*(axi_z-aux_z1);
   Double_t x     = radii*cos(v_phi) + aux_x1;
   // identifying the cell in x
   Int_t icell = 0;
   if (x>=0){
     while (1)
     {
       if ( ( (x > (icell*tile_width_ch) ) && (x <= ((icell+1)*tile_width_ch)
     ) ) || (icell==8) )  break;
       icell++;
     }
     if (icell < 8)
     {
       return icell;
     } else
     {
       return 1000;
     }
   }else return 1000;
}
Int_t SearchChannely(Double_t axi_z,Double_t v_phi, Double_t v_theta){
   Double_t radii = tan(v_theta)*(axi_z-aux_z1);
   Double_t y     = radii*sin(v_phi) + aux_y1;
   Int_t icell = 0;
   if (y>=0){
```

```
    while (1)
    {
      if ( ( (y > (icell*tile_width_ch) ) && (y <= ((icell+1)*tile_width_ch)
    ) ) || (icell==8) )  break;
      icell++;
    }
    if (icell < 8)
    {
      return = icell;
    } else
    {
      return = 1000;
    }
  } else return = 1000;
}
void MakePoints(Int_t max_npoints, Int_t min_num_detectors){
  Int_t aux_axi_z[10000];
  Double_t aux_x[8][10000], aux_y[8][10000], axi_z[10000];
  Double_t axi_x[10000], axi_y[10000], ex[10000], ey[10000], ez[10000];
  Int_t npoints=0;
  Int_t contaux_x[8]={0,0,0,0,0,0,0,0};
  Int_t contaux_y[8]={0,0,0,0,0,0,0,0};
  Int_t y=0;
  Int_t max=0;
  Double_t val_in=0;
  for (y=0;y<4;y++) {
    for (Int_t k=0;k<8;k++) {
      if((Coin_Map_hit_xz[y][k]!=0)){
        aux_x[contaux_x[y]][y] = k;
        contaux_x[y]=contaux_x[y]+1;
      }
      if((Coin_Map_hit_yz[y][k]!=0)){
        aux_y[contaux_y[y]][y] = k;
        contaux_y[y]=contaux_y[y]+1;
      }
    }
  }
  for ( y=0;y<4;y++) {
    for (Int_t k1=0; k1<contaux_x[y]; k1++) {
      for (Int_t k2=0; k2<contaux_y[y]; k2++) {
        axi_x[npoints]=aux_x[k1][y];
```

```
        axi_y[npoints]=aux_y[k2][y];
        aux_axi_z[npoints]=y;
        npoints++;
      }
    }
  }
  if (npoints<=max_npoints){
    for ( k=0;k<npoints;k++) {
      max = axi_x[k];
      val_in = tile_width_ch/2;
      ex[k] = val_in;
      ez[k] = tile_height_ch;
      for ( y=0;y<max;y++) {
        val_in = val_in + tile_width_ch;
      }
      axi_x[k]=val_in;
    }
    for ( k=0;k<npoints;k++) {
      max = axi_y[k];
      val_in = tile_width_ch/2;
      ey[k] = val_in;
      for ( y=0;y<max;y++) {
        val_in = val_in + tile_width_ch;
      }
      axi_y[k]=val_in;
    }
    Int_t cont_plan=0;
    axi_z[0]=Map_height_z[aux_axi_z[0]];
    for ( k=1;k<npoints;k++) {
      axi_z[k]=Map_height_z[aux_axi_z[k]];
      if (aux_axi_z[k]!=aux_axi_z[k-1]) cont_plan++;
    }
    aux_npoints = npoints;
    aux_ndet = cont_plan+1;
    if (cont_plan>min_num_detectors-2){
      //faz todas as combinações possiveis entre os pontos de forma
      //que haja amenas um ponto em cada plano do detector
      CombiPoints (axi_x,ex,axi_y,ey,axi_z,aux_axi_z,ez,npoints,cont_plan);
    }
  } else {
    over_max_npoints_cont++;
```

```
    treeOK = 0;
  }
}
void CombiPoints (Double_t axi_x[10000],Double_t ex[10000],Double_t axi_y
    [10000],Double_t ey[10000],Double_t axi_z[10000],Int_t aux_axi_z_int
    [10000], Double_t ez[10000], Int_t npoints, Int_t cont_plan ){
  Int_t i=0;
  Int_t j;
  Int_t cont_z_points[4]={0,0,0,0};
  Int_t k[4]={0,0,0,0};
  Double_t aux_axi_x[4];
  Double_t aux_ex[4];
  Double_t aux_axi_y[4];
  Double_t aux_ey[4];
  Double_t aux_ez[4];
  Double_t aux_axi_z[4];
  for ( j=1;j<=npoints;j++) {
    if (aux_axi_z_int[j]==aux_axi_z_int[j-1]){
      cont_z_points[i]=cont_z_points[i]+1;
    }else{
      cont_z_points[i]=cont_z_points[i]+1;
      i++;
    }
  }
  if (cont_plan==3){
    for ( k[3]=0;k[3]<cont_z_points[3];k[3]++){
      for ( k[2]=0;k[2]<cont_z_points[2];k[2]++){
        for ( k[1]=0;k[1]<cont_z_points[1];k[1]++){
          for ( k[0]=0;k[0]<cont_z_points[0];k[0]++){
            aux_axi_x[0]=axi_x[k[0]];
            aux_axi_x[1]=axi_x[cont_z_points[0]+k[1]];
            aux_axi_x[2]=axi_x[cont_z_points[0]+cont_z_points[1]+k[2]];
            aux_axi_x[3]=axi_x[cont_z_points[0]+cont_z_points[1]+
    cont_z_points[2]+k[3]];
            aux_axi_y[0]=axi_y[k[0]];
            aux_axi_y[1]=axi_y[cont_z_points[0]+k[1]];
            aux_axi_y[2]=axi_y[cont_z_points[0]+cont_z_points[1]+k[2]];
            aux_axi_y[3]=axi_y[cont_z_points[0]+cont_z_points[1]+
    cont_z_points[2]+k[3]];
            aux_axi_z[0]=axi_z[k[0]];
            aux_axi_z[1]=axi_z[cont_z_points[0]+k[1]];
```

```
            aux_axi_z[2]=axi_z[cont_z_points[0]+cont_z_points[1]+k[2]];
            aux_axi_z[3]=axi_z[cont_z_points[0]+cont_z_points[1]+
 cont_z_points[2]+k[3]];
            aux_ex[0]=ex[k[0]];
            aux_ex[1]=ex[cont_z_points[0]+k[1]];
            aux_ex[2]=ex[cont_z_points[0]+cont_z_points[1]+k[2]];
            aux_ex[3]=ex[cont_z_points[0]+cont_z_points[1]+cont_z_points[2]+
 k[3]];
            aux_ey[0]=ey[k[0]];
            aux_ey[1]=ey[cont_z_points[0]+k[1]];
            aux_ey[2]=ey[cont_z_points[0]+cont_z_points[1]+k[2]];
            aux_ey[3]=ey[cont_z_points[0]+cont_z_points[1]+cont_z_points[2]+
 k[3]];
            aux_ez[0]=ez[k[0]];
            aux_ez[1]=ez[cont_z_points[0]+k[1]];
            aux_ez[2]=ez[cont_z_points[0]+cont_z_points[1]+k[2]];
            aux_ez[3]=ez[cont_z_points[0]+cont_z_points[1]+cont_z_points[2]+
 k[3]];
            FittingPoints(aux_axi_x,aux_ex,aux_axi_y,aux_ey,aux_axi_z,aux_ez
 ,cont_plan+1);
          }
        }
      }
    }
  }else if(cont_plan==2){
    for (k[2]=0;k[2]<cont_z_points[2];k[2]++){
      for (k[1]=0;k[1]<cont_z_points[1];k[1]++){
        for (k[0]=0;k[0]<cont_z_points[0];k[0]++){
          aux_axi_x[0]=axi_x[k[0]];
          aux_axi_x[1]=axi_x[cont_z_points[0]+k[1]];
          aux_axi_x[2]=axi_x[cont_z_points[0]+cont_z_points[1]+k[2]];
          aux_axi_y[0]=axi_y[k[0]];
          aux_axi_y[1]=axi_y[cont_z_points[0]+k[1]];
          aux_axi_y[2]=axi_y[cont_z_points[0]+cont_z_points[1]+k[2]];
          aux_axi_z[0]=axi_z[k[0]];
          aux_axi_z[1]=axi_z[cont_z_points[0]+k[1]];
          aux_axi_z[2]=axi_z[cont_z_points[0]+cont_z_points[1]+k[2]];
          aux_ex[0]=ex[k[0]];
          aux_ex[1]=ex[cont_z_points[0]+k[1]];
          aux_ex[2]=ex[cont_z_points[0]+cont_z_points[1]+k[2]];
          aux_ey[0]=ey[k[0]];
```

```
            aux_ey[1]=ey[cont_z_points[0]+k[1]];
            aux_ey[2]=ey[cont_z_points[0]+cont_z_points[1]+k[2]];
            aux_ez[0]=ez[k[0]];
            aux_ez[1]=ez[cont_z_points[0]+k[1]];
            aux_ez[2]=ez[cont_z_points[0]+cont_z_points[1]+k[2]];
            FittingPoints(aux_axi_x,aux_ex,aux_axi_y,aux_ey,aux_axi_z,aux_ez,
    cont_plan+1);
          }
        }
      }
    }else if(cont_plan==1){
      for ( k[1]=0;k[1]<cont_z_points[1];k[1]++){
        for ( k[0]=0;k[0]<cont_z_points[0];k[0]++){
          aux_axi_x[0]=axi_x[k[0]];
          aux_axi_x[1]=axi_x[cont_z_points[0]+k[1]];
          aux_axi_y[0]=axi_y[k[0]];
          aux_axi_y[1]=axi_y[cont_z_points[0]+k[1]];
          aux_axi_z[0]=axi_z[k[0]];
          aux_axi_z[1]=axi_z[cont_z_points[0]+k[1]];
          aux_ex[0]=ex[k[0]];
          aux_ex[1]=ex[cont_z_points[0]+k[1]];
          aux_ey[0]=ey[k[0]];
          aux_ey[1]=ey[cont_z_points[0]+k[1]];
          aux_ez[0]=ez[k[0]];
          aux_ez[1]=ez[cont_z_points[0]+k[1]];
          FittingPoints(aux_axi_x,aux_ex,aux_axi_y,aux_ey,aux_axi_z,aux_ez,
    cont_plan+1);
        }
      }
    }
    //Coloca os valores na Tree
    treeOK=1;
}
//faz o fit de todas ads combinações de pontos (um ponto em cada plano do
    detector) fit plano zx e zy
void FittingPoints(Double_t axi_x[10000],Double_t ex[10000],Double_t axi_y
    [10000],Double_t ey[10000],Double_t axi_z[10000],Double_t ez[10000],
    Int_t npoints){
  aux_n_combi_tr++;
  xzp0=0;
  xzp1=0;
```

```
yzp0=0;
yzp1=0;
Err_xzp0=0;
Err_xzp1=0;
Err_yzp0=0;
Err_yzp1=0;
chisquaxz=0;
chisquayz=0;
Double_t phi_meas=0;
Double_t theta_meas=0;
Double_t err_phi_meas=0;
Double_t err_theta_meas=0;
GetParametersFit(axi_x,axi_y,axi_z,ex,ey,ez,npoints);
if (((fabs(chisquaxz)+fabs(chisquayz))/2)<((fabs(aux_chisquaxz1)+fabs(
 aux_chisquayz1))/2)){
  aux_x3=aux_x2;
  aux_y3=aux_y2;
  aux_z3=aux_z2;
  aux_phi3=aux_phi2;
  aux_theta3=aux_theta2;
  aux_chisquaxz3=aux_chisquaxz2;
  aux_chisquayz3=aux_chisquayz2;
  aux_err_phi3=aux_err_phi2;
  aux_err_theta3=aux_err_theta2;
  aux_x2=aux_x1;
  aux_y2=aux_y1;
  aux_z2=aux_z1;
  aux_phi2=aux_phi1;
  aux_theta2=aux_theta1;
  aux_chisquaxz2=aux_chisquaxz1;
  aux_chisquayz2=aux_chisquayz1;
  aux_err_phi2=aux_err_phi1;
  aux_err_theta2=aux_err_theta1;
  //acha o phi e theta por geometria do plano cartesiano
  Double_t delta_z = axi_z[npoints-1]-axi_z[0];
  Double_t err_delta_z = sqrt(pow(ez[npoints-1],2)+pow(ez[npoints-1],2));
  Double_t delta_x = delta_z*xzp1;
  Double_t err_delta_x = delta_x*(err_delta_z/delta_z+Err_xzp1/xzp1);
  Double_t delta_y = delta_z*yzp1;
  Double_t err_delta_y = delta_y*(err_delta_z/delta_z+Err_yzp1/yzp1);
  Double_t Rxy = sqrt(pow(delta_x,2)+pow(delta_y,2));
```

```
    Double_t err_Rxy = 1/(2*sqrt(pow(delta_x,2)+pow(delta_y,2)))*( sqrt( pow
    (2*delta_x*err_delta_x,2) + pow(2*delta_y*err_delta_y,2) ) );
    phi_meas = atan2(delta_y,delta_x);
    err_phi_meas = sqrt(pow(delta_x,2)*pow(err_delta_y,2)+pow(delta_y,2)*pow
    (err_delta_x,2))/(pow(delta_y,2)+pow(delta_x,2));
    theta_meas = atan2(Rxy,delta_z);
    err_theta_meas = sqrt(pow(delta_z,2)*pow(err_Rxy,2)+pow(Rxy,2)*pow(
    err_delta_z,2))/(pow(Rxy,2)+pow(delta_z,2));
    aux_x1=xzp1*axi_z[0]+xzp0;
    aux_y1=yzp1*axi_z[0]+yzp0;
    aux_z1=axi_z[0];
    aux_phi1=phi_meas;
    aux_theta1=theta_meas;
    aux_chisquaxz1=chisquaxz;
    aux_chisquayz1=chisquayz;
    aux_err_phi1=err_phi_meas;
    aux_err_theta1=err_theta_meas;
  }
}
void GetParametersFit(Double_t x2[4], Double_t y2[4], Double_t z2[4],
    Double_t ex[4],Double_t ey[4],Double_t ez[4],int cont){
  TF1 *f2 = new TF1("f2", "pol1", z2[0], z2[cont-1]);
  TGraphErrors *dt3 = new TGraphErrors(cont,z2,x2,ez,ex);
  TGraphErrors *dt4 = new TGraphErrors(cont,z2,y2,ez,ey);
  dt3->Fit("f2","RQ");
  xzp0 = f2->GetParameter(0);
  xzp1 = f2->GetParameter(1);
  Err_xzp0 = f2->GetParError(0);
  Err_xzp1 = f2->GetParError(1);
  chisquaxz=f2->GetChisquare();
  dt4->Fit("f2","RQ");
  yzp0=f2->GetParameter(0);
  yzp1=f2->GetParameter(1);
  Err_yzp0 = f2->GetParError(0);
  Err_yzp1 = f2->GetParError(1);
  chisquayz=f2->GetChisquare();
  delete f2;
}
```

Apêndice L

Código de Análise de Dados do Detector CREAT3

```
//////////////////////////////////////////////////////////
// Código de Análise de Dados do Detector CREAT3
// Author: Leonardo Chaves Ruiz Guedes (leocrg@cbpf.br)
//////////////////////////////////////////////////////////

#include <stdio.h>
#include "TGraph.h"
#include "TAxis.h"
#include "TCanvas.h"
#include "TLine.h"
#include <time.h>
#include <TGraph2D.h>
#include <TRandom.h>
#include <TStyle.h>
#include <TF2.h>
#include <TH1.h>
#include "TMath.h"
#include "TFile.h"
#include "TTree.h"

Double_t pi = atan(1)*4;
typedef struct {
  Double_t x_det0, y_det0, z_det0, phi, theta, chisquaxz, chisquayz, err_phi
    , err_theta;
```

```
  Int_t ncells, ncellsdthetaP, ncellsdthetaM, ncellsdphiP, ncellsdphiM;
} tracktype1;
typedef struct {
  Double_t x_det0, y_det0, z_det0, phi, theta, chisquaxz, chisquayz, err_phi
    , err_theta;
} tracktype2;
TH1F *hdifphi2det;
TH1F *hdiftheta2det;
TH1F *hdifphi3det;
TH1F *hdiftheta3det;
TH1F *hdifphi4det;
TH1F *hdiftheta4det;
TH1F *hdifphi;
TH1F *hdiftheta;
TH1F *htheta_sim_rec;
TH1F *hphi_sim_rec;
TH1F *Acc_sim_theta2det;
TH1F *Acc_sim_phi2det;
TH1F *Acc_sim_theta3det;
TH1F *Acc_sim_phi3det;
TH1F *Acc_sim_theta4det;
TH1F *Acc_sim_phi4det;
TH1F *Acc_sim_theta;
TH1F *Acc_sim_phi;
TH1F *htheta_prod;
TH1F *hphi_prod;
TH1F *hphi_rec;
TH1F *htheta_rec;
TH1F *htheta_prod2det;
TH1F *hphi_prod2det;
TH1F *hphi_rec2det;
TH1F *htheta_rec2det;
TH1F *htheta_prod3det;
TH1F *hphi_prod3det;
TH1F *hphi_rec3det;
TH1F *htheta_rec3det;
TH1F *htheta_prod4det;
TH1F *hphi_prod4det;
TH1F *hphi_rec4det;
TH1F *htheta_rec4det;
Int_t NhitEff[64];
```

```
Int_t NnormEff[64];
TH1F *hphiprod_gen;
TH1F *hthetaprod_gen;
//Mapa de posição de cada canal
Int_t Coin_Map_pos_xz[4][8];
Int_t Coin_Map_pos_yz[4][8];
//largura de cada canal
Double_t tile_width_ch;
Double_t tile_height_ch;
Double_t Map_height_z[4];
Double_t MaxFit=90;
Int_t ModelProd=0;
Int_t nbin;
Float_t ErrDiv;
char Reg_DAC0[] = "01C2";
char Reg_DAC1[] = "0000";
char USB_Port[30] = "0";
char Reg_Ctest[16] = "8080202008080202";
char Reg_N_gate[30] = "0A";
char Reg_word_0[30] = "A8";
char Reg_word_1[30] = "3B";
char Reg_word_2[30] = "C6";
char Reg_word_3[30] = "23";
char Reg_word_4[30] = "D9";
char Reg_word_5[30] = "02";
char Reg_word_6[30] = "";
char Reg_OTHERS[30] = "";
Double_t TimeStepCount=0;
Int_t hit_vector[64];

Float_t ErrDiv(Float_t a, Float_t da, Float_t b, Float_t db)
{
  /// SIGX =SQRT( { D_A**2/A**2 + D_B**2/B**2)} * X**2)
  return ErrDiv = (1/b)*(1/b)*TMath::Sqrt((b*b*da*da) + (a*a*db*db));
}
void analyse_simulation_tree( Int_t nevents=-1, Int_t minnplans=2, char*
    condition1= ">=" , Int_t auxnbin=100, char* input_Treefile = "Acquisition
    /DefautFile.root",Double_t  AuxMaxFit=90, Double_t chisqua=0, char*
    condition2= ">=",  Int_t maxn_points=50){
  for (Int_t i=0;i<64;i++){
    NhitEff[i]=0;
```

```
  NnormEff[i]=0;
}
nbin=auxnbin;
MaxFit= AuxMaxFit;
hdifphi2det = new TH1F("hdifphi2det","phi dif 2det",nbin,-60,60);
hdiftheta2det = new TH1F("hdiftheta2det","theta dif 2det",nbin,-10,10);
hdifphi3det = new TH1F("hdifphi3det","phi dif 3det",nbin,-60,60);
hdiftheta3det = new TH1F("hdiftheta3det","theta dif 3det",nbin,-10,10);
hdifphi4det = new TH1F("hdifphi4det","phi dif 4det",nbin,-60,60);
hdiftheta4det = new TH1F("hdiftheta4det","theta dif 4det",nbin,-10,10);

hdifphi = new TH1F("hdifphi","phi dif",nbin,-60,60);
hdiftheta = new TH1F("hdiftheta","theta dif",nbin,-10,10);
htheta_sim_rec2det = new TH1F("htheta_sim_rec2det","h theta simulado
 Recostruido 2det",nbin,0,90);
hphi_sim_rec2det = new TH1F("hphi_sim_rec2det","h phi simulado Recostruido
  2det",nbin,-180,180);
htheta_sim_rec3det = new TH1F("htheta_sim_rec3det","h theta simulado
 Recostruido 3det",nbin,0,90);
hphi_sim_rec3det = new TH1F("hphi_sim_rec3det","h phi simulado Recostruido
  3det",nbin,-180,180);
htheta_sim_rec4det = new TH1F("htheta_sim_rec4det","h theta simulado
 Recostruido 4det",nbin,0,90);
hphi_sim_rec4det = new TH1F("hphi_sim_rec4det","h phi simulado Recostruido
  4det",nbin,-180,180);
htheta_sim_rec = new TH1F("htheta_sim_rec","h theta simulado Recostruido",
 nbin,0,90);
hphi_sim_rec = new TH1F("hphi_sim_rec","h phi simulado Recostruido",nbin
 ,-180,180);
Acc_sim_theta2det = new TH1F("Acc_sim_theta2det","Acc_geo_sim 2det",nbin
 ,0,90);
Acc_sim_phi2det = new TH1F("Acc_sim_phi2det","Acc_geo_sim 2det",nbin
 ,-180,180);
Acc_sim_theta3det = new TH1F("Acc_sim_theta3det","Acc_geo_sim 3det",nbin
 ,0,90);
Acc_sim_phi3det = new TH1F("Acc_sim_phi3det","Acc_geo_sim 3det",nbin
 ,-180,180);
Acc_sim_theta4det = new TH1F("Acc_sim_theta4det","Acc_geo_sim 4det",nbin
 ,0,90);
Acc_sim_phi4det = new TH1F("Acc_sim_phi4det","Acc_geo_sim 4det",nbin
 ,-180,180);
```

```
Acc_sim_theta = new TH1F("Acc_sim_theta","Acc_geo_sim",nbin,0,90);
Acc_sim_phi = new TH1F("Acc_sim_phi","Acc_geo_sim",nbin,-180,180);
htheta_prod2det = new TH1F("htheta_prod2det","Theta_Prod 2det",nbin,0,90);
hphi_prod2det = new TH1F("hphi_prod2det","Phi_Prod 2det",nbin,-180,180);
htheta_prod3det = new TH1F("htheta_prod3det","Theta_Prod 3det",nbin,0,90);
hphi_prod3det = new TH1F("hphi_prod3det","Phi_Prod 3det",nbin,-180,180);
htheta_prod4det = new TH1F("htheta_prod4det","Theta_Prod 4det",nbin,0,90);
hphi_prod4det = new TH1F("hphi_prod4det","Phi_Prod 4det",nbin,-180,180);
htheta_prod = new TH1F("htheta_prod","Theta_Prod",nbin,0,90);
hphi_prod = new TH1F("hphi_prod","Phi_Prod",nbin,-180,180);
hphi_rec2det = new TH1F("hphi_rec2det","phi 2det",nbin,-180,180);
htheta_rec2det = new TH1F("htheta_rec2det","theta 2det",nbin,0,90);
hphi_rec3det = new TH1F("hphi_rec3det","phi 3det",nbin,-180,180);
htheta_rec3det = new TH1F("htheta_rec3det","theta 3det",nbin,0,90);
hphi_rec4det = new TH1F("hphi_rec4det","phi 4det",nbin,-180,180);
htheta_rec4det = new TH1F("htheta_rec4det","theta 4det",nbin,0,90);
hphi_rec = new TH1F("hphi_rec","phi",nbin,-180,180);
htheta_rec = new TH1F("htheta_rec","theta",nbin,0,90);
hphi_rec = new TH1F("hphi_rec","phi",nbin,-180,180);
htheta_rec = new TH1F("htheta_rec","theta",nbin,0,90);
hdifphi2det->Sumw2();
hdiftheta2det->Sumw2();
hdifphi3det->Sumw2();
hdiftheta3det->Sumw2();
hdifphi4det->Sumw2();
hdiftheta4det->Sumw2();
hdifphi->Sumw2();
hdiftheta->Sumw2();
htheta_sim_rec->Sumw2();
hphi_sim_rec->Sumw2();
Acc_sim_phi2det->Sumw2();
Acc_sim_theta2det->Sumw2();
Acc_sim_phi3det->Sumw2();
Acc_sim_theta3det->Sumw2();
Acc_sim_phi4det->Sumw2();
Acc_sim_theta4det->Sumw2();
Acc_sim_theta->Sumw2();
Acc_sim_phi->Sumw2();
hphi_rec2det ->Sumw2();
htheta_rec2det ->Sumw2();
hphi_rec3det ->Sumw2();
```

```
htheta_rec3det ->Sumw2();
hphi_rec4det ->Sumw2();
htheta_rec4det ->Sumw2();

hphi_rec->Sumw2();
htheta_rec->Sumw2();
htheta_prod->Sumw2();
hphi_prod->Sumw2();

char genrootfile[1024]="";
strncpy(genrootfile,input_Treefile,strlen(input_Treefile)-5);
sprintf(genrootfile,"%s_gen.root",genrootfile);
char SetupInfoFile[1024]="";
strncpy(SetupInfoFile,input_Treefile,strlen(input_Treefile)-4);
sprintf(SetupInfoFile,"%scfg",SetupInfoFile);
cout<< SetupInfoFile << endl;
static tracktype1 Track1;
static tracktype2 Track2, Track3;
Int_t n_tracks, n_points;
Double_t phi_sim, theta_sim;
Int_t n_det;

Int_t nentries;
if (ReadSetupInfo(SetupInfoFile)==0){
  hdifphi2det->Reset();
  hdiftheta2det->Reset();
  hdifphi3det->Reset();
  hdiftheta3det->Reset();
  hdifphi4det->Reset();
  hdiftheta4det->Reset();
  hdifphi2det -> Reset();
  hdiftheta2det -> Reset();
  htheta_sim_rec2det -> Reset();
  hphi_sim_rec2det -> Reset();
  hdifphi3det ->Reset();
  hdiftheta3det ->Reset();
  htheta_sim_rec3det ->Reset();
  hphi_sim_rec3det ->Reset();
  hdifphi4det ->Reset();
  hdiftheta4det ->Reset();
  htheta_sim_rec4det ->Reset();
```

```
hphi_sim_rec4det ->Reset();
hdifphi->Reset();
hdiftheta->Reset();
htheta_sim_rec->Reset();
hphi_sim_rec->Reset();
Acc_sim_phi2det->Reset();
Acc_sim_theta2det->Reset();
Acc_sim_phi3det->Reset();
Acc_sim_theta3det->Reset();
Acc_sim_phi4det->Reset();
Acc_sim_theta4det->Reset();
htheta_prod2det ->Reset();
hphi_prod2det ->Reset();
htheta_prod3det ->Reset();
hphi_prod3det ->Reset();
htheta_prod4det ->Reset();
hphi_prod4det ->Reset();
Acc_sim_theta->Reset();
Acc_sim_phi->Reset();
hphi_rec2det ->Reset();
htheta_rec2det ->Reset();
hphi_rec3det ->Reset();
htheta_rec3det ->Reset();
hphi_rec4det ->Reset();
htheta_rec4det ->Reset();
hphi_rec->Reset();
htheta_rec->Reset();
htheta_prod->Reset();
hphi_prod->Reset();
TFile *file = new TFile(input_Treefile,"UPDATE");
TTree *j = (TTree*)file->Get("Tracking");

TFile *filegen = new TFile(genrootfile,"UPDATE");

char hphiprod[1024]="";
sprintf(hphiprod,"hphiprod_%d",nbin);
char hthetaprod[1024]="";
sprintf(hthetaprod,"hthetaprod_%d",nbin);

hphiprod_gen = (TH1F*)filegen->Get(hphiprod);
hthetaprod_gen = (TH1F*)filegen->Get(hthetaprod);
```

```
 j->SetBranchAddress("n_tracks",&n_tracks);
 j->SetBranchAddress("n_points",&n_points);
 j->SetBranchAddress("phi_sim",&phi_sim);
 j->SetBranchAddress("theta_sim",&theta_sim);
 j->SetBranchAddress("n_det",&n_det);
 j->SetBranchAddress("hit_vector",&hit_vector);
 j->SetBranchAddress("Track1",&Track1);

 if ((nevents==-1)||(nevents>(Int_t)j->GetEntries())) nentries = (Int_t)j
->GetEntries();
 else nentries = nevents;
 printf("Entries= %d\n",nentries);

 for (Int_t i=0;i<nentries;i++) {
   j->GetEntry(i);
   if((compare_int_variables(n_det, condition1, minnplans) == 1)&&(
n_points<=maxn_points)&&(compare_double_variables(Track1.chisquaxz,
condition2, chisqua) == 1)&&(compare_double_variables(Track1.chisquayz,
condition2, chisqua) == 1)){
     //preenche os histogramas com os angulos de thi e theta convertidos
de radianos em graus
     if (n_det==2) {
       hdifphi2det -> Fill(phi_sim*180/pi - Track1.phi*180/pi);
       hdiftheta2det -> Fill(theta_sim*180/pi-Track1.theta*180/pi);
       htheta_sim_rec2det -> Fill(Double_t(theta_sim*180/pi));
       htheta_rec2det -> Fill(Double_t(Track1.theta*180/pi));
       hphi_sim_rec2det -> Fill(Double_t(phi_sim*180/pi));
       hphi_rec2det -> Fill(Double_t(Track1.phi*180/pi));
       Acc_sim_theta2det -> Divide(htheta_sim_rec2det,hthetaprod_gen);
       Acc_sim_phi2det -> Divide(hphi_sim_rec2det,hphiprod_gen);
       htheta_prod2det -> Divide(htheta_rec2det,Acc_sim_theta2det);
       hphi_prod2det -> Divide(hphi_rec2det,Acc_sim_phi2det);
     }
     if (n_det==3) {
       hdifphi3det -> Fill(phi_sim*180/pi - Track1.phi*180/pi);
       hdiftheta3det -> Fill(theta_sim*180/pi-Track1.theta*180/pi);
       htheta_sim_rec3det -> Fill(Double_t(theta_sim*180/pi));
       htheta_rec3det -> Fill(Double_t(Track1.theta*180/pi));
       hphi_sim_rec3det -> Fill(Double_t(phi_sim*180/pi));
       hphi_rec3det -> Fill(Double_t(Track1.phi*180/pi));
       Acc_sim_theta3det -> Divide(htheta_sim_rec3det,hthetaprod_gen);
```

```
        Acc_sim_phi3det -> Divide(hphi_sim_rec3det,hphiprod_gen);
        htheta_prod3det -> Divide(htheta_rec3det,Acc_sim_theta3det);
        hphi_prod3det -> Divide(hphi_rec3det,Acc_sim_phi3det);
      }
      if (n_det==4) {
        hdifphi4det -> Fill(phi_sim*180/pi - Track1.phi*180/pi);
        hdiftheta4det -> Fill(theta_sim*180/pi-Track1.theta*180/pi);
        htheta_sim_rec4det -> Fill(Double_t(theta_sim*180/pi));
        htheta_rec4det -> Fill(Double_t(Track1.theta*180/pi));
        hphi_sim_rec4det -> Fill(Double_t(phi_sim*180/pi));
        hphi_rec4det -> Fill(Double_t(Track1.phi*180/pi));
        Acc_sim_theta4det -> Divide(htheta_sim_rec4det,hthetaprod_gen);
        Acc_sim_phi4det -> Divide(hphi_sim_rec4det,hphiprod_gen);
        htheta_prod4det -> Divide(htheta_rec4det,Acc_sim_theta4det);
        hphi_prod4det -> Divide(hphi_rec4det,Acc_sim_phi4det);
      }
      hdifphi -> Fill(phi_sim*180/pi - Track1.phi*180/pi);
      hdiftheta -> Fill(theta_sim*180/pi-Track1.theta*180/pi);
      htheta_sim_rec -> Fill(Double_t(theta_sim*180/pi));
      htheta_rec -> Fill(Double_t(Track1.theta*180/pi));
      hphi_sim_rec -> Fill(Double_t(phi_sim*180/pi));
      hphi_rec -> Fill(Double_t(Track1.phi*180/pi));

      Acc_sim_theta -> Divide(htheta_sim_rec,hthetaprod_gen);
      Acc_sim_phi -> Divide(hphi_sim_rec,hphiprod_gen);
      htheta_prod -> Divide(htheta_rec,Acc_sim_theta);
      hphi_prod -> Divide(hphi_rec,Acc_sim_phi);
      GetEffParameters(Track1.phi,Track1.theta,Track1.x_det0,Track1.y_det0
  ,Track1.z_det0);

    }
   }
   Simulation_Results(ModelProd);
  }
}
void GetEffParameters(Double_t aux_phi, Double_t aux_theta, Double_t aux_x1,
    Double_t aux_y1, Double_t aux_z1){
  Int_t aux_funcx=1000, aux_funcy=1000 ;
  for(Int_t i=0;i<4;i++){
   aux_funcx = SearchChannelx(Map_height_z[i],aux_phi,aux_theta,aux_x1,
   aux_z1);
```

```
    aux_funcy = SearchChannely(Map_height_z[i],aux_phi,aux_theta,aux_y1,
    aux_z1);
    if ((aux_funcx!=1000)&&(aux_funcy!=1000)){
      if (hit_vector[Coin_Map_pos_xz[i][aux_funcx]]==1) NhitEff[
    Coin_Map_pos_xz[i][aux_funcx]]++;
      NnormEff[Coin_Map_pos_xz[i][aux_funcx]]++;
      if (hit_vector[Coin_Map_pos_yz[i][aux_funcy]]==1) NhitEff[
    Coin_Map_pos_yz[i][aux_funcy]]++;
      NnormEff[Coin_Map_pos_yz[i][aux_funcy]]++;
    }
  }
}
Int_t SearchChannelx(Double_t axi_z,Double_t v_phi, Double_t v_theta,
    Double_t aux_x1, Double_t aux_z1){
  Double_t radii = tan(v_theta)*(axi_z-aux_z1);
  Double_t x     = radii*cos(v_phi) + aux_x1;
  // identifying the cell in x
  Int_t icell = 0;
  if (x>=0){
    while (1)
    {
      if ( ( (x > (icell*tile_width_ch) ) && (x <= ((icell+1)*tile_width_ch)
    ) ) || (icell==8) )  break;
      icell++;
    }
    if (icell < 8)
    {
      return icell;
    } else
    {
      return 1000;
    }
  }else return 1000;
}
Int_t SearchChannely(Double_t axi_z,Double_t v_phi, Double_t v_theta,
    Double_t aux_y1, Double_t aux_z1){
  Double_t radii = tan(v_theta)*(axi_z-aux_z1);
  Double_t y     = radii*sin(v_phi) + aux_y1;
  // identifying the cell in y
  Int_t icell = 0;
  if (y>=0){
```

```
    while (1)
    {
      if ( ( (y > (icell*tile_width_ch) ) && (y <= ((icell+1)*tile_width_ch)
    ) ) || (icell==8) )  break;
      icell++;
    }
    if (icell < 8)
    {
      return = icell;
    } else
    {
      return = 1000;
    }
  } else return = 1000;
}


Int_t compare_int_variables(Int_t var1, char* condition, Int_t var2){
  Int_t flagcond=0;
  switch(condition){
    case ">=":
    if((var1>=var2)) flagcond = 1;
    break;
    case "<=":
    if((var1<=var2)) flagcond = 1;
    break;
    case ">":
    if((var1>var2)) flagcond = 1;
    break;
    case "<":
    if((var1<var2)) flagcond = 1;
    break;
    case "==":
    if((var1==var2)) flagcond = 1;
    break;
    case "!=":
    if((var1!=var2)) flagcond = 1;
    break;
  }
  return flagcond;
}
```

```
Int_t compare_double_variables(Double_t var1, char* condition, Double_t var2
    ){
  Int_t flagcond=0;
  switch(condition){
    case ">=":
    if((var1>=var2)) flagcond = 1;
    break;
    case "<=":
    if((var1<=var2)) flagcond = 1;
    break;
    case ">":
    if((var1>var2)) flagcond = 1;
    break;
    case "<":
    if((var1<var2)) flagcond = 1;
    break;
    case "==":
    if((var1==var2)) flagcond = 1;
    break;
    case "!=":
    if((var1!=var2)) flagcond = 1;
    break;
  }
  return flagcond;
}

void Simulation_Results(Int_t ModelProd){
  TF1 *gcostheta, *fitrec;
  gStyle->SetOptFit(1111);
  switch(ModelProd){
    case 0: gcostheta = new TF1("gcostheta","[0]*pow(cos(x*pi/180),[1])*sin(
   x*pi/180)*cos(x*pi/180)",0,90);
    fitrec = new TF1("fitrec","[0]*pow(cos(x*pi/180),[1])*sin(x*pi/180)*cos(
   x*pi/180)",0,MaxFit);
    break;
    case 1: gcostheta = new TF1("gcostheta","[0]*cos([1]*x*pi/180)",0,90);
    fitrec = new TF1("fitrec","[0]*cos([1]*x*pi/180)",0,MaxFit);
    break;
    case 2: gcostheta = new TF1("gcostheta","pol0",0,90);
    fitrec = new TF1("fitrec","pol0",0,MaxFit);
    break;
```

```
}
if (ModelProd!=2) gcostheta->SetParameters(1,2);

TCanvas *c2 = new TCanvas("c2","theta",200,10,700,500);
c2->SetFillColor(0);
c2->GetFrame()->SetFillColor(6);
c2->GetFrame()->SetBorderSize(10);
c2->GetFrame()->SetBorderMode(3);
c2->Divide(2,3);

c2->cd(1);
hthetaprod_gen->SetLineWidth(3);
hthetaprod_gen->SetMinimum(0);
hthetaprod_gen->Fit("gcostheta","R");
c2->cd(2);
htheta_sim_rec->SetLineWidth(3);
htheta_sim_rec->SetMinimum(0);
htheta_sim_rec->Draw("HIST");
c2->cd(3);
htheta_rec->SetLineWidth(3);
htheta_rec->SetMinimum(0);
htheta_rec->Draw("SAMEHIST");
c2->cd(4);
hdiftheta->SetLineWidth(3);
hdiftheta->SetMinimum(0);
hdiftheta->Draw("HIST");

THStack *hsdiftheta = new THStack("hsdiftheta","");
hdiftheta4det->SetFillColor(kGreen);
hsdiftheta->Add(hdiftheta4det);
hdiftheta3det->SetFillColor(kYellow);
hsdiftheta->Add(hdiftheta3det);
hdiftheta2det->SetFillColor(kRed);
hsdiftheta->Add(hdiftheta2det);
hsdiftheta->Draw("SAMEHIST");
TLegend *leg = new TLegend(0.1,0.7,0.4,0.9);
leg->AddEntry(hdiftheta2det,"2 Detectores","f");
leg->AddEntry(hdiftheta3det,"3 Detectores","f");
leg->AddEntry(hdiftheta4det,"4 Detectores","f");
leg->AddEntry(hdiftheta,"Total","l");
leg->Draw();
```

```
c2->cd(5);
Acc_sim_theta->SetLineWidth(3);
Acc_sim_theta->SetMinimum(0);
Acc_sim_theta->Draw("HIST");
THStack *hsacctheta = new THStack("hsacctheta","");
Acc_sim_theta4det->SetFillColor(kGreen);
hsacctheta->Add(Acc_sim_theta4det);
Acc_sim_theta3det->SetFillColor(kYellow);
hsacctheta->Add(Acc_sim_theta3det);
Acc_sim_theta2det->SetFillColor(kRed);
hsacctheta->Add(Acc_sim_theta2det);
hsacctheta->Draw("SAMEHIST");
TLegend *leg3 = new TLegend(0.48,0.7,0.78,0.9); //(43,0.7,76,1)
leg3->AddEntry(Acc_sim_theta2det,"2 Detectores","f");
leg3->AddEntry(Acc_sim_theta3det,"3 Detectores","f");
leg3->AddEntry(Acc_sim_theta4det,"4 Detectores","f");
leg3->AddEntry(Acc_sim_theta,"Total","l");
leg3->Draw();
c2->cd(6);
htheta_prod->SetLineWidth(3);
htheta_prod->SetMinimum(0);
htheta_prod->Fit("fitrec","R");
TCanvas *c3 = new TCanvas("c3","Phi",200,10,700,500);
c3->SetFillColor(0);
c3->GetFrame()->SetFillColor(6);
c3->GetFrame()->SetBorderSize(10);
c3->GetFrame()->SetBorderMode(3);
c3->Divide(2,3);

c3->cd(1);
hphiprod_gen->SetLineWidth(3);
hphiprod_gen->SetMinimum(0);
hphiprod_gen->Draw("HIST");
c3->cd(2);
hphi_sim_rec->SetLineWidth(3);
hphi_sim_rec->SetMinimum(0);
hphi_sim_rec->Draw("HIST");
c3->cd(3);
hphi_rec->SetLineWidth(3);
hphi_rec->SetMinimum(0);
```

```
hphi_rec->Draw("HIST");

c3->cd(4);
hdifphi->SetLineWidth(3);
hdifphi->SetMinimum(0);
hdifphi->Draw("HIST");
THStack *hsdifphi = new THStack("hsdifphi","");
hdifphi4det->SetFillColor(kGreen);
hsdifphi->Add(hdifphi4det);
hdifphi3det->SetFillColor(kYellow);
hsdifphi->Add(hdifphi3det);
hdifphi2det->SetFillColor(kRed);
hsdifphi->Add(hdifphi2det);
hsdifphi->Draw("SAMEHIST");

TLegend *leg2 = new TLegend(0.1,0.7,0.4,0.9);
leg2->AddEntry(hdifphi2det,"2 Detectores","f");
leg2->AddEntry(hdifphi3det,"3 Detectores","f");
leg2->AddEntry(hdifphi4det,"4 Detectores","f");
leg2->AddEntry(hdifphi,"Total","l");
leg2->Draw();

c3->cd(5);
Acc_sim_phi->SetLineWidth(3);
Acc_sim_phi->SetMinimum(0);
Acc_sim_phi->SetMaximum(0.1);
Acc_sim_phi->Draw("HIST");
THStack *hsaccphi = new THStack("hsaccphi","");
Acc_sim_phi4det->SetFillColor(kGreen);
hsaccphi->Add(Acc_sim_phi4det);
Acc_sim_phi3det->SetFillColor(kYellow);
hsaccphi->Add(Acc_sim_phi3det);
Acc_sim_phi2det->SetFillColor(kRed);
hsaccphi->Add(Acc_sim_phi2det);
hsaccphi->Draw("SAMEHIST");

TLegend *leg4 = new TLegend(0.48,0.7,0.78,0.9); //(43,0.7,76,1)
leg4->AddEntry(Acc_sim_phi2det,"2 Detectores","f");
leg4->AddEntry(Acc_sim_phi3det,"3 Detectores","f");
leg4->AddEntry(Acc_sim_phi4det,"4 Detectores","f");
leg4->AddEntry(Acc_sim_phi,"Total","l");
```

```
  leg4->Draw();

  c3->cd(6);
  hphi_prod->SetLineWidth(3);
  hphi_prod->SetMinimum(0);
  hphi_prod->Draw("HIST");
  for (Int_t i=0;i<64;i++) {
    if(NnormEff[i]>0) cout << "Eff["<< i <<"]= " << ((Double_t)NhitEff[i])
    /((Double_t)NnormEff[i]) << endl;// << " NhitEff["<< i <<"]= " << NhitEff
    [i] << " NnormEff["<< i <<"]= " << NnormEff[i] << endl;
     else  cout << "Eff["<< i <<"]= No events" << endl;
  }
}
void analyse_measurement_tree( Int_t nevents=-1, Int_t minnplans=2,char*
    condition= ">", Int_t maxn_points=50 , char* input_Treefile = "
    Acquisition/DefautFile.root"){
  Int_t time_register=0;
  Int_t TimeCountOld=0;
  Int_t deltaTime=0;
  Int_t Flux;
  Int_t ContEv;
  char SetupInfoFile[1024]="";
  strncpy(SetupInfoFile,input_Treefile,strlen(input_Treefile)-4);
  sprintf(SetupInfoFile,"%scfg",SetupInfoFile);
  static tracktype1 Track1;
  static tracktype2 Track2, Track3;
  Int_t n_tracks, n_points;
  Double_t phi_sim, theta_sim;
  Int_t n_det;
  Int_t time_register;
  Int_t nentries;
  if (ReadSetupInfo(SetupInfoFile)==0){
    TFile *file = new TFile(input_Treefile,"UPDATE");
    TTree *j = (TTree*)file->Get("Tracking");

    j->SetBranchAddress("n_tracks",&n_tracks);
    j->SetBranchAddress("n_points",&n_points);
    j->SetBranchAddress("phi_sim",&phi_sim);
    j->SetBranchAddress("theta_sim",&theta_sim);
    j->SetBranchAddress("n_det",&n_det);
    j->SetBranchAddress("time_register",&time_register);
```

```
 j->SetBranchAddress("Track1",&Track1);

 hphi_rec->Reset();
 htheta_rec->Reset();
 if ((nevents==-1)||(nevents>(Int_t)j->GetEntries())) nentries = (Int_t)j
->GetEntries();
 else nentries = nevents;
 printf("Entries= %d\n",nentries);

 for (Int_t i=0;i<nentries;i++) {
   j->GetEntry(i);
   switch(condition){
     case ">=":
     if((n_det>=minnplans)&&(n_points<=maxn_points)){
       //preenche os histogramas com os angulos de thi e theta
convertidos de radianos em graus
       hphi_rec -> Fill((Track1.phi*180/pi));
       htheta_rec -> Fill((Track1.theta*180/pi));
       if (i!=0){
         if(TimeCountOld>=time_register) deltaTime = 65535-TimeCountOld +
 time_register;
         else  deltaTime = time_register - TimeCountOld;
         Flux=Flux+deltaTime;
       }
       TimeCountOld = time_register;
       ContEv++;
     }
     break;
     case "<=":
     if((n_det<=minnplans)&&(n_points<=maxn_points)){
       //preenche os histogramas com os angulos de thi e theta
convertidos de radianos em graus
       hphi_rec -> Fill((Track1.phi*180/pi));
       htheta_rec -> Fill((Track1.theta*180/pi));
       if (i!=0){
         if(TimeCountOld>=time_register) deltaTime = 65535-TimeCountOld +
 time_register;
         else  deltaTime = time_register - TimeCountOld;
         Flux=Flux+deltaTime;
       }
```

```
        TimeCountOld = time_register;
        ContEv++;
      }
      break;
      case ">":
      if((n_det>minnplans)&&(n_points<=maxn_points)){
        //preenche os histogramas com os angulos de thi e theta
convertidos de radianos em graus
        hphi_rec -> Fill((Track1.phi*180/pi));
        htheta_rec -> Fill((Track1.theta*180/pi));
        if (i!=0){
          if(TimeCountOld>=time_register) deltaTime = 65535-TimeCountOld +
 time_register;
          else  deltaTime = time_register - TimeCountOld;
          Flux=Flux+deltaTime;
        }
        TimeCountOld = time_register;
        ContEv++;
      }
      break;
      case "<":
      if((n_det<minnplans)&&(n_points<=maxn_points)){
        //preenche os histogramas com os angulos de thi e theta
convertidos de radianos em graus
        hphi_rec -> Fill((Track1.phi*180/pi));
        htheta_rec -> Fill((Track1.theta*180/pi));
        if (i!=0){
          if(TimeCountOld>=time_register) deltaTime = 65535-TimeCountOld +
 time_register;
          else  deltaTime = time_register - TimeCountOld;
          Flux=Flux+deltaTime;
        }
        TimeCountOld = time_register;
        ContEv++;
      }
      break;
      case "==":
      if((n_det==minnplans)&&(n_points<=maxn_points)){
        hphi_rec -> Fill((Track1.phi*180/pi));
        htheta_rec -> Fill((Track1.theta*180/pi));
        if (i!=0){
```

```
            if(TimeCountOld>=time_register) deltaTime = 65535-TimeCountOld +
    time_register;
            else  deltaTime = time_register - TimeCountOld;
            Flux=Flux+deltaTime;
          }
          TimeCountOld = time_register;
          ContEv++;
        }
        break;
        case "!=":
        if((n_det!=minnplans)&&(n_points<=maxn_points)){
          //preenche os histogramas com os angulos de thi e theta
   convertidos de radianos em graus
          hphi_rec -> Fill((Track1.phi*180/pi));
          htheta_rec -> Fill((Track1.theta*180/pi));
          if (i!=0){
            if(TimeCountOld>=time_register) deltaTime = 65535-TimeCountOld +
    time_register;
            else  deltaTime = time_register - TimeCountOld;
            Flux=Flux+deltaTime;
          }
          TimeCountOld = time_register;
          ContEv++;
        }
        break;
      }
    }
    if ((Flux*TimeStepCount)>0){
      cout << "Freq. = " << (ContEv-1)/(Flux*TimeStepCount) << "Hz" << endl;
    }
    else cout << "***[Warming] Flux can't be calculated" << endl;
    Measurement_Results(ModelProd);
  }
}
void Measurement_Results(Int_t ModelProd){

  TCanvas *c2 = new TCanvas("c2","phi and theta",200,10,700,500);
  c2->SetFillColor(0);
  c2->GetFrame()->SetFillColor(6);
  c2->GetFrame()->SetBorderSize(10);
  c2->GetFrame()->SetBorderMode(3);
```

```
  c2->Divide(1,2);
  hdiftheta->SetLineWidth(3);
  hdifphi->SetLineWidth(3);
  c2->cd(1);
  hphi_rec->SetMinimum(0);
  hphi_rec->Draw("HIST");
  c2->cd(2);
  htheta_rec->SetMinimum(0);
  htheta_rec->Draw("HIST");
}
Int_t ReadSetupInfo(char* namedata ){
  ifstream in;
  in.open(namedata);
  Int_t line=0;
  Int_t x=0;
  if(!in.is_open( ))  // * verifica se o programa conseguiu abrir o arquivo
   de nome informado pelo usuario *
  {
    cout<<"\n***[ERROR] file \"" << namedata << "\" not exist!" << endl;
    in.clear( ); //reseta o objeto leitura
    x=1;
  }
  if(x==0){
    int AuxTimeStepCount=0;
    char CharIn[400];
    char c;
    while (line<33) {
      in.get(c);
      if(c!='\n') (void)sprintf(CharIn,"%s%c",CharIn,c);
      if (c=='\n') {
        if(line==1){
          sscanf(CharIn,"%d %d %d %d %d %d %d %d",
          &Coin_Map_pos_xz[0][0],&Coin_Map_pos_xz[0][1],&Coin_Map_pos_xz
    [0][2],&Coin_Map_pos_xz[0][3],
          &Coin_Map_pos_xz[0][4],&Coin_Map_pos_xz[0][5],&Coin_Map_pos_xz
    [0][6],&Coin_Map_pos_xz[0][7]);
        }
        if(line==2){
          sscanf(CharIn,"%d %d %d %d %d %d %d %d",
          &Coin_Map_pos_yz[0][0],&Coin_Map_pos_yz[0][1],&Coin_Map_pos_yz
    [0][2],&Coin_Map_pos_yz[0][3],
```

```
        &Coin_Map_pos_yz[0][4],&Coin_Map_pos_yz[0][5],&Coin_Map_pos_yz
[0][6],&Coin_Map_pos_yz[0][7]);
      }
      if(line==3){
        sscanf(CharIn,"%d %d %d %d %d %d %d %d",
        &Coin_Map_pos_xz[1][0],&Coin_Map_pos_xz[1][1],&Coin_Map_pos_xz
[1][2],&Coin_Map_pos_xz[1][3],
        &Coin_Map_pos_xz[1][4],&Coin_Map_pos_xz[1][5],&Coin_Map_pos_xz
[1][6],&Coin_Map_pos_xz[1][7]);
      }
      if(line==4){
        sscanf(CharIn,"%d %d %d %d %d %d %d %d",
        &Coin_Map_pos_yz[1][0],&Coin_Map_pos_yz[1][1],&Coin_Map_pos_yz
[1][2],&Coin_Map_pos_yz[1][3],
        &Coin_Map_pos_yz[1][4],&Coin_Map_pos_yz[1][5],&Coin_Map_pos_yz
[1][6],&Coin_Map_pos_yz[1][7]);
      }
      if(line==5){
        sscanf(CharIn,"%d %d %d %d %d %d %d %d",
        &Coin_Map_pos_xz[2][0],&Coin_Map_pos_xz[2][1],&Coin_Map_pos_xz
[2][2],&Coin_Map_pos_xz[2][3],
        &Coin_Map_pos_xz[2][4],&Coin_Map_pos_xz[2][5],&Coin_Map_pos_xz
[2][6],&Coin_Map_pos_xz[2][7]);
      }
      if(line==6){
        sscanf(CharIn,"%d %d %d %d %d %d %d %d",
        &Coin_Map_pos_yz[2][0],&Coin_Map_pos_yz[2][1],&Coin_Map_pos_yz
[2][2],&Coin_Map_pos_yz[2][3],
        &Coin_Map_pos_yz[2][4],&Coin_Map_pos_yz[2][5],&Coin_Map_pos_yz
[2][6],&Coin_Map_pos_yz[2][7]);
      }
      if(line==7){
        sscanf(CharIn,"%d %d %d %d %d %d %d %d",
        &Coin_Map_pos_xz[3][0],&Coin_Map_pos_xz[3][1],&Coin_Map_pos_xz
[3][2],&Coin_Map_pos_xz[3][3],
        &Coin_Map_pos_xz[3][4],&Coin_Map_pos_xz[3][5],&Coin_Map_pos_xz
[3][6],&Coin_Map_pos_xz[3][7]);
      }
      if(line==8){
        sscanf(CharIn,"%d %d %d %d %d %d %d %d",
        &Coin_Map_pos_yz[3][0],&Coin_Map_pos_yz[3][1],&Coin_Map_pos_yz
```

```
[3][2],&Coin_Map_pos_yz[3][3],
        &Coin_Map_pos_yz[3][4],&Coin_Map_pos_yz[3][5],&Coin_Map_pos_yz
[3][6],&Coin_Map_pos_yz[3][7]);
      }
      if(line==9){
        sscanf(CharIn,"%*s %lf",&tile_width_ch);
      }
      if(line==11){
        sscanf(CharIn,"%*s %lf",&tile_height_ch);
      }
      if(line==12){
        sscanf(CharIn,"%*s %lf %lf %lf %lf",&Map_height_z[0],&Map_height_z
[1],&Map_height_z[2],&Map_height_z[3]);
      }
      if(line==15){
        sscanf(CharIn,"%*s %d",&ModelProd);
      }
      if(line==21){
        sscanf(CharIn,"%*s %s",&Reg_word_0);
      }
      if(line==22){
        sscanf(CharIn,"%*s %s",&Reg_word_1);
      }
      if(line==23){
        sscanf(CharIn,"%*s %s",&Reg_word_2);
      }
      if(line==24){
        sscanf(CharIn,"%*s %s",&Reg_word_3);
      }
      if(line==25){
        sscanf(CharIn,"%*s %s",&Reg_word_4);
      }
      if(line==26){
        sscanf(CharIn,"%*s %s",&Reg_word_5);
      }
      if(line==27){
        sscanf(CharIn,"%*s %s",&Reg_word_6);
      }
      if(line==28){
        sscanf(CharIn,"%*s %s",&Reg_N_gate);
      }
```

```
        if(line==29){
          sscanf(CharIn,"%*s %s",&Reg_Ctest);
        }
        if(line==30){
          sscanf(CharIn,"%*s %s",&Reg_DAC0);
        }
        if(line==31){
          sscanf(CharIn,"%*s %s",&Reg_DAC1);
        }
        if(line==32){
          sscanf(CharIn,"%*s %s",&Reg_OTHERS);
          sscanf(CharIn,"%*s %x",&AuxTimeStepCount);
          TimeStepCount=pow(2,(AuxTimeStepCount >> 3))/1000.;
        }
        (void)sprintf(CharIn,"");
        line++;
      }
    }
    cout << "Tile_width= \t" << tile_width_ch << endl;
    cout << "Height(z)= \t[" << Map_height_z[0] << " " << Map_height_z[1] <<
    " " << Map_height_z[2] << " " << Map_height_z[3]<< "]" << endl;
    cout << "MAP:" << endl;
    for(Int_t a=0;a<4;a++){
      for(Int_t b=0;b<8;b++){
        cout << Coin_Map_pos_xz[a][b] << "\t";
      }
      cout << "\n";
      for(Int_t b=0;b<8;b++){
        cout << Coin_Map_pos_yz[a][b] << "\t";
      }
      cout << "\n";
    }

    in.close();
  }
  return x;
}
```