**Dissertação de Mestrado Profissional em Física com Ênfase em Instrumentação Científica**

# Contribuições para o Desenvolvimento de uma Plataforma para Experimentos de Monitoramento Remoto

**Bruno Cerqueira Rente Ribeiro**

**Rio de Janeiro**

**2013**

**MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO**

**Centro Brasileiro de Pesquisas Físicas**

**Coordenação de Física Aplicada**

**CBPF/RJ**

**Bruno Cerqueira Rente Ribeiro**

# Contribuições para o Desenvolvimento de uma Plataforma para Experimentos de Monitoramento Remoto

Dissertação submetida ao Mestrado Profissional em Física com Ênfase em Instrumentação Científica do Centro Brasileiro de Pesquisas Físicas para a obtenção do grau de Mestre
**Orientador: Geraldo Roberto Carvalho Cernicchiaro**

**Rio de Janeiro**
**2013**

# Agradecimentos

Primeiramente, e por motivos óbvios, agradeço aos meus pais, por terem me guiado até a maturidade, para que eu pudesse então escolher o caminho do qual considero certo trilhar.

À minha companheira e em breve doutora, Juliana, por ajudas (científicas e emocionais) infindáveis e 24 horas por dia, nos momentos mais difíceis, além de ser um exemplo de pessoa e de cientista para mim, por sua ética e competência.

Ao meu orientador desde o início de minha graduação, Geraldo Cernicchiaro, por todas as oportunidades dadas e orientações providenciais (incluindo aquelas escritas em guardanapos, que julgo igualmente importantes).

Ao Alexandre Benevento pelo crédito dado a mim ao colocar sob minha responsabilidade a execução do projeto Caipora, do qual acredito vir a ser uma excelente ferramenta contra alguns dos problemas ambientais que enfrentamos atualmente.

Agradeço a todos os integrantes do Laboratório de Instrumentação Científica (LIM) do CBPF, especialmente aos que mais convivi durante este trabalho. Pessoas que contribuíram demais para o desenvolvimento do projeto ou para o desenvolvimento humano no LIM, como o Pedro, a Johanna, o Leonardo, o Marcos, o Rubem, o João, o Gabriel e a Vivian.

A toda a equipe de Agrobiologia da Embrapa, em especial ao Bruno Dias, além da Natalia Bizaia, da Abengoa Bioenergia, pela oportunidade de aplicação do dispositivo em campo.

Agradeço a todos do Laboratório de Recobrimentos Protetores, da metalurgia da UFRJ, que me apoiaram quando precisei me concentrar no mestrado, muitas vezes em detrimento de algumas coisas no dia a dia.

Por fim, agradeço a duas pessoas que, infelizmente, deixaram saudades a todos nós e que também contribuíram para chegarmos até aqui: o professor Ademarlaudo Barbosa e a Rafaela Menezes.

**Resumo**

Neste trabalho é apresentado o desenvolvimento de hardware e software para contribuição na implementação de uma arquitetura voltada para aplicações em sistemas de monitoramento ambiental remoto. Dispositivos que possibilitem o monitoramento ambiental de parâmetros físico-químicos possuem aplicações nos mais variados experimentos científicos ou processos industriais. Os parâmetros físicos ambientais, tais como temperatura, pressão, umidade, condutividade, pH, entre outros, afetam não apenas os experimentos e processos físicos, mas também os equipamentos e sistemas de controle e medida.

A resposta de dispositivos sensores e detectores, bem como a eletrônica de condicionamento destes sistemas, pode ser significativamente afetada, ou comprometida, pelas variações de parâmetros do ambiente experimental. Também podemos constatar o crescente interesse em se monitorar, e quantificar tais parâmetros para compreensão, fiscalização e segurança do próprio meio ambiente, bem como de mecanismos climáticos e ecológicos, no ar, no solo ou na água.

A arquitetura é baseada em microcontroladores e estabelece diversas formas de comunicação com o programa supervisório: WIFI, Ethernet, serial RS-232 e rádio. Essa arquitetura visa a interconexão entre o sistema de aquisição de dados e sondas multiparamêtricas inteligentes de diferentes protocolos de comunicação, facilitando sua implementação e integração com outros sistemas e experimentos.

**Abstract**

This work presents the development of hardware and software to contribute to implement a remote environmental monitoring system. Devices that allow the environmental monitoring of physicochemical parameters are applicable in several scientific experiments and industrial processes. The environmental physical parameters, such as temperature, pressure, humidity, conductivity, pH, among others, affect not only the experiments and physical processes, but also the apparatus and system of control and measurement.

The response of sensing and detection devices, as well as the conditioning electronics of this systems, can be significantly affected or compromised by variations of parameters of the experimental environment. Also we note the increasing interest in monitoring and quantifying these parameters for understanding, monitoring and for the security of the environment as well as climatic and ecological mechanisms, in air, soil or water.

The system uses an architecture based on microcontrollers and provides various forms of communication with the program supervisory: WIFI, Ethernet, RS-232 and radio. This architecture aims the interconnection between the system data acquisition and intelligent multiparametric probes with different communication protocols, facilitating its implementation and integration with other systems and experiments.

**ÍNDICE**

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE SIGLAS

ADC- Analog to Digital Converter

ASCII- American Standard Code for Information Interchange

BIOS- Basic Input/Output System

DO- Dissolved Oxigen

GEE- Gases de Efeito Estufa

GPIO- General Purpose Input/Output

GPS- Global Positioning System

GPRS- General Package Radio Service

I²C- Inter Integrated Circuit

I/O- *Input/Output*

ISO- International Organization for Standardization

MAC- Media Access Control

MISO- Master In Slave Out (relativo ao SPI)

MOSI- Master Out Slave In (relativo ao SPI)

ORP- Oxidation Reduction Potential

OSI- Open System Interconection

PCB- Plated Circuit Board

RS232- Recomended Standard 232

SDA- Serial Data (relativo ao I²C)

SCL- Serial Clock (relativo ao I²C)

SCLK- Serial Clock (relativo ao SPI)

SE- Standard Error

SPI- Serial Peripheral Interface

SS- Slave Select (relativo ao SPI)

TWI- Two Wire Interface

UART- Universal Asynchronous Receiver/Transmitter

## 1. INTRODUÇÃO

Dispositivos que objetivem a medida de parâmetros físicos e químicos são indispensáveis para o desenvolvimento de pesquisas e experimentos nas mais diversas áreas do conhecimento. Além disso, a quantificação de grandezas variadas se faz necessária para ações de monitoramento e fiscalização, incluindo o controle de ações humanas possivelmente prejudiciais ao meio ambiente.

Dessa maneira, a existência de uma grande demanda nesta área de monitoramento gera um mercado de sensores muito amplo e diversificado. Porém a falta de versatilidade e incompatibilidade entre os sensores comerciais acaba afastando possíveis usuários dos sistemas de instrumentação. Essa dificuldade traz aos usuários perdas de produtividade ou objetividade, tanto na área científica quanto na área industrial, ou ainda, diretamente para a sociedade, como em aplicações voltadas para fins de fiscalização.

Atualmente, encontram-se disponíveis sensores de diversos tipos e para diversas aplicações, desde os mais simples e tradicionais, até os mais complexos e específicos. Como exemplo temos medidas de temperatura, que podem ser feitas com os tradicionais termopares e resistores de platina, até modernos sensores de fibras ópticas. Cada aplicação necessita de atenção especial na escolha dos sensores, levando-se em conta parâmetros como resolução, faixa de operação, custo, entre outros fundamentais para o bom funcionamento do experimento ou processo.

Dependendo da aplicação, será então necessária a escolha de algumas características desses sensores, podendo levar à preferência por um ou outro fabricante. Neste ponto são detectadas algumas dificuldades para o usuário deste tipo de equipamento.

A primeira dificuldade é a falta de padronização entre os fabricantes. Apesar de toda pressão por parte dos consumidores em se obter sensores de tecnologia mais

aberta, muitas dificuldades são encontradas ao se propor uma integração entre estes os diversos sensores. Cada aplicação é única, sendo necessária, na maioria das vezes, uma gama de dispositivos de fabricantes diferentes. Porém, não há no mercado um equipamento que faça a leitura desses diversos tipos de sensores simultaneamente.

Outra dificuldade que faz essa demanda de monitoramento não ser atendida corretamente é o fato de não se conseguir equipamentos versáteis, que consigam atender as configurações específicas de cada desenho experimental. Nesses casos, os usuários muitas vezes abrem mão do monitoramento em tempo real ou em campo para a utilização de uma metodologia baseada na coleta de amostras para posterior análise laboratorial.

Esse sistema de coleta para posterior análise possui diversas desvantagens em relação ao monitoramento em campo e em tempo real. Em primeiro lugar, o tempo entre coletas é limitado, o que dificulta a identificação de fenômenos que ocorrem em intervalos de tempo inferiores aos de amostragem. Em segundo lugar, as condições encontradas no ambiente de análise, nem sempre simulam de maneira satisfatória às que seriam encontradas em campo.

## 1.1. OBJETIVOS

Esse trabalho tem como objetivo geral o desenvolvimento de hardware e software que contribui para um sistema voltado para aplicações em monitoramento ambiental remoto.

O sistema do qual este trabalho é integrante deve ser capaz de minimizar as incompatibilidades provenientes da falta de padronização nos protocolos utilizados em sensores digitais. Isso irá permitir a utilização de diversos sensores necessários a uma determinada aplicação. Deve ainda ter usabilidade para que seja aplicável em medidas em campo, onde as condições de medida podem ser adversas.

## 1.2. INSTRUMENTOS DE MEDIDAS

Para a compreensão de um processo de medida na área de instrumentação, alguns conceitos e elementos básicos devem ser definidos. Após essa breve apresentação, será esboçado o esquema de um sistema padrão básico de instrumentação.

O primeiro elemento a ser citado é o sensor. Este elemento gera uma variação em sua saída que está relacionada com o mesurando, ou seja, a entrada do sistema [1]. Para que esta saída seja utilizada de forma conveniente, é necessário um segundo elemento, o conversor, que irá transformar esta saída em uma grandeza que seja facilmente utilizada como entrada para os próximos elementos. Quando este elemento é combinado ao primeiro (sensor), passa a ser chamado de transdutor [1].

Na grande maioria das vezes é necessário, ainda, um elemento de processamento de sinais, para que a entrada seja amplificada, filtrada, entre outros tipos de processamento. Este elemento é capaz de melhorar algumas características do sistema que serão explicadas posteriormente, como a resolução e a acurácia. Quando combinado com o sensor e o conversor, é chamado de transmissor [1].

Há ainda a necessidade de elementos para o envio do sinal para uma estação remota e/ou para sua visualização e gravação. Esses elementos são normalmente separados dos demais, ou por conveniência, ou por dificuldades na acessibilidade ao sistema.

A Figura 1 mostra um diagrama ilustrando o sistema de instrumentação proposto acima.

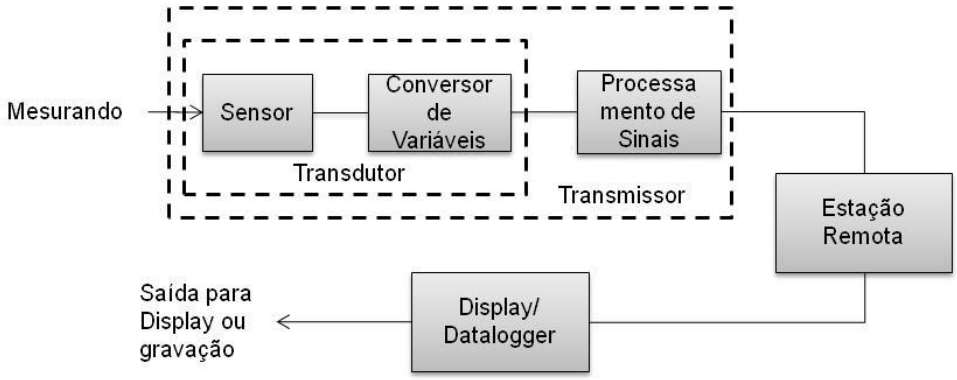



Figura 1: Diagrama de blocos de um sistema de medidas (adaptado de “Principles of Measurement”). Neste diagrama é ilustrado o caminho percorrido pelo sinal desde a medição da variável pelo sensor até a transmissão, gravação ou visualização da saída para possíveis ações posteriores.

Tendo estabelecido esses conceitos, já podemos determinar algumas características de um sistema de medidas padrão. Sobre todo sistema de medida, é possível quantificar características como resolução, acurácia, precisão, etc. Sobre estes parâmetros, será discutido apenas o conteúdo essencial para a correta utilização e calibração do sistema.

Um primeiro parâmetro importante é a excursão. É o primeiro a ser citado, pois é o primeiro a ser pensado quando se deseja projetar um sistema de medidas. Um exemplo simples para explicar a importância do prévio conhecimento desse parâmetro é o de um sistema de medida de temperatura de uma sala. Sabe-se que é improvável que na cidade do Rio de Janeiro a temperatura fique abaixo de $0^{o}C$ ou acima de $100^{o}C$. Dessa forma, no caso em que se deseje medir a temperatura para que seja, por exemplo, controlada por um ar-condicionado, não seria otimizado utilizar sensores que saiam dessa faixa. Chamamos essa faixa (do valor mínimo ao valor máximo) de excursão [2].

A acurácia e a precisão são características que muitas vezes são confundidas, porém, possuem significados distintos e ambas devem ser levadas em conta no desenvolvimento e aplicação de um sistema de medida. Enquanto a acurácia mostra o quão perto a leitura está do valor correto, a precisão está relacionada à

repetibilidade (variação da saída dado o mesmo valor de entrada e condições metodológicas) e à reprodutibilidade (variação da saída dado o mesmo valor de entrada, porém em diferentes condições metodológicas) [2].

Outro parâmetro importante é a resolução. Em todo instrumento de medida, existe uma mudança mínima no mesurando necessária para causar algum efeito visível na saída do sistema, sendo este valor mínimo chamado de resolução [2].

Dentre as características dos instrumentos de medida, talvez uma das mais importantes seja a sensibilidade à perturbação. Todos os parâmetros abordados anteriormente têm valores definidos que somente serão válidos se respeitadas algumas condições ambientais, como temperatura e pressão. Essas condições podem afetar a medida de diferentes maneiras, definindo sua sensibilidade à perturbação [2].

## 1.3. SISTEMA EMBARCADO

Sistemas embarcados são sistemas que possuem software embarcado e hardware necessários para que sejam aplicados em uma função específica [3]. Na literatura é comum se encontrar que um sistema embarcado é aquele que possui um sistema computacional, mas este fica escondido em seu interior.

Os sistemas embarcados possuem então inteligência necessária para a construção de equipamentos e dispositivos, como o exemplo dado na Figura 1. Desse modo, o embarque de dispositivos inteligentes conferem ao sistema maiores funcionalidades, transformando, por exemplo, um sensor em um transmissor.

Estas definições fazem dos microcontroladores dispositivos chave na construção de um sistema embarcado, pois supre a necessidade de inteligência, necessária a um sistema desse tipo.

## 1.4. MICROCONTROLADORES

Como abordado na seção anterior é necessária a adição de alguma inteligência aos dispositivos para que estes possam ser classificados como sistemas embarcados. Esse trabalho é feito por circuitos eletrônicos, especialmente os processadores, que juntamente com alguns circuitos associados, são capazes de executar tarefas previamente programadas, como em um microcomputador [4]. Nesse contexto, os processadores se dividem em relação ao tipo de aplicação: aqueles que são aplicados em sistemas computacionais e os que são aplicados em sistemas de controle [5]. No caso deste trabalho, como se visa um sistema embarcado, será abordado apenas o caso dos microprocessadores utilizados em sistemas de controle, especificamente os microcontroladores.

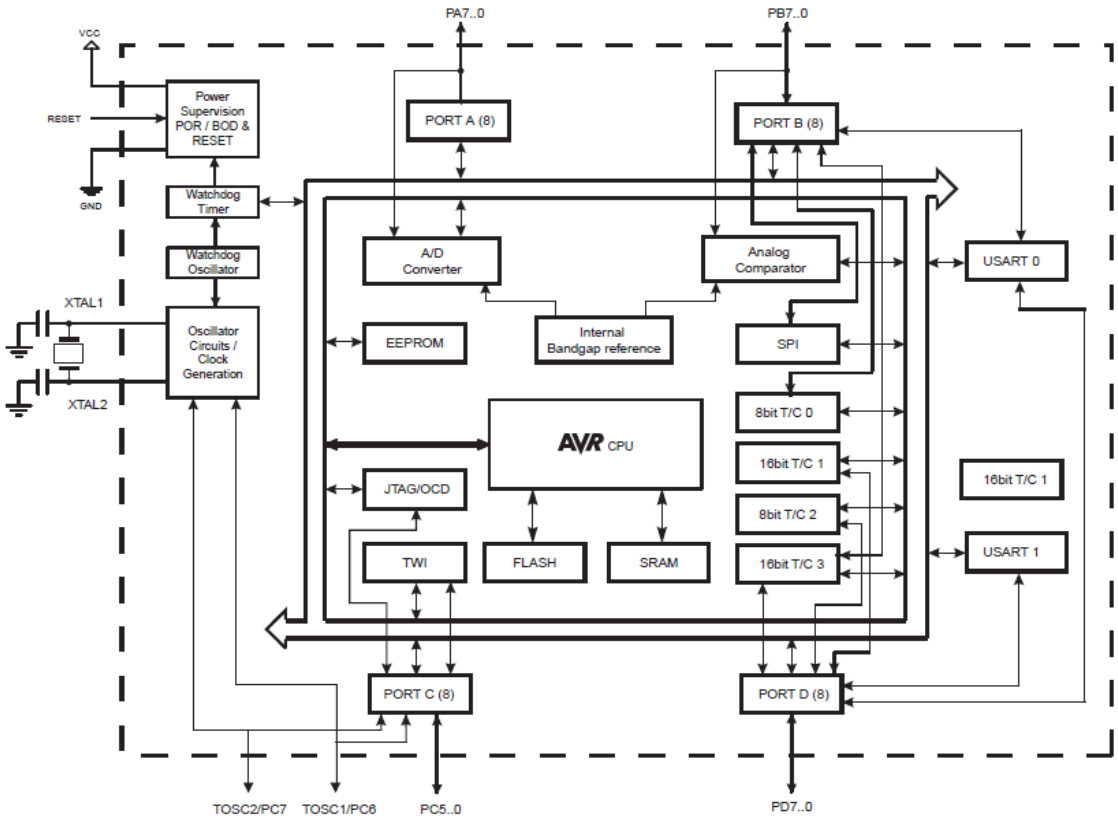



Figura 2: Exemplo de diagrama esquemático de um sistema baseado em microprocessador (retirado de Atmel 8 bit Microcontroler Datasheet). O microcontrolador integra circuitos de memória, portas de entrada e saída genéricas (PORTs), interfaces seriais (USART, SPI, TWI), contadores, entre outros tipos de periféricos.

Os microcontroladores são computadores implementados em um único circuito integrado [6]. Para isso, necessitam funções adicionais a um microprocessador, como pinos de interrupção, gerador de *clock* interno, pinos de I/O; assim como módulos de comunicação (e.g. UART), módulos contadores etc [5], como mostrado na Figura 2. Em contrapartida, por serem utilizados em aplicações dedicadas, costumam ter menos memória que os sistemas orientados à computação [5].

A característica dos microcontroladores de englobarem as várias funções descritas acima lhes confere a possibilidade de facilmente conectarem-se com outros microcontroladores, caracterizando um sistema multiprocessado. Esses sistemas possuem diversas vantagens em relação ao sistema com somente um processador, permitindo, por exemplo, a modularidade do sistema (que possibilita que cada módulo opcional seja adicionado de acordo com a aplicação), a possibilidade de escolha dos dispositivos mais adequados a cada tarefa, a simplificação do código, entre outras vantagens Porém, ao se utilizar essa arquitetura, é necessário que se lance mão de métodos para efetuar a comunicação entre eles, como será descrito na seção seguinte.

## 1.5. O MODELO OSI

O modelo OSI (Open System Interconnection, do inglês, Interconexão entre Sistemas Abertos) é um modelo de referência a ser seguido por sistemas integrantes de uma rede a fim de evitar problemas de padronização em sua comunicação [7]. Sua arquitetura poderia ser descrita em um documento a parte devido ao seu detalhamento, porém, neste documento, serão abordadas somente as características necessárias para o entendimento das questões citadas nos objetivos deste trabalho.

O modelo foi proposto pela Organização Internacional de Padrões (do inglês, ISO) e tem como objetivo, propor, em escala crescente de abstração, a integração dos

sistemas, como por exemplo uma rede de computadores. Com essa ideia em mente, o modelo foi criado contando com uma arquitetura composta por alguns protocolos padrões, definindo os elementos essenciais ao seu funcionamento.

O modelo OSI, através do padrão internacional conferido ao mesmo (ISO 7498) [8], propõe uma padronização somente nas regras de interação entre os sistemas, o que permite a possibilidade de conversão dos sistemas vigentes ao modelo OSI, já que somente o comportamento externo do sistema necessita ser compatível com o padrão [9].

O modelo é composto por alguns elementos importantes para o seu funcionamento. Como elemento inicial, podemos citar os próprios sistemas, que são as peças básicas, autônomas, inteligentes e capazes de, como dito anteriormente, ter sua operação externa compatível com o modelo [7].

Outro elemento importante para a implementação do modelo são suas camadas. A inclusão de camadas no padrão tem o objetivo de tornar possível a decomposição de um sistema, de forma lógica, em subsistemas caracterizados pelo seu nível de abstração dentro do modelo. São sete as camadas que constituem o modelo, cada uma delas, agregando funcionalidades à rede criada [9]:

1- Camada de Aplicação: Como o nome sugere, a camada com maior abstração está diretamente ligada ao usuário através de uma aplicação. Todas as outras camadas só existem para que a camada de aplicação funcione corretamente.

2- Camada de Apresentação: É a camada que faz a camada de aplicação ser capaz de diferenciar os tipos de dados a serem conectados pela camada imediatamente abaixo, a de sessão.

3- Camada de Sessão: Relaciona os dados da camada de apresentação, conectando e desconectando sua comunicação. É responsável também pela sincronização, troca dos dados e limitadores da comunicação.

4- Camada de Transporte: Esta camada é responsável pelo transporte dos dados, ou seja, garantir que os dados cheguem ao destino de forma confiável, sem que seja necessário o conhecimento do caminho percorrido por eles.

5- Camada de Rede: É a camada que torna possíveis as características da camada de transporte, dando uma resolução ao problema de roteamento dos dados.

6- Camada de Dados: Responsável pelo estabelecimento da comunicação relacionada na camada de rede.

7- Camada Física: Estabelece as características mecânicas, elétricas e os meios necessários para que seja estabelecida uma ligação física na camada de dados.

Através destas sete camadas, é então possível desenvolver um sistema aberto que seja interconectável com outros sistemas que também sejam compatíveis com o padrão estabelecido no modelo OSI. Neste trabalho, a compatibilização de dispositivos para o modelo OSI será necessária para que seja implementada uma arquitetura capaz de desenvolver os experimentos de monitoramento desejados.

## 1.6. PROTOCOLOS DE COMUNICAÇÃO

Este trabalho tem como foco a forma de processamento e armazenamento dos dados gerados pelos sensores, propondo uma arquitetura para esse propósito. Essa arquitetura utiliza os sinais de saída dos sensores utilizados como entrada para o sistema de aquisição de dados. Sendo assim, os blocos de interface do sistema de medidas exercem grande importância no projeto, sendo responsáveis pela compatibilização dos sensores com o bloco de armazenamento e supervisão.

Um bloco de interface pode estar nas duas pontas do meio de transmissão, ou seja, pode estar diretamente ligado ao transdutor ou diretamente ligado ao sistema de aquisição de dados. Muitos fabricantes já disponibilizam seus produtos integrados a sistemas de condicionamento de sinais, conversores analógico-digitais (ADC), etc.,

provendo um sinal digital já codificado em algum tipo de protocolo padrão de comunicação utilizado na área.

Para o melhor entendimento da utilidade dos protocolos de comunicação, é necessário que se conheça um conceito importante em instrumentação nos dias atuais: o de sistema de controle distribuído. Como dito anteriormente, muitos fabricantes têm utilizado a capacidade de processamento dos microprocessadores modernos para incluir inteligência em seus dispositivos. Desse modo, a complexidade do sistema é dividida entre estes elementos, formando uma arquitetura de sistema de controle distribuído. Isso torna o sistema mais tolerante a erros do que aqueles baseados em um único computador, interligado com os dispositivos através de longos cabos de instrumentação; assim como aumenta seu poder de processamento, à medida que mais processadores são incluídos [10].

Para que os diversos dispositivos se comuniquem, algumas informações de controle necessitam ser enviadas para que se minimizem alguns tipos de erros comuns em comunicações digitais, como por exemplo, dois dispositivos tentarem acessar ao mesmo tempo o mesmo destino [10]. Neste projeto serão abordados alguns tipos de protocolos de comunicação serial, necessários na interface dos dispositivos sensores com o sistema proposto, entre o sistema supervisório e o sistema de aquisição, e até mesmo entre as diversas partes internas do sistema.

As comunicações seriais podem ser caracterizadas de diversas formas, sendo as mais informativas relacionadas ao seu sincronismo e ao direcionamento das linhas de dados. Em relação ao sincronismo, podemos classificá-las como síncronas (as que possuem um sinal de *clock* para que sejam sincronizados os dados do transmissor ao receptor) ou assíncronas (que realizam essa sincronia sem a necessidade de sinais de *clock*) [10]. Já em relação às linhas de dados, existem três tipos: *simplex, half-duplex* e *full-duplex*. Na *simplex*, apenas uma linha de dados é

disponibilizada e a comunicação é feita somente em uma direção. Na *half-duplex*, também só existe uma linha de dados, porém, trafegando sinais de transmissão e recepção. A comunicação *full-duplex*, por outro lado, possui duas linhas de dados, possibilitando o envio e a recepção dos dados simultaneamente [10].

Como dito anteriormente, alguns desses protocolos de comunicação são importantes para o entendimento deste trabalho, e.g., RS-232, I²C, SPI.

O RS-232 é um padrão baseado na comunicação assíncrona, o que lhe confere a possibilidade de utilizar um sinal de *clock* independente de outros dispositivos para transmitir e receber dados [10]. Normalmente são codificados em ASCII e frequentemente conectados a uma interface UART [11].

Em contrapartida à comunicação assíncrona descrita acima, o barramento I²C é comumente utilizado em aplicações de microcontroladores e utiliza uma comunicação síncrona e *half-duplex*. Deste modo, possui dois pinos, um responsável pelo *clock* (SCL, do inglês, *Serial Clock*) e outro responsável pelos dados de entrada e saída (DAS, do inglês, *Serial Data*). Pode endereçar até 128 dispositivos através de uma palavra de 7 *bits* enviada durante a comunicação [6]. Atualmente este protocolo é utilizado por diversos fabricantes de dispositivos eletrônicos, o que gera uma gama de possibilidades de periféricos já implementados comercialmente, como EEPROMs, expansores de *bits*, conversores analógico-digitais etc. Devido a esta facilidade, o protocolo $I^2C$ é utilizado atualmente em milhares de circuitos integrados diferentes, pelos mais diversos fabricantes [12]. A Figura 3 ilustra uma arquitetura utilizando esse tipo de comunicação.

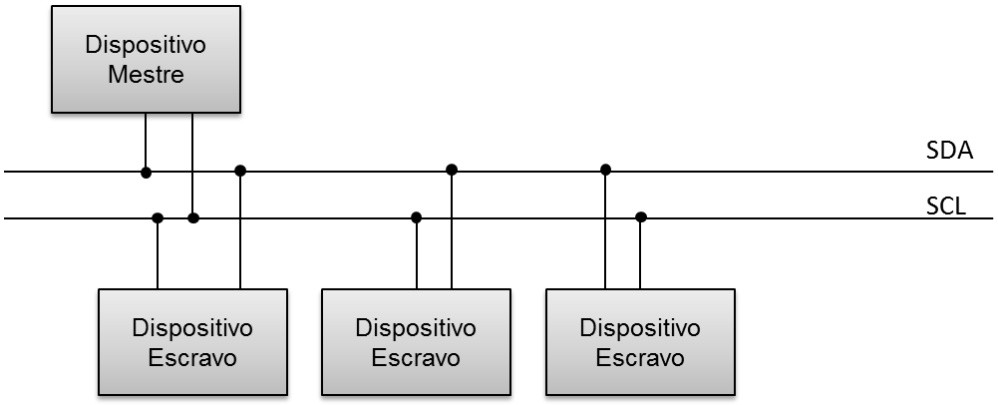



Figura 3: Ilustração de um barramento I²C com dispositivos mestre e escravo. Sua linha de *clock* é controlada pelo mestre para o sincronismo na comunicação.

Por ser um protocolo com principal aplicação sendo a comunicação entre dispositivos do mesmo circuito, não existe um conector especial para este padrão, já que, normalmente, são utilizadas as próprias trilhas do circuito impresso.

Em relação à tensão de operação, o protocolo também é flexível, pois pretende englobar diversos tipos de dispositivos, que operam com tecnologias diferentes. O padrão impõe somente que a tensão do nível baixo (“0” lógico) seja abaixo de 30% da alimentação e o nível alto (“1” lógico) seja acima de 70% do valor da alimentação [12]. Além disso, os sinais SDA e SCL precisam ser conectados à alimentação por um resistor de *pull-up*.

Com a conexão e os níveis de tensão estabelecidos, o próximo passo é a formação do byte. Todo o byte no protocolo I²C é iniciado por uma condição de *Start*, que é uma transição de nível alto para nível baixo enquanto o sinal SCL permanece em nível alto. Do mesmo modo, o byte é terminado por uma condição de *Stop*, que, analogamente à condição de *Start*, é constituído de uma transição de nível baixo para nível alto quando o sinal SCL permanece em nível alto.

Após a condição de *Start*, são enviados os sete bits de endereço do dispositivo escravo, seguido de um bit de direção (nível alto = *read* e nível baixo = *write*). Depois desse primeiro byte, são transmitidos os bytes de dados, constituídos de

8 bits. Cada byte é separado por um bit de *Acknowledge* gerado pelo receptor (dispositivo mestre no caso de *read* e escravo no caso de *write*). A Figura 4 mostra esse procedimento, desde a condição de *Start* até a condição de *Stop*.

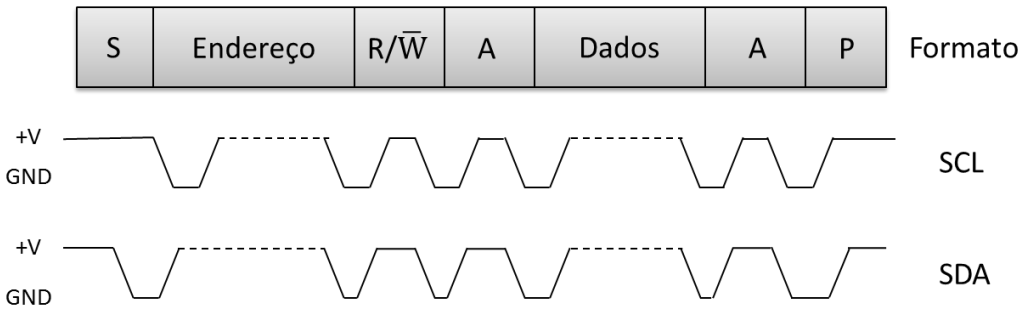



Figura 4: Formato dos bytes do protocolo I²C. A condição de *Start* (S) é seguida pelos 7 bits de endereço do dispositivo escravo e do bit de direção (R/W), aguardando uma resposta de um *Acknowledge* (A) pelo receptor. O dispositivo transmissor (mestre em caso de "W" e escravo em caso de "R") envia um byte de dados e espera a resposta de outro *Acknowledge*. Caso a transmissão seja encerrada, o mestre envia uma condição de *Stop* (P) para finalizar a comunicação.

O barramento SPI, assim como o I²C, também é um protocolo de comunicação serial síncrona, porém *full-duplex*. Para isso, dispõe de uma linha para a entrada e outra para saída de dados, ambas relativas ao dispositivo mestre (respectivamente MISO e MOSI). Possui ainda uma linha de *clock* controlada pelo dispositivo mestre (SCLK) e uma linha de seleção para cada periférico conectado ao barramento (SS). Sua topologia mostrada na Figura 5 possui diferenças em relação ao I²C, em termos de velocidade (o SPI é endereçado por *hardware*, reduzindo o tamanho do cabeçalho e aumentando a velocidade) e versatilidade (o SPI necessita de 3 pinos para a comunicação contra 2 do I²C, e ainda um pino a mais para cada periférico conectado) [6].

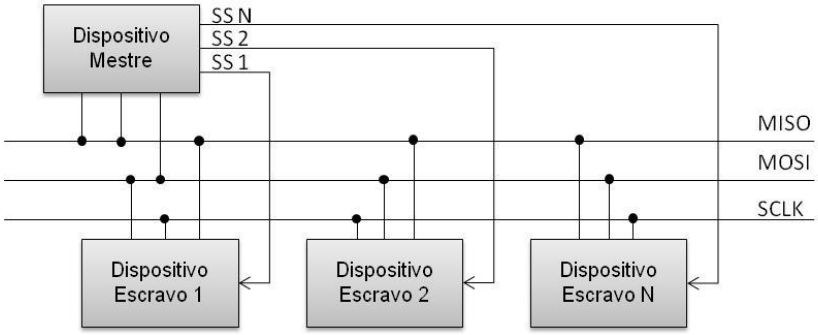



Figura 5: Ilustração de um barramento SPI. Este possui comunicação *full-duplex*, onde a linha MISO corresponde aos dados de entrada no dispositivo mestre e saída no dispositivo escravo; e a linha MOSI corresponde aos dados de saída no dispositivo mestre e entrada no dispositivo escravo. Possui ainda uma linha de *clock* controlada pelo mestre e uma linha de seleção para cada dispositivo periférico escravo ligado ao barramento.

Além dos padrões de comunicação mostrados anteriormente, diversos padrões são criados de acordo com cada área de aplicação, incluindo algumas interfaces proprietárias empregadas por alguns fabricantes de dispositivos eletrônicos [6].

## 1.7. O MONITORAMENTO REMOTO

O monitoramento remoto e em tempo real é uma área de pesquisa multidisciplinar e amplamente aplicada atualmente. Alguns fabricantes de dispositivos sensores já colocaram no mercado produtos para a realização desse tipo de monitoramento ambiental, como é o caso da Hydrolab, que possui alguns produtos para comunicação, além de sondas multiparamétricas com uma estação remota [13].

Outra prática comum nessas aplicações é a contratação de empresas para a realização de soluções integradas, ou seja, o desenvolvimento de dispositivos bem específicos para uma aplicação. Esse tipo de solução já pode ser encontrada através de empresas no Brasil [14], porém, envolve altos custos, já que para cada caso específico é feito um projeto a parte.

Na área científica, um grande esforço é atualmente voltado para o monitoramento ambiental realizado por redes sem fio de sensores. Aplicações diversas são atualmente usuárias desse tipo de tecnologia. De experimentos físicos a

agricultura, utilizando diferentes metodologias de medidas (alguns exemplos podem ser encontrados nas referências bibliográficas deste trabalho).

O sistema do qual este trabalho faz parte visa um diferencial: a flexibilidade dos experimentos de monitoramento ambiental, utilizando configurações via software para a adição de sensores de protocolos diversos. Desse modo, a metodologia de um experimento como este pode facilmente ser mudada, somente trocando-se os sensores desejados.

## 2. DESCRIÇÃO DO SISTEMA

Neste capítulo é descrita a implementação do hardware e do software desenvolvidos: Os circuitos da Placa Mãe e do Módulo de Compatibilização; e a programação embarcada nos microcontroladores e no Sistema Supervisório. Esses componentes irão integrar o sistema cujo objetivo é a realização de experimentos de monitoramento remoto em tempo real. A Figura 6 mostra um diagrama esquemático deste sistema.

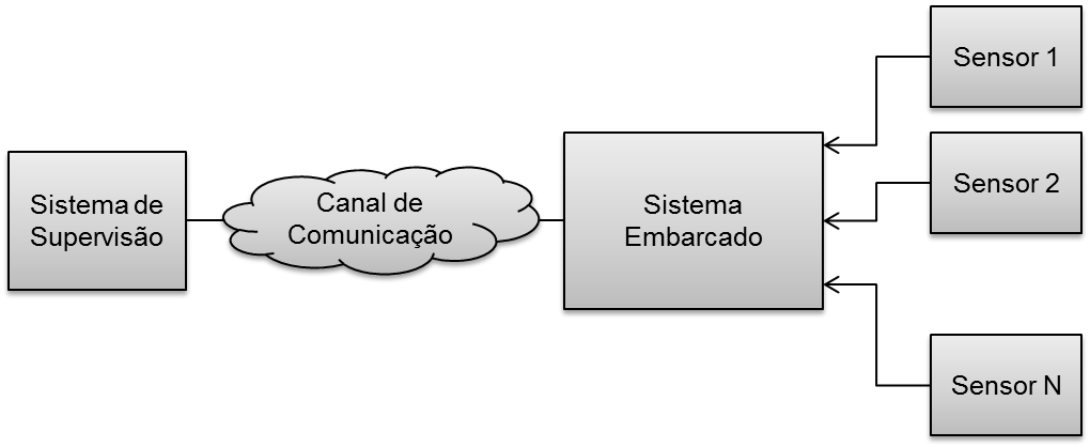



Figura 6: Diagrama do experimento de monitoramento remoto. Neste método, um software de supervisão se comunica com a parte embarcada do sistema, que possui inteligência para desenvolver tarefas básicas.

As ferramentas desenvolvidas para o sistema proposto possuem algumas funções lógicas e processuais básicas para seu funcionamento. Estas são exercidas por cada bloco funcional do hardware, separadamente, como mostra a Figura 7.

Esta arquitetura possui dois microcontroladores centrais que compõem a placa principal do dispositivo (chamada neste documento de Placa Mãe) e que terão suas funções descritas posteriormente: o **Microcontrolador Leitura** e o **Microcontrolador Mestre**. Além desses dois microcontroladores centrais, outros também são utilizados nos **Módulos de Compatibilização** (que não foram finalizados neste trabalho), onde os sinais provenientes dos sensores são compatibilizados para a comunicação do **Barramento de Medidas**.

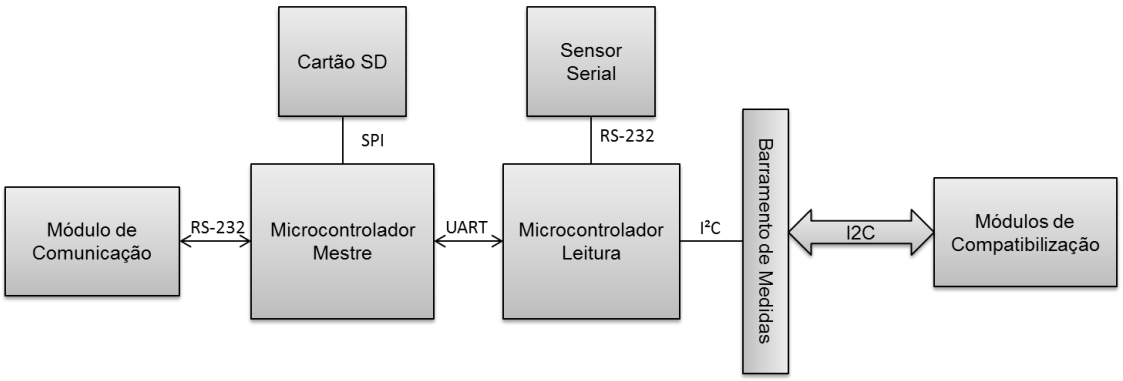



Figura 7: Diagrama de blocos das funções do hardware desenvolvido para a operação remota. Possui um microcontrolador (leitura) para o controle das medidas adquiridas pelo barramento $I^2C$ e porta serial; e um microcontrolador (mestre) para controle do armazenamento no cartão SD e comunicação com o software supervisório (através do módulo de comunicação). Apesar de não finalizados logicamente , essa arquitetura possui também módulos de compatibilização, responsáveis por compatibilizar os sensores para a comunicação $I^2C$ do barramento de medidas.

O sistema possui armazenamento não volátil, na forma de um **Cartão SD**, que confere ao sistema capacidade de armazenamento dos dados gerados e outras características que serão abordadas adiante.

Outro bloco funcional do sistema proposto é um **Módulo de Comunicação**, que irá estabelecer a comunicação entre o hardware embarcado e o Sistema Supervisório.

Desse modo, propõe-se uma arquitetura responsável pela compatibilização, formatação, segurança e comunicação dos dados relativos às medidas realizadas, desenvolvendo-se algumas das ferramentas que serão utilizadas por esse sistema final.

## 2.1. DESCRIÇÃO DOS CIRCUITOS

Nesta sessão serão descritos os componentes e hardware necessários para a realização das funções dos blocos implementados. A Figura 8 mostra um protótipo que possibilita a realização de testes das ferramentas desenvolvidas.

Figura 8: Dispositivo embarcado montado e pronto para a conexão dos sensores. Esta versão possui uma conexão serial para comunicação com o Software Supervisório, outra para a comunicação com o Sensor serial embarcado, além de uma conexão SDI-12 e entrada para cartão SD. Outros dois conectores DB9 foram incluídos ao chassi para possíveis inclusões de módulos.

### 2.1.1. PLACA MÃE

A Placa Mãe é a principal peça de hardware do sistema e possui uma arquitetura planejada para, em sua configuração mínima, ser capaz de efetuar medidas com um sensor acoplado em sua porta serial (RS-232) e comunicar-se com o Sistema Supervisório, também através de uma porta RS-232. Porém, é capaz de integrar funcionalidades de outras placas para compor sistemas mais complexos, como aquele mostrado anteriormente na Figura 7. Esse hardware é descrito em detalhes na Figura 9.

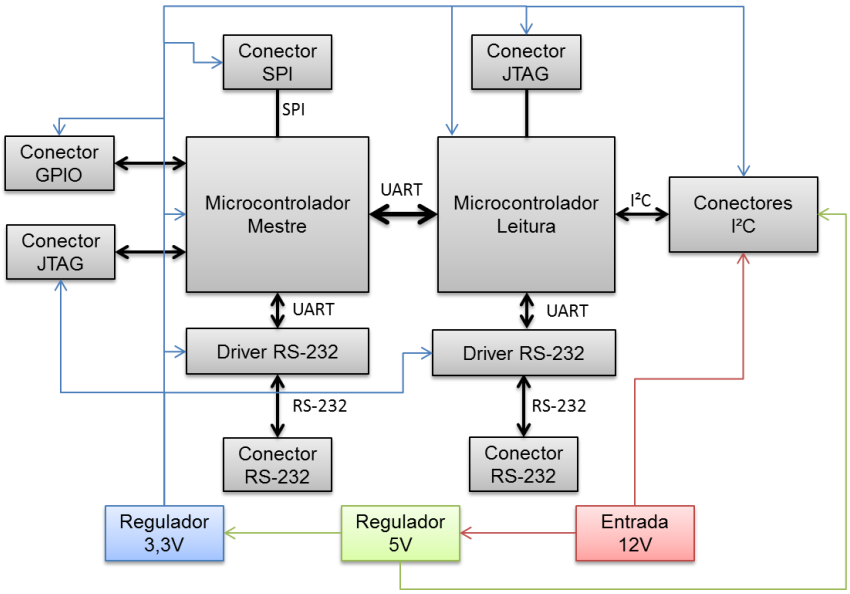



Figura 9: Diagrama com a descrição do hardware que compõe a Placa Mãe. As linhas de alimentação são representadas pelas cores de seus respectivos valores de tensão.

Os dispositivos que adicionam inteligência a esta placa são o Microcontrolador Mestre e o Microcontrolador Leitura. Para a construção do circuito, foi utilizado o microcontrolador ATmega 1284p, em ambos os casos. Essa escolha se deu pelas características de memória RAM (possui 16k Bytes, suficiente para a formação e movimentação de um bloco de medidas e outro de configuração, ambos de 512 bytes); e de periféricos de comunicação (possui implementado o "*two wire interface*" (TWI) compatível com o protocolo $I^2C$, comunicação SPI e, ainda, duas interfaces UART, necessárias para a implementação dessa arquitetura) [15]. A Figura 10 mostra as possíveis configurações dos pinos desse microcontrolador que, devido à restrição de número de pinos, podem ter diferentes funções, dependendo da sua programação. Em verde são mostrados os "ports", ou seja, registradores responsáveis por sinais digitais de entrada/saida (I/O). Em azul claro, identificamos os conversores analógico-digitais (ADC0 a ADC7), em laranja os sinais da comunicação I²C, entre outros. Por exemplo, o pino 40 do microcontrolador pode ser programado para funcionar como uma entrada ou saída digital (Port) ou como um canal do conversor analógico digital (ADC).

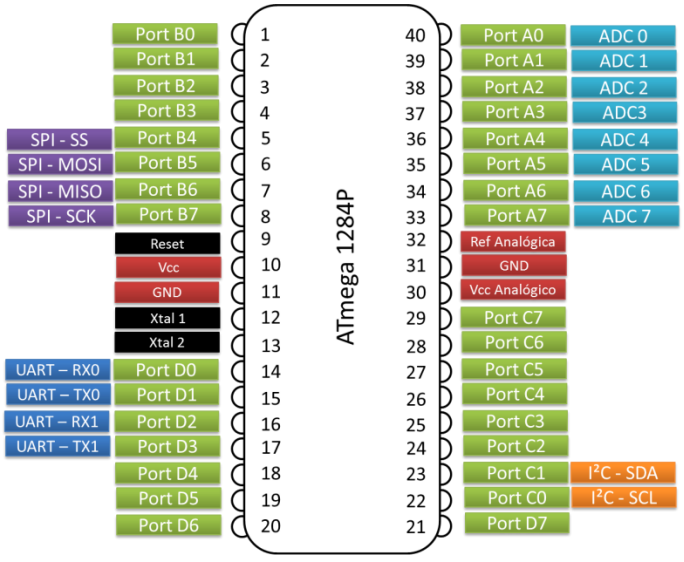



Figura 10: Configuração simplificada de pinos no microcontrolador ATmega 1284P. Além dos pinos de I/O gerais ("Port", ou seja, portas de entrada e saída digital), possui pinos para diversos tipos de comunicação, conversores analógico-digitais (ADC) e entrada de clock para um cristal (Xtal). Outras funções não especificadas também podem ser configuradas nos pinos, como alguns contadores, timers e comparadores (adaptado de Atmel 8 bit Microcontroler Datasheet).

O hardware desenvolvido prevê a alimentação de 3,3V a todos os componentes do sistema. Para isso, são utilizados dois reguladores de tensão: um que disponibiliza 5V através de uma entrada de 12V e outro que disponibiliza 3,3V a partir da linha de 5V já regulada. Apesar de todos os componentes serem alimentados com 3,3V, as linhas de 5V e 12V são disponibilizadas nos conectores I²C (Barramento de Medidas). Essas linhas são disponibilizadas pois são úteis a alguns tipos de módulos conectados ao barramento, como por exemplo o módulo SDI-12, que será explicado adiante.

Foi incluída à Placa Mãe uma interface de programação e testes dentro da própria placa, chamado de JTAG. Apesar de estar incluído no hardware desenvolvido, esta funcionalidade não foi implementada neste trabalho. Os conectores JTAG, assim como todos os conectores que serão descritos, possuem 10 pinos e podem ser conectados através de um flat cable.

Para a conexão do Microcontrolador Mestre com o módulo de armazenamento, onde se localiza a memória SD, foi adicionado um conector com os sinais do protocolo SPI. Esse conector possibilita a ligação do cartão SD ao painel do dispositivo para facilitar o acesso.

Outro conector cuja utilização é opcional é o GPIO (do inglês, General Purpose Input/Output). Este conector é ligado diretamente à pinos homônimos (GPIO) do Microcontrolador e é responsável por eventuais necessidades de sinais adicionais no sistema, como por exemplo, LEDs de indicação de atividade ou botão para a realização de medidas no Modo Local, modo este que será explicitado posteriormente.

Com o mesmo tipo de conector, foram incluídos à Placa Mãe os sinais do protocolo I²C para formar o Barramento de Medidas. Essa inclusão utiliza dois conectores idênticos para que se possa realizar uma ligação em *daisy chain*, ou seja, um módulo conectado ao barramento é capaz de conectar outros módulos em cascata, como mostrado na Figura 11.

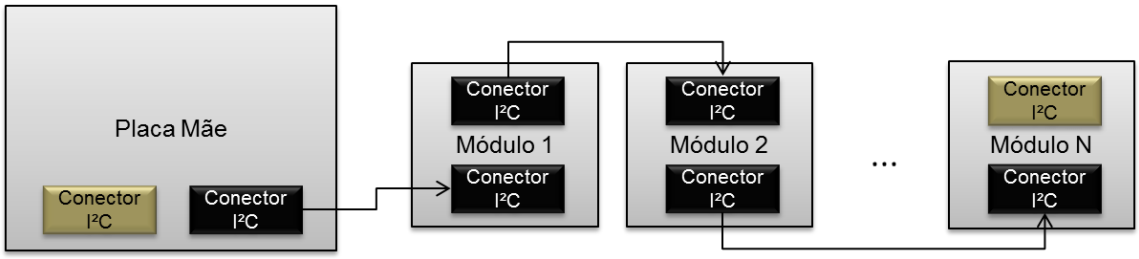



Figura 11: Conexão entre os módulos e a Placa Mãe, formando o Barramento de Medidas. Cada placa possui dois conectores, possibilitando uma ligação em *daisy chain*.

Outro tipo de comunicação prevista na arquitetura apresentada é o RS-232. Existem duas interfaces deste tipo de protocolo: uma conectada ao Microcontrolador Mestre e outra ao Microcontrolador Leitura. A interface relativa ao Microcontrolador Mestre é responsável pela comunicação com o Sistema Supervisório. Já a interface do Microcontrolador Leitura é responsável pela possível conexão de um sensor serial extra, que não passa pelo barramento I²C.

Para a implementação deste protocolo é necessária a conversão dos níveis de tensão para que sejam utilizados os sinais UART dos microcontroladores, já que esses não possuem comunicação que atenda aos padrões do RS-232. Para esta tarefa foi utilizado um *driver* capaz de, através de um controlador de comunicação assíncrona, gerar um sinal compatível com os níveis de tensão do protocolo RS-232 [16].

Após a compatibilização dos níveis da comunicação UART com o protocolo RS-232, esses sinais são disponibilizados em dois tipos de conectores diferentes: em um conector DB9, próprio do protocolo ou em um conector de 10 pinos para a conexão de um *flat cable* (como nos conectores mostrados anteriormente). Essa segunda opção é dada para que seja possível a conexão do DB9 no painel do equipamento, nas situações que não seja possível incluí-lo no PCB.

Com essas características em mente, a Placa Mãe foi desenvolvida em uma prototipadora no Laboratório de Instrumentação e Medidas (LIM) do CBPF, resultando no hardware esquematizado na Figura 12 e mostrado na foto da Figura 13.

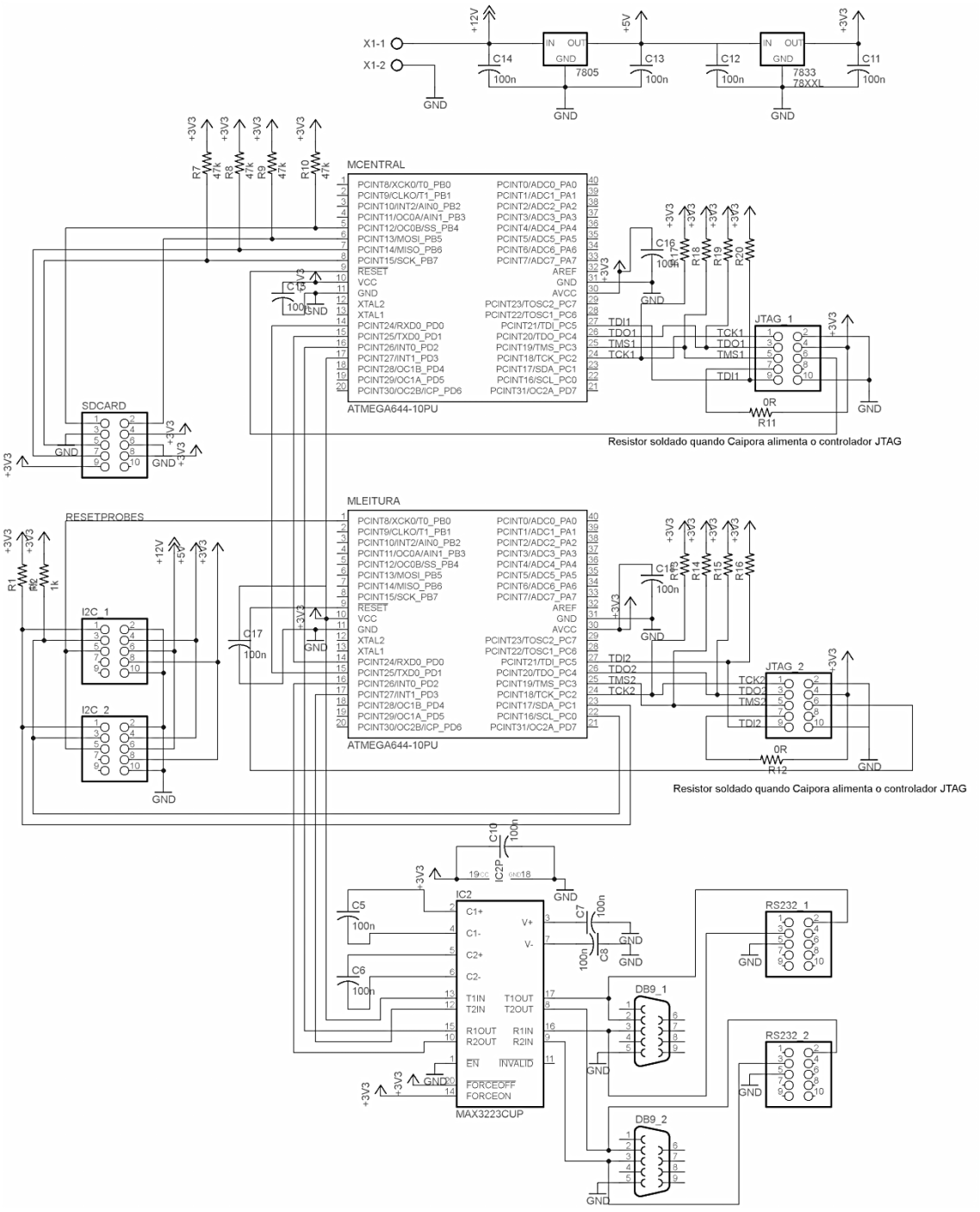



Figura 12: Esquemático da Placa Mãe do sistema. Esse esquema mostra as ligações dos dois microcontroladores (Mestre e Leitura), o *driver* RS-232 e os conectores utilizados.

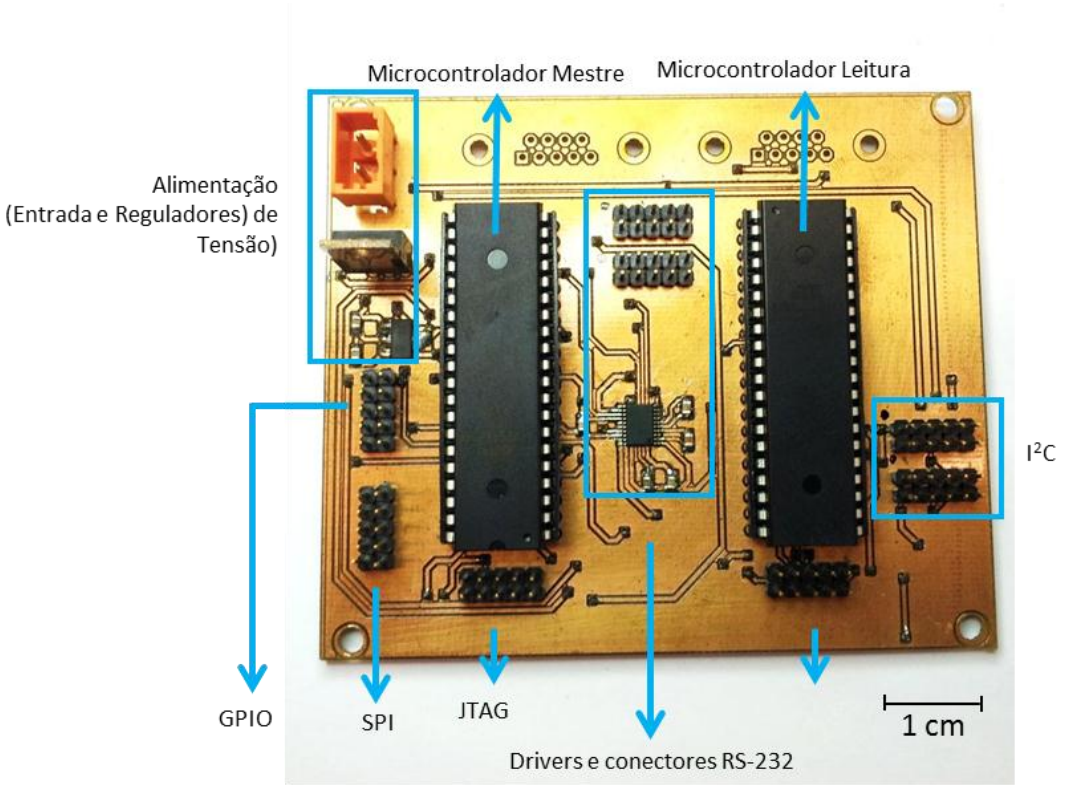



Figura 13: Protótipo montado da Placa Mãe, que inclui o Microcontrolador Mestre, o Microcontrolador Leitura e as interfaces necessárias à arquitetura proposta.

### 2.1.2. MICROCONTROLADOR MESTRE

Assim que o sistema entra em operação, o Microcontrolador Mestre inicializa suas variáveis internas através dos parâmetros de configuração gravados no cartão SD. Essas variáveis internas dizem respeito à operação do sistema e à configuração do experimento (parâmetros que irão tornar o sistema flexível suficiente para que se possa modificar as características das medidas, como por exemplo os comandos e formatos para a comunicação com os sensores, ou os tempos de aquisição envolvidos) e serão detalhadas adiante. Após a inicialização dessas variáveis, os blocos de configuração são enviados ao Microcontrolador Leitura, para que este também inicialize as suas variáveis de configuração do processo de medida.

O Microcontrolador Mestre passa a, então, realizar o *loop* principal. Este consiste em enviar requisição de medida, receber o bloco de medidas, armazená-lo

(caso a função de armazenamento esteja ativa) e enviá-lo ao sistema de supervisão de acordo com as necessidades do experimento.

O Microcontrolador Mestre também é capaz de receber alguns comandos especiais, responsáveis por funções chave dentro do processo: configuração do sistema, requisição do *status* atual das configurações e descarregamento do conteúdo do cartão SD, funções essas implementadas no *firmware* do microcontrolador em questão, como mostra a Figura 14.

```
if (USART1_Chars_In_Buff()>0)
    {

    switch (USART1_Receive()) //check_comando_central();
        {
            case  'C':      //   Comando de envio de configuração pela Central para o MC Mestre
                leitura_configuracao_central();
                USART1_Clr_Rxbuff();
            break;
            case  'c':      //   Comando de questionamento de configuração do MC Mestre
                mostra_config();
            break;

            case 'L':
            case 'l':
                erro_leitura=le_bloco();
                USART0_Clr_Rxbuff();
            break;

            case 'Y':
                descarrega_sd_card_serial();
                USART1_Clr_Rxbuff();
            break;

        }
    }
```

Figura 14: Implementação das funções relativas aos comandos vindos do Sistema Supervisório no *firmware* do microcontrolador.

A configuração do sistema é feita através da gravação dos dados referentes às configurações, que ficam alocadas nos primeiros blocos do cartão SD (as mesmas configurações que inicializaram as variáveis no início do programa), formando uma espécie de cabeçalho. A central (sistema supervisório) inicia uma troca de dados com o Microcontrolador Mestre e este sobrescreve as configurações anteriores no cartão de memória. Esses dados armazenados serão requisitados para a inicialização do sistema.

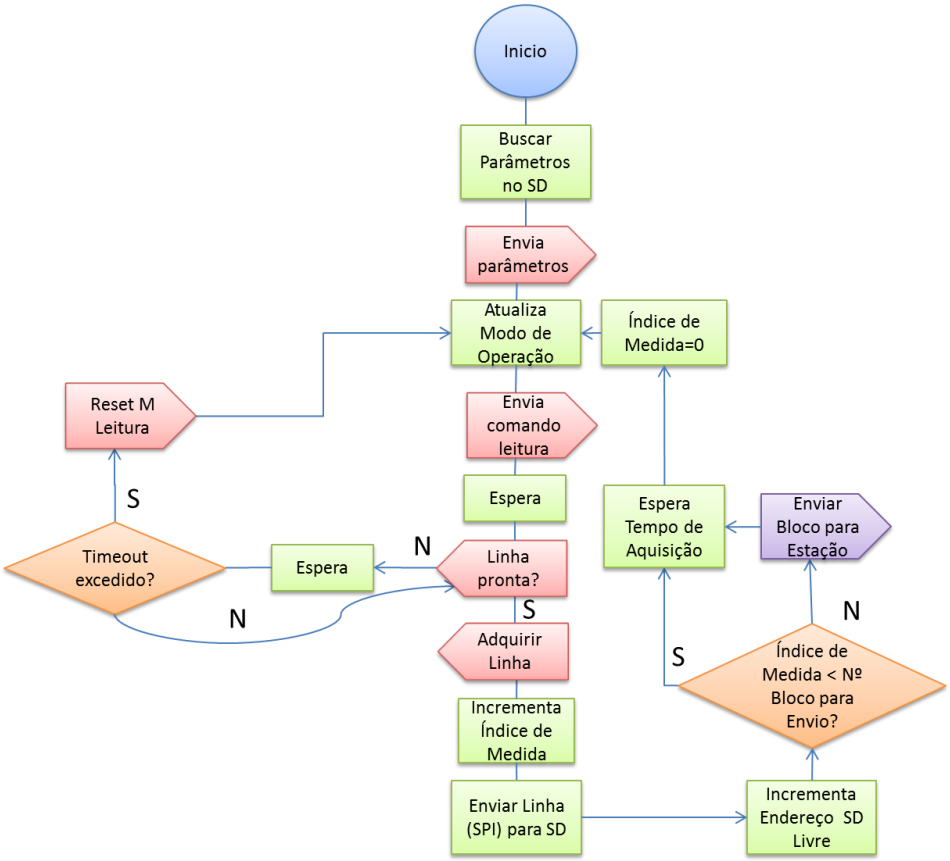



Figura 15: Fluxograma referente ao processo realizado pelo Microcontrolador Mestre. Nesta figura, os processos que apontam para a esquerda ilustram uma ação de recepção, enquanto os que apontam para a direita ilustram uma ação de envio.

A requisição das configurações é um simples pedido enviado pelo Sistema Supervisório para que lhe sejam enviados estes blocos correspondentes às configurações. Já o descarregamento do conteúdo do cartão SD é realizado quando o supervisório necessita dos dados contidos na memória, iniciando então o processo de envio de todos os dados do cartão SD, desde o primeiro endereço (incluindo as configurações atuais) até o endereço da última medida. A Figura 15 ilustra o processo realizado pelo Microcontrolador Mestre e sua interação com o Microcontrolador Leitura e a estação de supervisão.

### 2.1.3. MICROCONTROLADOR LEITURA

Com o objetivo de receber e concentrar os dados provenientes das medidas dos sensores acoplados ao sistema,foi empregado um barramento de comunicação

serial. Esse barramento integra sensores que utilizam protocolos diversos em um meio único, onde se pode obter uma comunicação com o Microcontrolador Leitura, para ser formado o bloco de medidas e posterior, armazenamento e envio.

O barramento de medidas opera através do protocolo de comunicação serial síncrona, o $I^2C$. Como dito anteriormente, este protocolo possui como sua vantagem principal em relação aos protocolos similares, a capacidade de expansão da quantidade de dispositivos conectados ao barramento sem que haja necessidade de mudanças no hardware ou no processo. Esse protocolo confere uma das vantagens do sistema descrito neste projeto: sua versatilidade em, futuramente, adicionar ou remover módulos e sensores ao experimento.

O barramento de medidas está diretamente conectado ao Microcontrolador Leitura. Este microcontrolador é o responsável pela formação do bloco de medidas, ou seja, a *string* que contém as medidas dos sensores e que será armazenada e, posteriormente, processada pelo software de supervisão.

O Microcontrolador Leitura ainda possui uma interface serial assíncrona que é utilizada para a comunicação com um sensor serial. Os dados gerados por este sensor também são adicionados ao bloco de medidas, mencionado anteriormente. A ideia dessa interface é amparar a operação através da conexão de dispositivos que auxiliem a interpretação e a tomada de decisões em relação às medidas, como por exemplo, um GPS. Esses dispositivos são importantes para a maioria dos experimentos em monitoramento remoto, assim como evitam alguns artefatos de medidas.

Para iniciar a operação de adquirir os dados provenientes dos módulos, o Microcontrolador Leitura necessita receber as configurações do Microcontrolador Mestre. Portanto, o Microcontrolador Leitura fica a espera desses parâmetros, para que possa inicializar suas variáveis internas. Apesar de não ter sido implementado

neste trabalho, propõe-se que no sistema final o Microcontrolador Leitura repasse as configurações para os módulos e teste sua atividade.

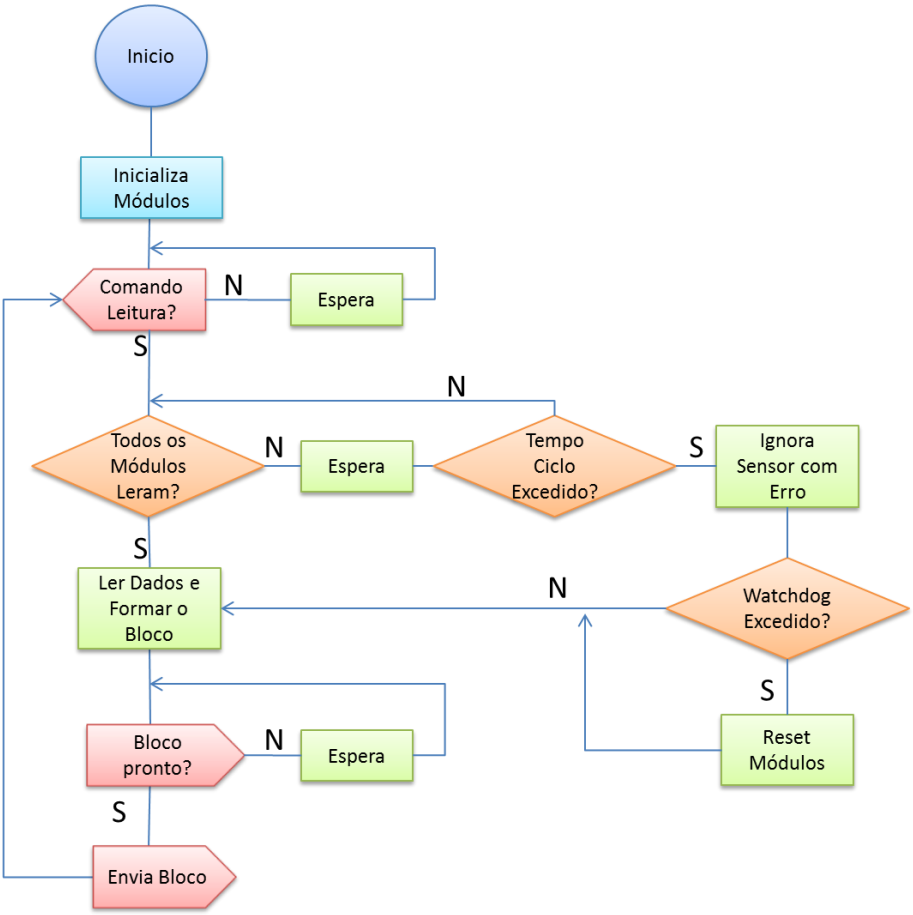



Figura 16: Fluxograma referente ao processo realizado pelo Microcontrolador Leitura. Nesta figura, os processos que apontam para a esquerda ilustram uma ação de recepção, enquanto os que apontam para a direita ilustram uma ação de envio.

Sendo assim, pode ter início o ciclo de operação da parte do sistema responsável pela medição. Este ciclo termina com o envio do bloco de medidas para o Microcontrolador Mestre, que irá controlar as próximas etapas do experimento. O processo realizado pelo Microcontrolador Leitura é ilustrado no fluxograma da Figura 16. Lembrando que o processo de inicialização dos módulos ainda não foi implementado.

### 2.1.4. MÓDULOS DE COMPATIBILIZAÇÃO

Ao se propor um sistema que utilize ao mesmo tempo sensores com diferentes protocolos, comandos e configurações, é necessária alguma adaptação. No caso deste trabalho, os sensores precisam se comunicar com o barramento de medidas.

Para a compatibilização dos sensores, a arquitetura de hardware do sistema prevê a adição de módulos capazes de realizar a tarefa de interconectar o Microcontrolador Leitura com cada sensor, via barramento de medidas. Como ilustrado anteriormente na Figura 7, os módulos de compatibilização serão responsáveis pela adaptação de protocolos. Cada modulo recebe um endereço I²C e cada canal é configurado em função do dispositivo sensor. Normalmente, este dispositivo sensor dispõe de uma tabela de comandos de identificação, configuração, escala, leitura etc. Estes comandos podem ser encontrados no manual dos dispositivos, disponibilizado pelo fabricante, e configurados como uma cadeia de caracteres definidos no painel de configuração, como será explicado adiante e ilustrado na Figura 32.

Os módulos de compatibilização devem então ter a capacidade de interconectar os sensores com o barramento de medidas. Para isto, esta compatibilização, quando o sensor for digital, deve converter o protocolo próprio do sensor para o protocolo próprio do barramento, ou seja, $I^2C$.

Sendo assim o Microcontrolador Leitura lê os dados de cada módulo de compatibilização, requerendo medida conforme programado.

Cada módulo de compatibilização é responsável por realizar medidas, comunicando-se com o sensor a ele conectado, e enviá-las ao Microcontrolador Leitura, quando requisitado. Para isso, é necessário que o módulo possua comunicação $I^2C$, para conectar-se com o barramento de medidas, e comunicação com o sensor, com todas as suas características específicas. Como dito

anteriormente, ainda não está implementada a função para que o usuário envie, via Software Supervisório, as configurações que irão fazer esta comunicação se estabelecer corretamente. Portanto, os módulos desenvolvidos até o momento são pré-programados para enviar comandos e receber respostas de cada sensor especificamente.

Para a construção dos módulos de compatibilização foram utilizados microcontroladores ATMega168PV como dispositivo que adiciona a inteligência ao módulo. Este dispositivo é uma versão de menor capacidade, porém similar aos utilizados como Microcontrolador Mestre e Microcontrolador Leitura.

Para a realização da função de compatibilizar os sinais provenientes dos sensores com o Barramento de Medidas, foram planejados hardwares para transformar os protocolos desejados para I²C, como mostra o diagrama da

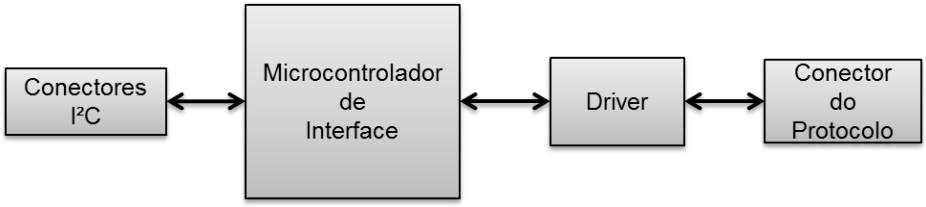



Figura 17: Diagrama do hardware de um Módulo de Compatibilização padrão.

#### 2.1.4.1. MÓDULO SDI-12 e RS-232

O hardware do módulo desenvolvido foi elaborado de modo a poder ter dois canais, um SDI-12 e outro RS-232, em só uma placa (consequentemente, no mesmo endereço I²C).

O SDI-12 consiste em um protocolo de comunicação serial assíncrono codificado em ASCII. Este padrão opera em uma taxa fixa de transferência de dados, *baud rate* de 1200. Uma de suas características mais importantes é o fato de permitir a conexão de um ponto central para múltiplos pontos periféricos. Deste modo é possível interligar um dispositivo mestre, de aquisição de dados, com vários sensores

inteligentes que compartilham o mesmo meio físico, ou barramento, de comunicação. Isto implica em se definir um endereço para cada sensor, e viabiliza a inclusão, ou exclusão, de sensores sem grandes alterações do arranjo experimental.

O protocolo SDI-12 apresenta um conjunto de comandos fundamentais padrão que normatizam a utilização de sensores de medida. O protocolo possui ainda caracteres especiais que são incluídos em alguns comandos como é o caso do caractere *"Spacing"* e *"Marking"* que consistem em apenas selecionar o estado da linha, para *baixo* e *alto*, respectivamente. Alguns exemplos de comandos e respostas são dados na Tabela 1.

Tabela 1: Exemplos de comandos e respostas no protocolo SDI-12.

| Nome | Comando | Resposta |
|---|---|---|
| **Atividade de Dispositivo** | a! | a\r\n |
| **Ativar dispositivo do endereço 0** | aX1! | a\r\n |
| **Realizar uma medida com o dispositivo 0** | aD0! | a" valores medidos" \r\n |

A comunicação realizada no barramento SDI-12 é multiplexada no tempo. Isto significa que na mesma linha o sentido de transmissão e recepção de dados é alternado. Na Figura 18 é apresentado o diagrama com as especificações de tempo para acesso ao barramento SDI-12. O sinal de Break corresponde a um nível lógico alto por 12 ms e é seguido de um Marking, correspondente a um nível lógico baixo durante pelo menos 8,33 ms. Em seguida é enviado o do comando desejado. Após o envio do comando, o Data Recorder espera a resposta do sensor, que precisa acontecer em até 15 ms. Essa resposta é precedida de um outro sinal de Marking.

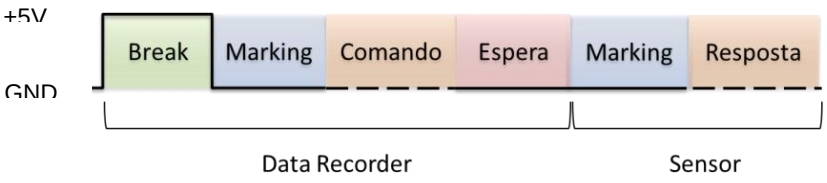



Figura 18: Diagrama de tempo para acesso ao barramento SDI-12.

Para atender às exigências do protocolo, o *driver* necessita ser capaz de, com a ajuda do microcontrolador, multiplexar os sinais de *input* e *output* para que estes dividam a linha de dados corretamente (tarefa realizada por um *buffer tristate*,74LS125), além de utilizar um transistor para compatibilização dos níveis de tensão. A Figura 19a mostra o driver utilizado para esta tarefa. Com esse driver implementado, a adição do conector próprio do protocolo, mostrado na Figura 19b.

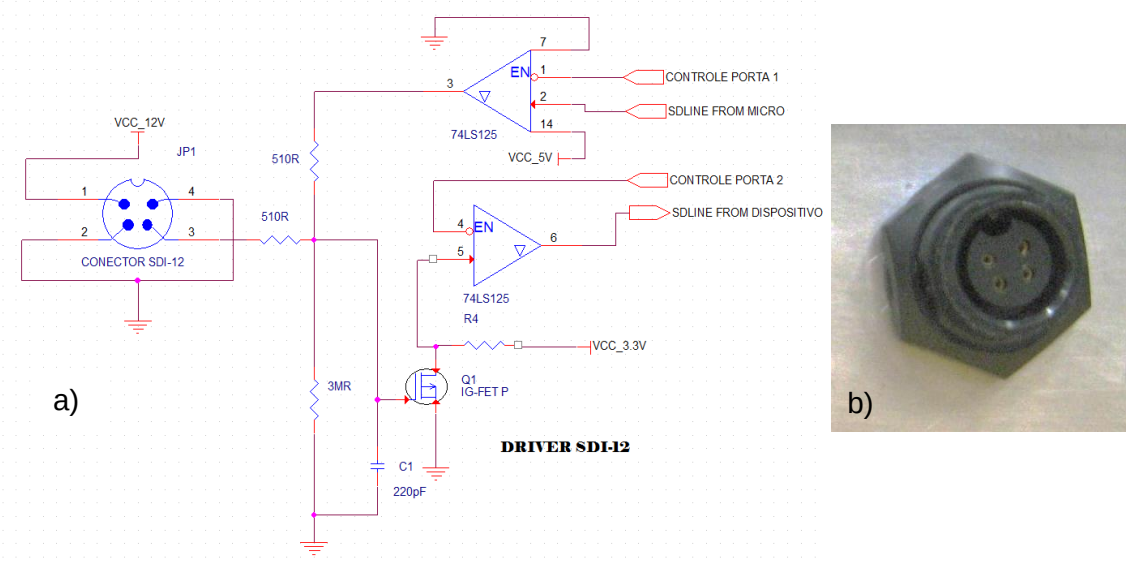



Figura 19: Driver para compatibilização do protocolo SDI-12 com o microcontrolador. Utiliza um transistor para compatibilizar os níveis e um *buffer* (74LS125) para controle e multiplexação da linha de dados do protocolo (a) e conector padrão utilizado no protocolo SDI-12 (b).

O hardware desenvolvido possui um microcontrolador para a compatibilização lógica do protocolo SDI-12 com o protocolo I²C do Barramento de Medidas, além do driver da Figura 19 para a conversão dos níveis de tensão envolvidos. Possui também, da parte do canal RS-232, dispositivos integrados que exercem a função de *driver*

para este protocolo utilizando a comunicação assíncrona do microcontrolador, como vimos na Placa Mãe. Sendo assim, em termos de hardware, o circuito se resume ao conector do Barramento de Medidas, o microcontrolador, o *driver* mencionado e o conector DB9, característico do protocolo RS-232. A Figura 20 mostra o esquemático do circuito empregado neste módulo.

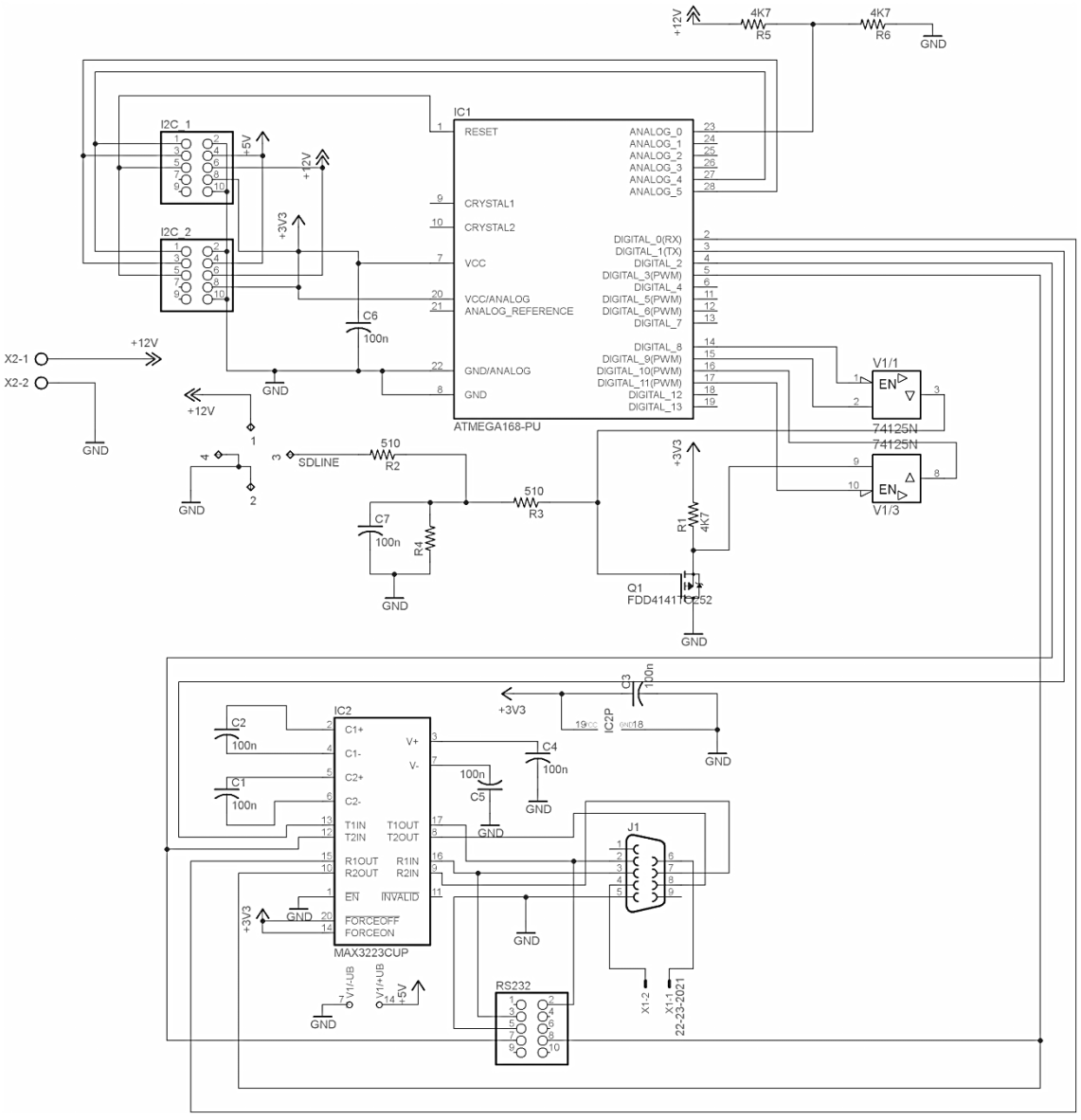



Figura 20: Esquemático da placa desenvolvida para o Módulo de Compatibilização SDI-12. Esse módulo possui um microcontrolador que compatibiliza logicamente os protocolos e um driver para a compatibilização dos níveis de tensão.

Este módulo permite ser configurado como interface de calibração de um dispositivo SDI-12 (pois pode ser conectado diretamente via RS-232 com um

computador), além de sua funcionalidade em trabalhar com dois canais (um SDI-12 e outro RS-232) no Barramento de Medidas I²C.

Uma Nota Técnica encontra-se em redação, abordando detalhadamente todas as funcionalidades desse módulo, que conectado diretamente a um PC, tem como objetivo ser uma interface microcontrolada do padrão SDI-12 para monitoramento ambiental.

### 2.1.5. PLACA SD CARD

Além do armazenamento dos dados gerados, a disponibilidade de dispositivos de armazenamento introduz um ganho metodológico grande ao sistemas de instrumentação. Sistemas embarcados podem adquirir funcionalidades, como as de microcomputadores, que possuem memória não volátil, o que permite que o sistema já seja iniciado com algumas configurações mínimas [17].

Uma memória não volátil é utilizada para conferir algumas funcionalidades ao sistema. Essa memória consiste em um cartão SD e é necessária, principalmente, para que aconteça a inicialização do sistema com as características desejadas.

O cartão SD é o cartão de memória utilizado neste trabalho, escolhido principalmente pela sua versatilidade. Este cartão possui um protocolo próprio de comunicação para sua escrita e leitura, porém, nos casos onde o cartão possui até 2GB de memória (os chamados *Standard Capacity SD Memory Card*), pode ser utilizado o protocolo SPI (que é implementado via hardware no microcontrolador escolhido) [18]. Como padrão, este dispositivo possui uma arquitetura baseada em blocos de 512 bytes, o que foi aproveitado para definir o tamanho dos blocos dentro do sistema desenvolvido.

Dentro dessa arquitetura de memória, os primeiros 12 blocos foram separados para que ocorram as configurações do sistema, necessárias para a sua

inicialização e correta configuração dos sensores. Esse cabeçalho foi separado de forma que o primeiro dos 12 blocos é utilizado para configurações gerais (como tempo entre cada aquisição e modo de operação), assim como armazenamento do endereço de memória da última medida realizada. O segundo bloco é utilizado para configurar o sensor conectado à porta serial, suas características e comandos. No terceiro ao décimo segundo blocos são armazenadas as configurações dos sensores ligados ao barramento de medidas, possibilitando, portanto, até dez sensores acoplados em um único experimento.

Além das configurações, outra importante função do cartão SD é o armazenamento dos dados gerados pelos sensores. Esse armazenamento é utilizado como redundância, de grande importância principalmente nos casos em que o usuário esteja com problemas de comunicação ou de energia na estação de supervisão, sendo impossível ter acesso aos dados de imediato. Além disso, a redundância nas medidas é imperativa em casos nos quais é exigido um nível maior de confiabilidade (como é o caso das aplicações em fiscalização). A Figura 21 mostra a distribuição da memória de um cartão de 2 GB utilizado no sistema.

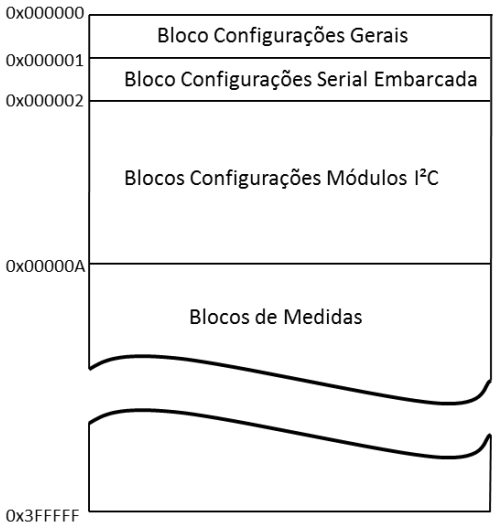



Figura 21: Distribuição dos diferentes blocos na memória do cartão SD. Esse exemplo mostra um cartão de 2GB (que possui 0x3FFFFF blocos de 512 bytes).

Como dito anteriormente, para maior flexibilidade no sistema, o cartão SD deve ser fixado ao chassi de forma que o acesso seja facilitado. Para isso, foi desenvolvida uma placa cuja finalidade é abrigar o cartão de memória e facilitar sua fixação ao chassi. Como os outros módulos, este também é conectado à Placa Mãe Através de um *flat cable* de 10 pinos. Sua pinagem e circuito encontram-se ilustradas na Figura 22.

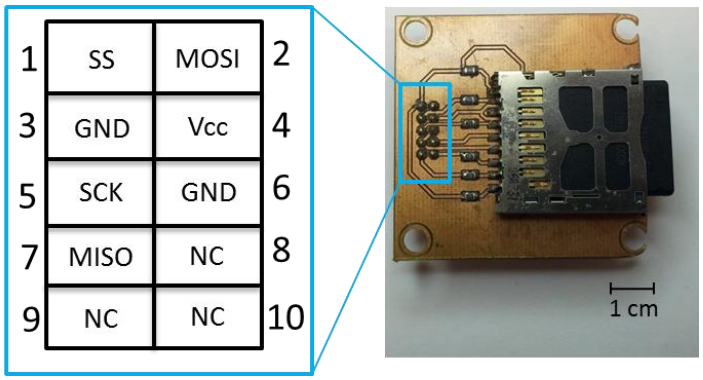



Figura 22: Placa utilizada para a fixação do cartão SD no chassi do dispositivo e pinagem do conector com a placa mãe, onde SS = Ship Select, MOSI = Master Out Slave In, MISO = Master In Slave Out, SCK= Serial Clock e NC = não conectado.

### 2.1.1. CONFIGURAÇÃO

As variáveis configuradas pelo Sistema Supervisório serão essenciais para o funcionamento do experimento com os dispositivos corretos acoplados aos módulos. Para que seja realizada essa configuração, após o preenchimento da tabela de configurações, o Software Supervisório monta blocos de 512 bytes (para que sejam enviadas para o cartão SD no sistema embarcado) contendo todas essas variáveis de configuração.

Após a formação pelo Sistema Supervisório, os blocos de configuração são enviados para o Sistema Embarcado, chegando no Microcontrolador Mestre. Este, por sua vez, configura suas variáveis internas, necessárias para o controle do processo, e

encaminha os blocos de configuração para o cartão SD, para que as configurações estejam armazenadas em caso de uma nova inicialização do sistema.

O Microcontrolador Mestre encaminha então esses blocos para o Microcontrolador Leitura para que este também realize as configurações necessárias para a operação, que neste caso são as características dos sensores a ele acoplados. Ele também será responsável por, em versões futuras do sistema, enviar, via Barramento de Medidas, as configurações de cada sensor para cada endereço (cada Módulo de Compatibilização). Cada Módulo de Compatibilização receberá então somente as configurações relativas ao sensor a ele acoplado. Esse processo de configuração, que passa do Sistema Supervisório até os Módulos de Compatibilização são ilustrados na Figura 23, processo esse que foi implementado até a recepção das configurações pelo Microcontrolador Leitura.

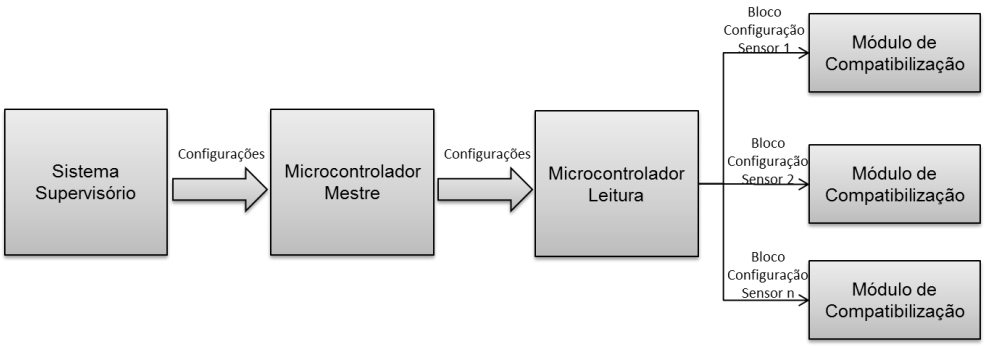



Figura 23: Processo de configuração do sistema. Inicia com o preenchimento das configurações no Software Supervisório, passando pelos microcontroladores da Placa Mãe. Futuramente, as configurações devem chegar individualmente nos módulos que se comunicam com os sensores, para que o sistema seja totalmente configurável via Software Supervisório.

Como dito anteriormente, os primeiros blocos do cartão de memória são utilizados para armazenar variáveis de configuração. O formato desses blocos é feito de forma que cada variável de configuração seja representada por um mnemônico característico e seu valor seja armazenado entre parênteses ao lado do mnemônico (e.g. modo de operação 3 = “MOD(3)”). Desta forma, cada variável vai sendo armazenada para posterior uso pelos microcontroladores, que irão inicializar suas

variáveis internas de acordo com os blocos de configuração, como mostrado na Figura 24. Como cada bloco do cartão de memória contém 512 bytes, caso sobre espaço dentro de um bloco o mesmo será completado com o caractere ASCII “?”.

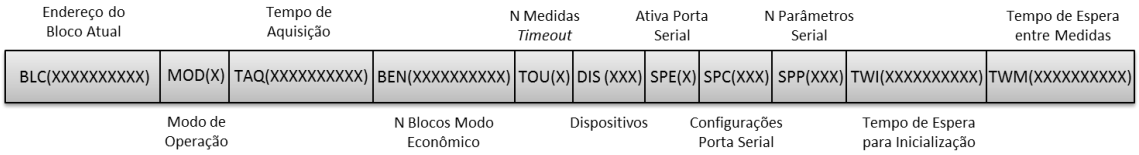



Figura 24: Exemplo de um bloco de configuração. As variáveis representadas por mnemônicos têm seu valor armazenado dentro dos parênteses. Este exemplo específico corresponde ao primeiro bloco de configurações, que contém as configurações gerais do experimento, ou seja, configurações de tempos, modo e quantidade de sensores conectados. Para completar o bloco de 512 bytes, o mesmo é preenchido com o caractere “?”.

O bloco mostrado sugere diversas configurações para o experimento. Esse bloco é o primeiro do cartão SD, e armazena, além das configurações, o endereço atual da ultima medida no cartão de memória, representado pelo mnemônico “BLC”. Esse valor é necessário para que o sistema não sobrescreva as medidas anteriores quando religado, pois situa o Microcontrolador Mestre sobre qual endereço ele deve iniciar a gravação.

Além do primeiro bloco, mostrado na Figura 24, existem outros que são específicos para as configurações dos sensores. O segundo bloco é responsável por armazenar as configurações para a operação da porta serial embarcada. Essas configurações são relativas às palavras de comando (necessárias para inicialização e requisição de medidas do sensor) e ao formato das palavras recebidas (inicialização e medidas). As configurações contidas nos blocos de configuração e seus respectivos mnemônicos são mostrados na Tabela 2.

Tabela 2: Descrição das variáveis de configuração do Sistema Embarcado e seus mnemônicos.

| Nome Variável | Mnemônico | Função |
|---|---|---|
| endereco_atual_sd_card | BLC | Indica o bloco da última medida no SD Card |
| modo_operacao | MOD | Modo de operação |
| tempo_aquisicao | TAQ | Tempo entre aquisições no modo automático |
| n_bloco_envio | BEN | Quantos blocos serão armazenados antes de envio no modo econômico |
| n_timeout | TOU | Quantas medidas são admitidas serem perdidas antes de resetar sistema |
| n_dispositivos_i2c | DIS | Número de dispositivos conectados ao barramento I2C |
| serial_port_en | SPE | Enable da porta serial do Microcontrolador Leitura |
| serial_port_config | SPC | Configurações de Baud Rate, start bit, stop bit e paridade |
| serial_port_n_param | SPP | Número de parâmetros requeridos do dispositivo serial |
| serial_port_cmd_init | SCI | Comando de inicialização do dispositivo serial |
| serial_port_fmt_init | SFI | Formato de inicialização do dispositivo serial (se houver) |
| serial_port_cmd_leit | SC**n** | Comando de leitura do parâmetro **n** da porta serial |
| serial_port_fmt_leit | SFn | Formato de leitura do parâmetro n da porta serial |
| n_param_a_medir | INP | Número de parâmetros requeridos do dispositivo I2C |
| protocolo | IPR | Protocolo do dispositivo |
| config_protocolo | ICP | Configurações necessárias para o protocolo do dispositivo |
| endereco_i2c | IAD | Endereço I2C do dispositivo |
| cod_sensor | IC**n** | Código mnemônico a ser utilizado pela medida do parâmetro **n** do dispositivo |
| cmd_init | ICI | Comando de inicialização do dispositivo I2C |
| fmt_init | IFI | Formato de inicialização do dispositivos I2C(se houver) |
| fmt_leitura | FLn | Formato de leitura do parâmetro n do dispositivo I2C |
| cmd_leitura | CL**n** | Comando de leitura do parâmetro **n** do dispositivo I2C |

Os demais blocos são relativos aos sensores conectados ao barramento de medidas, sendo cada bloco responsável por um único sensor. Esses blocos carregam em si as informações de endereço $I^2C$ do sensor, configuração do protocolo utilizado, assim como as informações de comando e formato analogamente ao bloco da porta serial.

Configurando corretamente o sistema, pode-se realizar experimentos diversos sem que haja necessidade de reprogramação dos microcontroladores e módulos

envolvidos. Esta é uma característica que só foi possível ser adicionada com o desenvolvimento dos blocos de configuração e da memória não volátil embarcada no sistema.

### 2.1.2. MÓDULO DE COMUNICAÇÃO

Com as medidas prontas e formatadas nos blocos de medidas, a próxima etapa do processo é permitir ao usuário o acesso a esses dados. Para isso, será abordado nesta sessão, o modo de transmissão desses dados gerados para um dispositivo de supervisão, que pode ser um computador pessoal, um *tablet*, ou qualquer outro dispositivo que tenha conformidade com os meios de comunicações propostos neste trabalho.

Um dos objetivos específicos deste trabalho é deixar o sistema versátil o suficiente para estar inserido neste cenário de crescente mudança nas tecnologias em comunicações. Pensando deste modo, a arquitetura desenvolvida inclui uma interface RS-232 para a saída dos dados. Essa interface foi escolhida devido a sua fácil integração com outros tipos de comunicações através de módulos que convertem o sinal RS-232 em, por exemplo, Wi-Fi, *links* de rádio dedicados etc.

No momento, o dispositivo permite três tipos de conexão: serial, *radio tranceiver* e WiFi. A Figura 25 mostra os circuitos comerciais utilizados para a transformação dos sinais seriais para o método de comunicação desejado.

a) b)

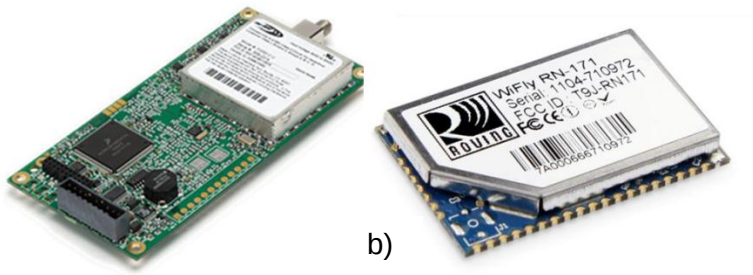



Figura 25: *Radio Tranceiver* (a), retirado de “www.freewave.com”; e conversor WiFi (b), retidado de “www.rovingnetworks.com”.

Através de um módulo RS-232/WiFi, o sistema obtêm acesso a redes regidas pelo padrão IEEE 802.11. Esse padrão, estabelecido em 1999, se tornou parte do cotidiano rapidamente. Ele é responsável por prover conexão sem fio para estações fixas, portáteis ou móveis, dentro de uma rede local [19]. O padrão define um meio de controle de acesso (MAC) e diversas especificações na camada física do modelo OSI. Esses objetivos vão ao encontro do objetivo deste documento, possibilitando a montagem de experimentos de monitoramento remoto que incluem um supervisório de simples operação, podendo inclusive ser utilizado em *tablets* em experimentos de campo.

Outra forma de comunicação remota é um *link* de dados via rádio. Através de dois módulos (um no sistema remoto e outro no supervisório) que utilizam frequências de rádio específicas para este tipo de aplicação, uma comunicação sem fio pode ser estabelecida [20].

O *radio tranceiver* opera com uma portadora de 900MHz, possui um alcance de aproximadamente 32km em linha de visada e consumo de corrente de 650mA durante a transmissão dos dados [21]. Já o modem WiFi provê ao sistema características de comunicação do padrão IEEE 812.11 e corrente máxima em transmissão de 190mA [22].

## 2.2. PROGRAMAÇÃO E USABILIDADE

Nesta sessão é descrito o conteúdo de software que é utilizado no processo de aquisição de dados, formação dos blocos necessários para a operação, configuração e interação do supervisório com o sistema.

### 2.2.1. OS MODOS DE OPERAÇÃO

Para que fossem atendidas as diversidades dos experimentos de campo, houve necessidade da implementação de alguns métodos de medição que pudessem

ser escolhidos através das configurações citadas anteriormente. São sete os modos de operação, como mostrado na Tabela 3:

Tabela 3: Tabela descritiva dos Modos de Operação.

| Modos de operação | | | | | |
|---|---|---|---|---|---|
| Número | Modo | Gravação SD | Necessidade do Supervisório | Requisição de medida | Envio de dados |
| 0 | Central | Não | Sim | Supervisório | Sim |
| 1 | Local | Não | Sim, para recepção dos dados | Hadware ou Botão | Sim |
| 2 | Contínuo | Não | Sim | Tempo de aquisição configurado | Sim |
| 3 | Central com SD | Sim | Sim | Supervisório | Sim |
| 4 | Local com SD | Sim | Não | Hadware ou Botão | Sim |
| 5 | Contínuo com SD | Sim | Não | Tempo de aquisição configurado | Sim |
| 6 | Econômico | Sim | Não | Tempo de aquisição configurado | Sim, por grupo de blocos |

- Modo 0 ou Modo Central: Cada medida é realizada a partir da chegada de uma requisição vinda do Sistema Supervisório. O Sistema Supervisório requisita a medida ao Microcontrolador Mestre que repassa a requisição ao Microcontrolador Leitura e espera o bloco de medida. O bloco é então enviado para o supervisório sem que seja armazenado no cartão de memória.

- Modo 1 ou Modo Local: A requisição de medida é enviada por hardware ou botão, localmente. Apesar de não necessitar da presença do supervisório para a requisição das medidas, este é necessário para a recepção dos dados dos blocos de medida, já que este modo também não armazena os dados no cartão de memória.

- Modo 2 ou Modo Contínuo: Neste modo, não é necessário o envio de nenhuma requisição de medidas externa. O sistema começa a medir automaticamente, dando, entre as medidas, um intervalo correspondente ao valor (em milissegundos) da variável de configuração “Tempo de Aquisição”, descrita

anteriormente. Assim como os anteriores, este modo também não armazena os dados no cartão SD.

- Modos 3, 4 e 5: Esses modos são os correspondentes aos modos 0, 1 e 2, respectivamente, porém utilizando o cartão SD para o armazenamento dos blocos de medida. Além da vantagem de segurança da informação conferida por estes modos, eles conferem ao sistema certa autonomia na operação em campo, ou seja, são independentes do supervisório para sua operação básica (com exceção do modo 3, que ainda necessita da requisição da central para iniciar cada medição). Por segurança, nos modos nos quais as medidas são armazenadas no cartão de memória, o armazenamento acontece antes do envio.

- Modo 6 ou Modo Econômico: Em algumas aplicações de monitoramento remoto, existe uma necessidade de economia de energia, seja por estar grandes intervalos de tempo sem abastecimento da concessionária responsável, ou por ser alimentado com células fotovoltaicas, nas quais a alimentação a noite é dependente de baterias. Sendo assim, o Modo Econômico opera com requisições de medidas iguais as do modo contínuo, porém, seu modo de armazenamento se difere dos demais. Utilizando a variável de configuração apresentada anteriormente (BEN), o sistema irá armazenar uma quantidade de blocos de medida igual ao valor desta variável antes de enviar esse grupo de blocos para o Sistema Supervisório. Essa característica possibilita que o Módulo de Comunicação do sistema fique em estado *idle* (estado de economia de energia, onde só são realizadas operações essenciais) a maior parte da operação, economizando energia. Se o valor da variável for zero, o sistema não enviará bloco ao supervisório, fazendo a operação ser totalmente local, somente armazenando as medidas para posterior visualização.

Os Modos de Operação são implementados no Microcontrolador Mestre através de diferentes rotinas, escolhida através da variável “modo_operacao” mostrada na Figura 26.

```
void tipo_loop()
{
    switch (modo_operacao)
    {
    case 0:
        loop_central();
        break;
    case 1:
        loop_botao();
        break;
    case 2:
        loop_remoto();
        break;
    case 3:
        loop_central();
        break;
    case 4:
        loop_botao();
        break;
    case 5:
        loop_remoto();
        break;
    case 6:
        loop_remoto();
        break;
    }
}
```

Figura 26: *Switch* responsável pela escolha do Modo de Operação, presente no *firmware* do Microcontrolador Mestre. Cada função (*loop*) indica a rotina na qual o sistema deverá seguir.

A utilização do cartão SD ou não é programada dentro de cada rotina, justificando a utilização da mesma função para os modos 0 e 3; 1 e 4; e 2, 5 e 6.

A Figura 26 também mostra o local dentro do programa do Microcontrolador Mestre onde devem ser incluídos outros modos que venham a ser idealizados no futuro.

### 2.2.2. FORMAÇÃO DO BLOCO DE MEDIDA

Com o sistema completamente configurado e o modo de operação definido, o processo de medidas pode ter início. Esta tarefa é feita conjuntamente, através do

Microcontrolador Mestre, do Microcontrolador Leitura e dos módulos conectados ao barramento. Essa sessão se compromete então a detalhar o processo de formação do bloco de medida relacionado ao bloco de “Ler Dados e Formar o Bloco”, no fluxograma da Figura 16.

A medida se inicia quando o Microcontrolador Mestre, independentemente do modo de operação utilizado, envia uma requisição de medida para o Microcontrolador Leitura. Isto ocorre com o envio de um caractere “M” pelo canal UART que conecta os dois microcontroladores. O Microcontrolador Leitura então envia uma requisição de medida para todos os endereços $I^2C$ conectados ao barramento, assim como para o sensor conectado à porta serial embarcada, que corresponde ao mesmo caractere “M". A Tabela 4 mostra os comandos que podem ser enviados pelo Microcontrolador Mestre para o Microcontrolador Leitura pela linha UART que os conecta.

Tabela 4: Comandos enviados do Microcontrolador Mestre para o Microcontrolador Leitura.

| Comando | Função |
|---|---|
| C | Configurar Microcontrolador Leitura |
| M | Medir |
| t | Transmitir Bloco de Medidas |

Após serem requisitados, os módulos enviam os comandos de medida (configurados previamente através do bloco de configuração correspondente) aos sensores a eles conectados. Eles recebem então a resposta dos sensores e a formata de acordo com o mnemônico configurado para cada parâmetro. Por exemplo, se o mnemônico de um sensor for ABC, o módulo o formatará como ABC(“dados recebidos”).Com os dados formatados, o Microcontrolador Leitura faz então a aquisição de cada módulo para a formação do bloco de medida.

Assim como os blocos de configuração, o bloco de medida possui 512 bytes, para que seja facilmente armazenado no cartão SD. Dentro desses 512 bytes, é inserido um cabeçalho contendo o bloco correspondente ao endereço para o

armazenamento no cartão de memória além de um índice correspondente a quantas vezes o sistema foi iniciado, o que pode ser útil para identificar quando um experimento inicia e termina. Este cabeçalho é seguido pelos dados do sensor da porta serial embarcada e dos módulos, na ordem em que foram configurados, ou seja, primeiro os dados da serial, depois os dados do sensor conectado ao módulo correspondente ao primeiro bloco de configuração do Barramento de Medidas, e assim em diante.

Com o bloco de medidas formado, o Microcontrolador Leitura aguarda uma requisição vinda do Microcontrolador Mestre para lhe enviar este bloco (caractere “t”), realizando então esse envio. Os próximos passos são o armazenamento dos blocos no cartão SD (dependendo do modo de operação) e o posterior envio desses dados ao sistema supervisório. A Figura 27 mostra todo o processo de formação, armazenamento e envio do Bloco de Medidas

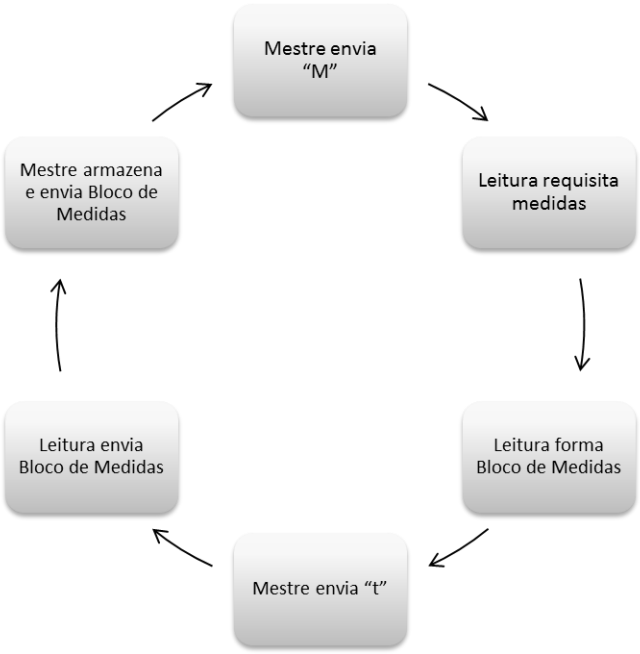



Figura 27: Processo de formação, armazenamento e envio do Bloco de Medidas.

### 2.2.3. O SISTEMA SUPERVISÓRIO

O Sistema Supervisório é programado através da linguagem de programação visual LabView, em sua versão 10.

Apesar de o sistema embarcado funcionar de forma independente, para que os dados gerados nas medidas dos sensores sejam de fato aproveitados para a tomada de qualquer decisão em tempo real é necessário que seja utilizado um software para aquisição desses dados enviados pelo módulo de comunicação.

O software do Sistema Supervisório deve ser capaz de adquirir as medidas, colocá-las em um gráfico em tempo real e salvá-los em formato adequado para posterior exportação para análises gráficas e/ou estatísticas. A Figura 28 ilustra este software em operação para um sistema de monitoramento de temperatura.

Para a realização das medidas, o software necessita identificar quando é recebido um Bloco de Medidas e formatá-lo adequadamente. Esse trabalho é feito separando os mnemônicos citados na sessão “Formação do Bloco de Medidas” e adquirindo cada valor relacionado a esses mnemônicos.

São adquiridos então valores do bloco relativo ao endereço da medida no cartão SD (nos Modos de Operação em que o cartão está ativo); o número do experimento (variável explicada anteriormente a qual é incrementada a cada reinicialização do sistema); os valores das medidas do Sensor Serial Embarcado; assim como os valores de cada medida realizada no barramento.


Figura 28: Painel do software para análise de dados e gravação dos mesmos em arquivos de texto. Os dados adquiridos são formatados para a formação de um arquivo contendo data, hora, entre outros parâmetros para a posterior análise (acima), assim como inseridos em um gráfico do tipo *chart* (abaixo).

Esses valores são definidos em variáveis do programa supervisório, formando então uma linha contendo esses valores, além de alguns outros parâmetros adicionados pelo próprio software supervisório. Esses parâmetros são relativos à data, hora e outros com relação ao tempo de aquisição do Bloco de Medidas.

A linha gerada pode então ser adicionada a um arquivo de texto, onde os dados adquiridos estarão guardados para a posterior análise.

Além dessas funções, o software também precisa ser capaz de realizar as configurações explicitadas em tópicos anteriores. Com esse objetivo, foi criado uma Interface de Controle do sistema, onde pode se efetuar essa configuração. A Figura 29 mostra algumas das configurações que podem ser adaptadas a cada aplicação.

A Interface de Controle também é responsável pela realização de medidas de teste, quando necessárias. Para isso, o software possui uma aba para a recepção de blocos de medida e um indicador caso algum erro venha a ocorrer, como mostrado na Figura 30.

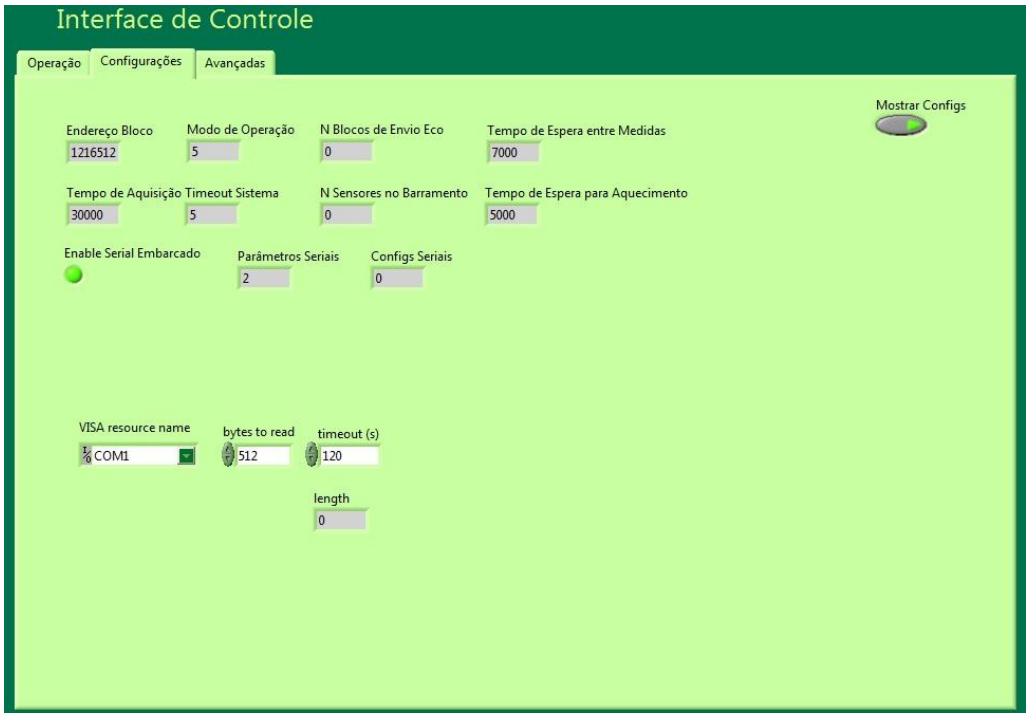



Figura 29: Aba de configurações da Interface de Controle. Através dessa interface é possível realizar algumas configurações básicas no sistema, assim como monitorar o status atual das suas configurações.

Estando no Modo Central, um Bloco de Medidas pode ser adquirido ao pressionar o botão “Ler (Modo Central)”. O sistema então enviará uma requisição de medida para o Hardware Embarcado esperando em resposta um Bloco de Medidas.

Caso o Modo de Operação seja o Contínuo, Local ou Econômico, esta aba atuará de forma passiva, mostrando os Blocos de Medidas que forem chegando à linha serial.

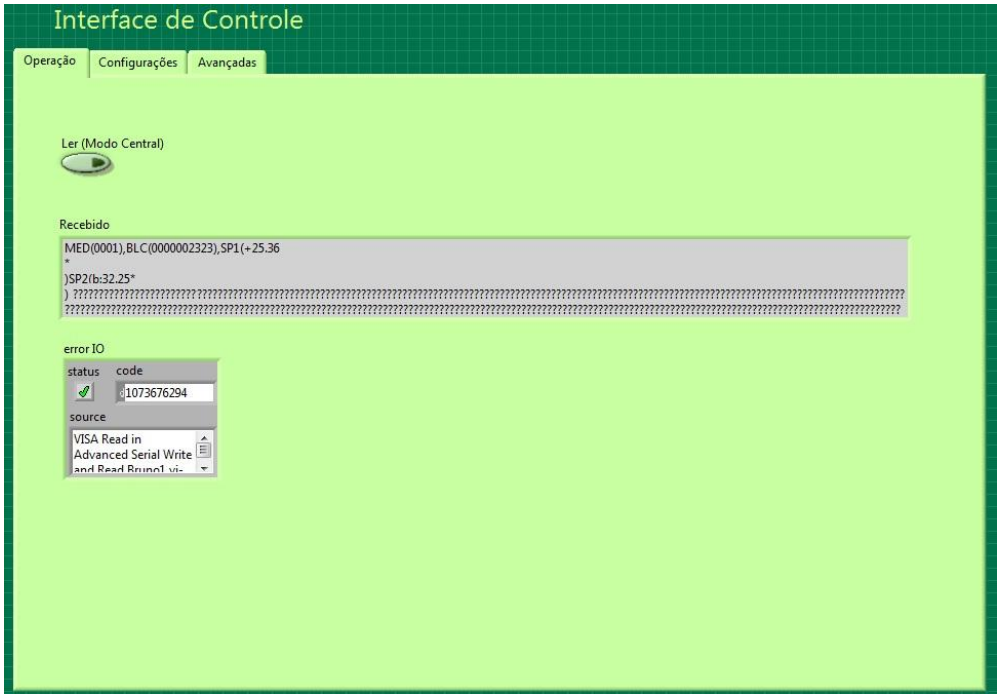



Figura 30: Interface de Controle em sua aba de testes na operação. Possui um botão para que seja feita uma medida no Modo Central, um indicador de erro e a palavra recebida durante o teste.

Uma aba de “Avançadas” fornece a opção de entrar no modo de Configurações Avançadas para que seja efetuada a configuração de todo o experimento, ou seja, para que se prepare o sistema com os respectivos sensores, tempos de medida e *timeout* corretos (valores calculados de acordo com os tempos necessários para a realização das medidas, normalmente encontrados nos manuais dos sensores), modo de operação, entre outras configurações (todas descritas no Anexo C).

Para a operação desta parte do programa – mostrada na Figura 32 – é necessário algum conhecimento, tanto do experimento em questão, como características dos sensores utilizados e do sistema como um todo. Isso se dá pelo fato do operador ter que ser capaz de adicionar os comandos necessários para a

realização de medidas pelos sensores do experimento, assim como conhecer o formato da resposta esperada (parâmetros normalmente encontrados nos manuais de operação fornecidos pelos fabricantes dos sensores). Como exemplo, um dos sensores utilizados neste trabalho – o sensor de temperatura a fibras ópticas NOMAD – possui como comando de medida “t\r”, esperando como resposta, oito bytes contendo a temperatura no formato “%f*\r” (por exemplo +25.00*\r). A Figura 31 mostra a tabela a ser preenchida no Sistema Supervisório para que seja possível a conexão do sensor.


Figura 31: Tabela a ser preenchida com os dados do sensor a ser utilizado. Nela é colocada uma identificação para posterior formatação (Cód Sensor), o endereço I²C do sensor no Barramento de Medidas (Endereço), assim como os comandos, respostas e timeout para inicialização e para operação do sensor.

Apesar das configurações não estarem completamente implementadas, já que os módulos de compatibilização ainda não são flexíveis para a recepção dessas configurações, todos os parâmetros dessa configuração já podem e precisam ser incluídos via software supervisório, pois os blocos de configuração são formados através dos dados fornecidos nesta etapa. A inclusão de todas essas configurações é necessária para que, tanto o Microcontrolador Mestre quanto o Microcontrolador Leitura, realizem suas funções corretamente. Além das configurações de *timeout*, tempo de aquisição e tempo inicialização, é necessário configurar alguns outros parâmetros, mostrados na Figura 32. O campo “Endereço” deve ser preenchido com o endereço I²C correspondente a cada sensor no barramento de medidas, definido através de conexões no hardware do respectivo módulo. É necessário também determinar um código para a identificação de cada um dos sensores (campo “Cód Sensor”), que servirá para identificar e diferenciar as medidas de cada sensor no bloco de medidas. Já os comandos para a inicialização dos sensores (campo “Init”) e para solicitação das medidas (campo “CmdLeitura”), e os formatos da palavra de

inicialização enviada pelo sensor no início da operação (campo “Init Formato”) e da medida recebida (campo “FormatLeitura”*n*) após o envio do comando de leitura, consistem em códigos específicos encontrados nos manuais de cada sensor utilizado.

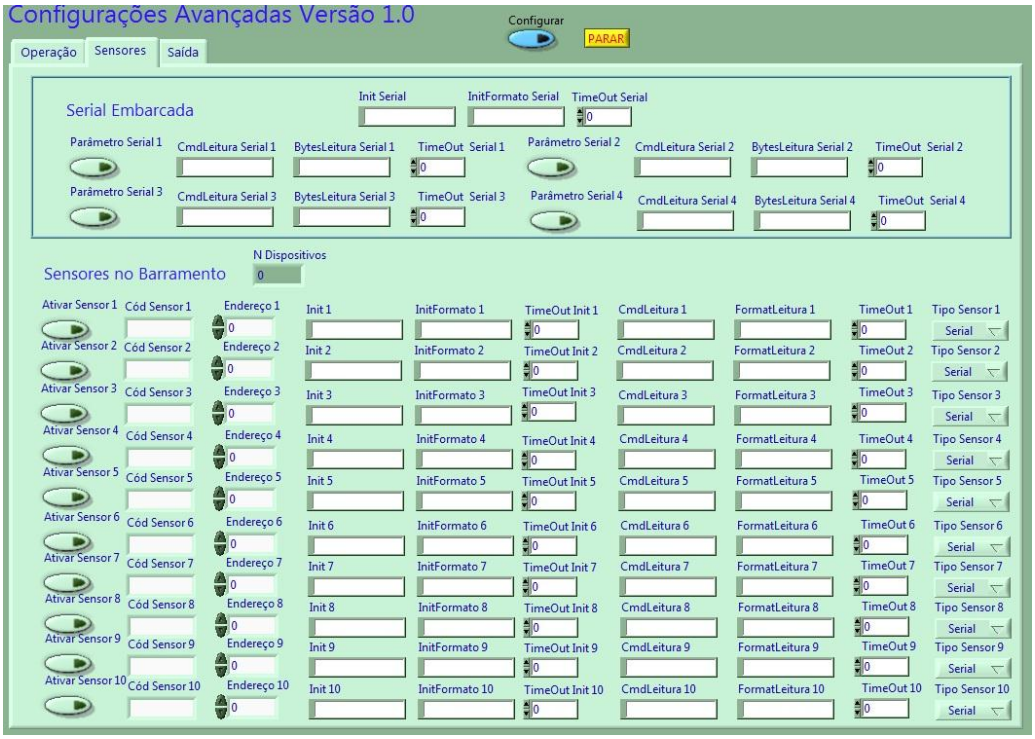



Figura 32: Configurações Avançadas do experimento, ou seja, configura o sistema para sua conexão com os sensores relativos a um experimento específico.

## 2.3. MONTAGEM DO EXPERIMENTO

As ferramentas necessárias para a montagem de um experimento de monitoramento, com exceção dos módulos de compatibilização que ainda não são configuráveis, foram descritas até o momento. É necessário então que essas ferramentas sejam utilizadas de forma correta, como será descrito a seguir.

O primeiro passo para a montagem do experimento é a escolha dos sensores que serão utilizados. Deve-se observar suas características técnicas, para que se saiba se o sensor é compatível eletricamente, assim como se existe algum Módulo de Compatibilização para o estabelecimento da conexão com o Barramento de Medidas.

Caso o sensor seja compatível com o sistema, é adicionado o hardware do Módulo de Compatibilização correspondente. Desta forma, o sensor já está totalmente incorporado em termos de hardware, restando somente configurá-lo para que o sistema seja capaz de realizar medidas com o mesmo.

A configuração dos parâmetros necessários para o funcionamento do sensor é feito via Sistema Supervisório. Após preencher e enviar a tabela das configurações avançadas do sistema, os parâmetros de comandos de envio, formatos das respostas e outras configurações características do protocolo do sensor já estarão armazenados para que o sistema realize as medidas de acordo com o sensor conectado ao barramento.

O Sistema Supervisório também é utilizado para configurar parâmetros do experimento como Modo de Operação e os tempos de aquisição, espera etc. Essas configurações são feitas de acordo com o objetivo desejado para o experimento: medidas em campo (utilizando um modo que dê mais versatilidade ao experimentador, que poderá fazer medidas quando julgar necessário) ou modo monitoramento remoto (utilizando o modo contínuo para a gravação dos dados no cartão de memória), como mostram os diagramas da Figura 33.

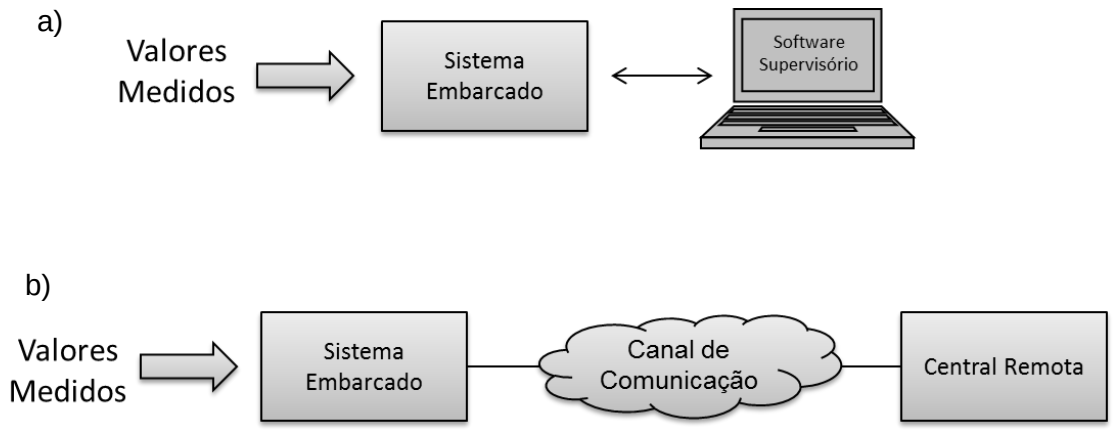



Figura 33: Sistema configurado para medidas em campo (a) ou monitoramento remoto (b).

O software do Sistema Supervisório necessita de ajustes em sua programação para que sejam gerados os gráficos em tempo real de novos sensores conectados, assim como também necessita a mudança da *string* de formatação (a *string* que separa cada valor do Bloco de Medidas em variáveis) para que o arquivo de texto seja gerado corretamente.

## 3. RESULTADOS EXPERIMENTAIS

Por se tratar de um projeto de instrumentação científica, este trabalho possui o principal resultado nos próprios dispositivos, circuitos eletrônicos e programação desenvolvidos: A Placa Mãe com os programas do Microcontrolador Mestre e do Microcontrolador Leitura em linguagem C, a Placa SD Card, o circuito do Módulo SDI-12/RS-232 e as funções do Sistema Supervisório em LabView.

Como já mencionado, a Placa Mãe sem módulos adicionais possui comunicação RS-232 com o Software Supervisório, assim como um canal de comunicação para a conexão de um Sensor Serial Embarcado. Porém, pode-se adicionar módulos de comunicação diversos, que sejam compatíveis com o protocolo supracitado. Foram testados dois módulos: um *radio tranceiver* e um modem WiFi.

Além de medidas realizadas com a adição de sensores e aquisição de dados, alguns valores importantes para o sistema foram determinados, valores estes essenciais para a validação técnica do mesmo.

As características elétricas do sistema são de extrema importância operacional. Como dito anteriormente, o dispositivo possui uma tensão de alimentação de 12V. Em termos de corrente de consumo, foram medidos alguns valores, dependendo da configuração do sistema. Quando não existem módulos acoplados, o sistema consome em torno de 20mA. Com o sistema operando com uma sonda SDI-12 do modelo Quanta, da Hydrolab e comunicação via rádio, o consumo fica em torno de 135mA durante as medições e em torno de 720mA durante a transmissão de dados via rádio (as especificações técnicas do sistema podem ser encontradas no Anexo B).

Em relação ao armazenamento, podemos fazer o seguinte cálculo: O valor máximo de um cartão SD para a utilização do sistema com a arquitetura atual é de 2GB (fato relacionado com o endereçamento via protocolo SPI [18]). Desta maneira,

como 12 blocos de 512 bytes estão reservados às configurações, temos a equação abaixo:

$$2GB = 2 \times 2^{30} bytes - (12 \times 512 bytes) = 2147477504\ bytes$$

Como o Bloco de Medidas possui 512 bytes, é possível o armazenamento de 4.194.292 medidas. Em termos de tempo de experimento, se considerarmos uma medida a cada minuto (situação da aplicação para o grupo de Agrobiologia da Embrapa), poderão ser gravados cerca de 8 anos ininterruptamente.

## 3.1. MEDIDAS EM LABORATÓRIO

Partes integrantes do sistema que será futuramente empregado em monitoramento remoto, foram testadas em ambiente de laboratório.

Para isso, foi utilizado um sensor de temperatura a fibras óticas (NOMAD), que possui uma saída digital RS-232 que foi conectada à porta serial embarcada do dispositivo.

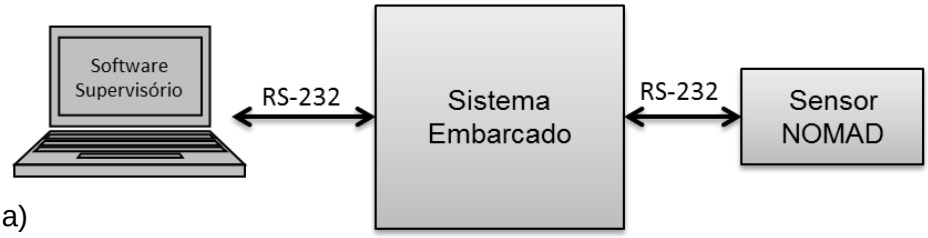
A comunicação foi configurada para trabalhar com um Baud Rate de 9600, sem controle de fluxo, oito bits de dados, sem paridade e com 1 bit de parada.

Em relação às medidas realizadas pelo NOMAD, foi configurado para enviar os dados com duas casas decimais. Neste caso, o sistema foi conectado à rede elétrica, não sendo utilizadas baterias. A Figura 34 mostra esse arranjo utilizado para as medidas de temperatura.

O sistema foi então configurado para aquisição de dados deste sensor, no Modo de Operação Contínuo, onde não é necessário o envio de requisições de medidas pelo Software Supervisório, e o sensor foi conectado e configurado para operar utilizando a conexão para um Sensor Serial Embarcado. O dispositivo envia as

medidas de acordo com o tempo de aquisição configurado previamente (que no caso foi configurado para 1 minuto).

a)




b)

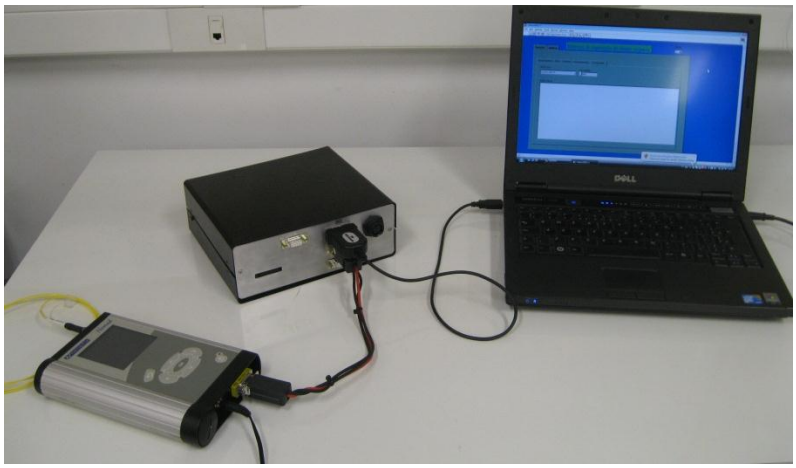


Figura 34:Diagrama (a) e montagem (b) do arranjo experimental utilizado para a medição de temperatura em ambiente de laboratório.

A tabela de configurações foi preenchida no Software Supervisório com os comandos e formatos para uma efetiva comunicação com o NOMAD. Para a medida, é enviado o comando “t\r”, esperando a resposta no formato “%f\r*”, onde “%f” é o valor da temperatura medida no tipo *float* com a inclusão do sinal (positivo ou negativo), como por exemplo, “+25.00*\r” (medidas em graus Celsius).

A Figura 35 mostra os dados gerados por este dispositivo, que mostrou estabilidade durante o experimento, que durou aproximadamente uma semana, medindo a temperatura ambiente do laboratório. Como as medidas foram feitas durante uma semana, após o desligamento do sistema de refrigeração do laboratório, nota-se a variação de temperatura entre o dia e a noite, gerando os patamares indicados pelas setas.

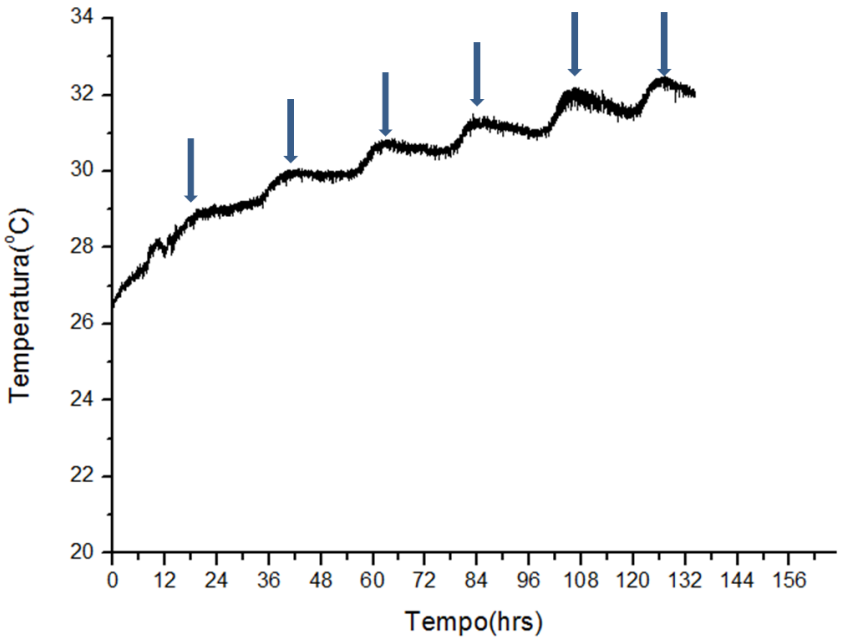



Figura 35: Teste em laboratório do dispositivo com um sensor de temperatura acoplado à conexão do Sensor Serial Embarcado(sensor a fibra óptica). As medidas foram feitas durante uma semana, aquisitando a temperatura ambiente do laboratório. As setas representam os patamares de mudança de temperatura relativos à diferença na variação das temperaturas entre dias e noites.

## 3.2. MEDIDAS DE CAMPO

Com o objetivo de testar alguns módulos do sistema, assim como planejar algumas partes da arquitetura proposta neste trabalho, foram feitas medidas com uma versão anterior à deste trabalho. O sistema citado foi então utilizado com uma sonda multiparamétrica, medindo-se os pontos de um canal de produção de vinhaça em colaboração com o grupo de Agrobiologia da Embrapa, coordenado pelo pesquisador Bruno Dias.

Propõe-se a utilização dessa versão do sistema para a análise de processos agroindustriais, como a geração de álcool através da cana de açúcar, pesquisa realizada pelo grupo de Agrobiologia da Embrapa. Uma das preocupações desse grupo de pesquisa é a sustentabilidade do processo de geração do álcool da cana de açúcar em relação às emissões de gases de efeito estufa. Um objeto de análise para essa pesquisa está na formação de um subproduto muito utilizado para a irrigação das

próprias lavouras: a vinhaça de cana de açúcar. O objetivo desta aplicação é analisar os dados obtidos com as medidas realizadas pelo sistema, na usina de bioenergia, em Pirassununga SP; para comparação com os dados de análise de emissão de gases feita pelo grupo de pesquisas em Agrobiologia da Embrapa.

O arranjo montado contava com baterias carregadas por uma célula fotovoltaica para o suprimento de energia elétrica. A comunicação entre o sistema embarcado e o software supervisório foi estabelecida com a utilização de um *radio transceiver* que, como dito anteriormente, opera em 900MHz. A sonda multiparamétrica Quanta é então conectada ao sistema embarcado através de uma versão do módulo conversor SDI-12/I²C. O arranjo é ilustrado na Figura 36.

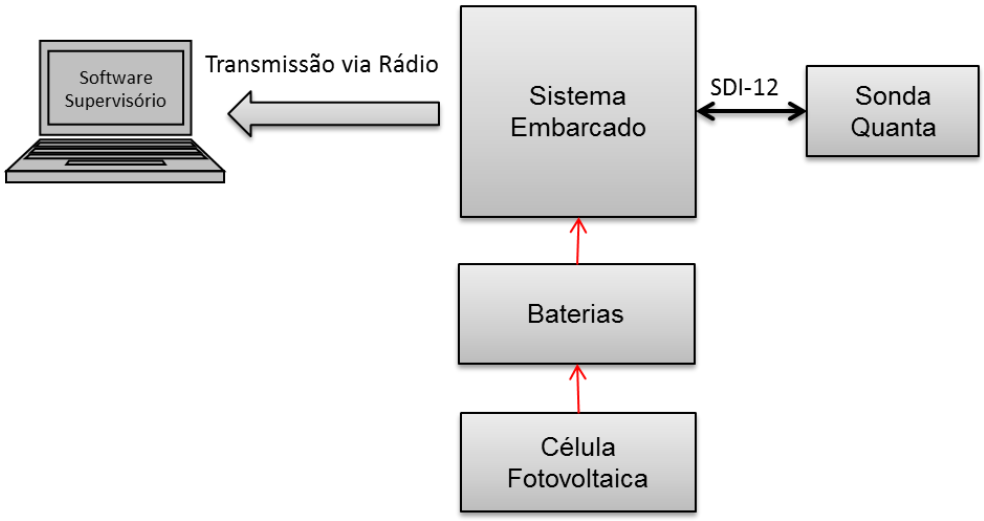



Figura 36: Arranjo experimental utilizado para as medidas na usina de bioenergia.

Neste caso, o sistema foi montado com uma sonda que utiliza o protocolo SDI-12, gravando suas medidas no cartão SD e enviando via rádio para um *notebook*. Essa aplicação, por ser em campo, contava com baterias e uma célula fotovoltaica, como mostrado na Figura 37.


Figura 37: Sistema de medidas montado para medições com o grupo de Agrobiologia da Embrapa, com um exemplo de painel de um Sistema Supervisório (a). Para a operação em campo, uma célula fotovoltaica é utilizada para carregar as baterias do sistema (b).

As medições foram todas realizadas dentro de um mesmo canal de vinhaça da Usina de cana de açúcar mostrado na Figura 38a, em três pontos diferentes. As medidas foram feitas com a imersão de uma sonda multiparamétrica no canal de vinhaça citado e adquirido pelo registrador do sistema, como mostrado na Figura 38b.

O primeiro ponto escolhido foi a 200 metros da saída da vinhaça. Neste local, com uma temperatura ambiente de 22,4°C, a vinhaça estava a 58,9°C. Nesta temperatura - acima de 50°C - a sonda utilizada não consegue realizar medições confiáveis e não se recomenda submetê-la a estas condições pelo risco de danos permanentes [13]. Por isso, só se conseguiu realizar poucas medidas de potencial de oxiredução, ainda com confiabilidade duvidosa.


Figura 38: Canal de vinhaça utilizado para as medições (a) e imersão da sonda no canal (b).

No segundo ponto (a 1,8 km da saída da vinhaça) a temperatura da vinhaça já estava abaixo do limite citado anteriormente (a 42,5°C), com temperatura ambiente de 27,3°C. Neste ponto foi possível realizar a medição de todos os parâmetros da sonda.

O terceiro e último ponto então foi a 3,5 km da saída da vinhaça. Neste ponto, a temperatura da vinhaça já era de 34,3°C e a temperatura ambiente se aproximava dos 30°C.

Como dito, no primeiro ponto testado, não foi possível a realização das medições da maioria dos parâmetros devido à alta temperatura da vinhaça no local, que excedia as condições de operação do sensor utilizado, tendo como resultado somente o potencial de oxirredução. Nesse contexto, o gráfico da Figura 39a mostra a variação da temperatura entre os pontos 2 e 3. Nessas medidas são observadas as temperaturas decrescendo em relação à distância da medição para a fonte da vinhaça, sendo o ponto 3, onde a vinhaça já está mais fria por causa da maior distância percorrida no canal.

As medidas de pH, condutividade específica e oxigênio dissolvido não sofreram mudanças significativas entre um ponto e outro, lembrando que, no primeiro ponto, não foi possível realizar as medidas, pois a temperatura do canal de vinhaça excedia a escala do sensor utilizado.

Já em relação ao potencial de oxirredução, o gráfico da Figura 39b mostra uma mudança nos valores medidos nos dois pontos.

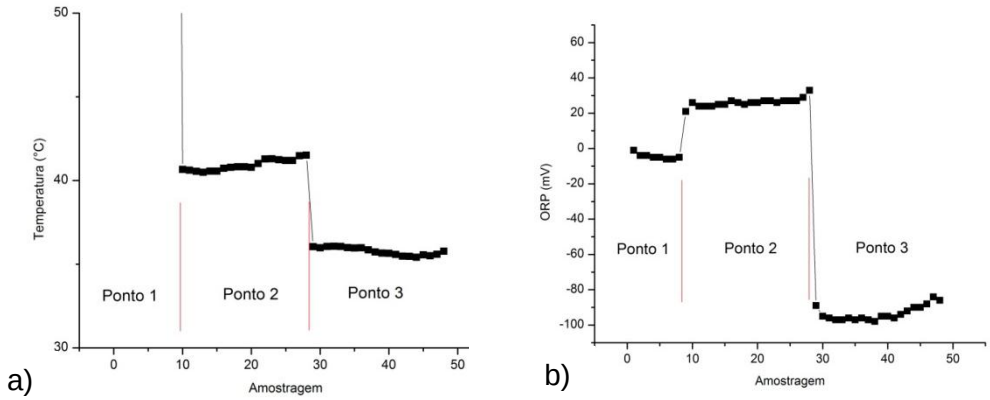



Figura 39: Variação da temperatura (a) e do potencial de oxirredução (b) nos pontos 2 e 3 do canal de vinhaça.

Os resultados obtidos com essas medidas serão utilizados pelo grupo de Agrobiologia da Embrapa, valores estes comparados com os níveis de gases emitidos para cada ponto.

Os valores estatísticos listados no Anexo A, principalmente os relacionados ao ORP, que mostrou diferenças significativas nos pontos testados, podem ajudar a esclarecer algumas hipóteses levantadas pelo grupo de pesquisa de Agrobiologia da Embrapa em seus estudos sobre a formação dos gases de efeito estufa, os GEE.

Viu-se necessária uma análise comparativa com os resultados obtidos na análise dos gases coletados nesses dias. Essa comparação poderá resultar em uma análise correlacionada de certos gases com a variação dos parâmetros medidos pelo sistema. Mais experimentos serão necessários para confirmar essa hipótese, visto que ocorreram erros de saturação na sonda, causadas pelas altas temperaturas do primeiro ponto medido (o mais próximo à saída da vinhaça). Além disso, poderia ser realizada uma medida adicional, contendo somente um ponto do canal, porém com uma maior excursão temporal, analisando então como esses parâmetros variam ao longo do dia.

## 4. CONCLUSÃO

Este projeto visa contribuir no desenvolvimento de dispositivos para experimentos de monitoramento ambiental. Acredita-se que, com um sistema suficientemente flexível, robusto e portátil em termos de montagem do experimento, seja possível otimizar o processo de medidas em um experimento como este, fazendo o usuário do sistema concentrar seus esforços na elaboração das metodologias e análise dos resultados específicos de sua pesquisa.

Circuitos eletrônicos e programação, integrantes de um sistema de monitoramento ambiental foram desenvolvidos para colaborar neste desenvolvimento. Os dispositivos montados têm a preocupação de convergirem para um sistema configurável, do qual necessita de uma arquitetura e programação flexíveis.

Define-se então essa arquitetura, que inclui uma placa multiprocessada, módulos de compatibilização de protocolos de comunicação, e que lança mão de diversos tipos de comunicação para estabelecer conexões, tanto dentro da placa, quanto entre módulos, e ainda, com um Sistema Supervisório.

Através do Sistema Supervisório, é possível realizar algumas configurações para o funcionamento do Sistema Embarcado, além de estar preparado para configurar outros parâmetros que ainda precisam ser implementados. O software desenvolvido para o Sistema Supervisório também possibilita a aquisição dos dados vindos do Sistema Embarcado, assim como sua formatação, que possibilita o desenho de gráficos em tempo real dentro deste mesmo software, assim como a exportação dos valores medidos para programas de análise de dados e estatística.

Uma das principais vantagens do tipo de arquitetura proposta nesse trabalho é a versatilidade na forma de armazenamento e comunicação dos dados. A memória não volátil na forma de um cartão SD proporciona ao sistema uma redundância nos dados adquiridos, além de permitir o armazenamento das configurações. Essas

características possibilitam a operação em campo, sem necessidade de equipamentos adicionais além de preparar o sistema para possíveis aplicações para fins de fiscalização.

O Sistema Embarcado pode também se comunicar de diversas formas com o Sistema Supervisório, como RS-232, WiFi e Ethernet. Dessa forma, é possível a realização de experimentos que necessitem ter uma estação remota, que se comunicará a distância com o Sistema Supervisório.

Alguns testes em laboratório foram realizados para a verificação das funções criadas para o sistema, gerando medidas de temperatura initerruptamente durante uma semana. Foi possível então analisar algumas características do dispositivo em atuação, como o seu consumo e capacidade de armazenamento.

A aplicabilidade deste tipo de sistema de monitoramento é estudada pelas medidas realizadas nos canais de vinhaça de cana de açúcar. Uma versão do sistema foi empregada para conhecer a demanda experimental dos pesquisadores da área agrícola (EMBRAPA), registrando em tempo real diversos parâmetros físico-químicos para futura correlação com a emissão de GEE. Estas medidas também auxiliaram em testes de algumas partes do sistema que já estavam funcionais, assim como também auxiliaram na detecção de alguns pontos fracos do sistema corrigidos para outras versões, como por exemplo a configuração de tempos de medida e armazenamento.

O trabalho contribui então para a criação de uma arquitetura que possibilitará a inserção de módulos de medida, adaptando o sistema para diferentes abordagens experimentais, tanto no que diz respeito à forma de aquisição dos dados gerados quanto à comunicação do sistema embarcado com o Sistema Supervisório.

Com todas as funções e módulos corretamente implementados, será concebido um sistema capaz de ser remotamente operável e configurável,

contribuindo para atender a demanda para monitoramento ambiental explicitada neste trabalho. A flexibilidade proposta nesta arquitetura permitirá ao experimentador fazer ajustes em sua metodologia de forma a convergir para um experimento otimizado, utilizando os sensores corretos e medindo os parâmetros que mais se adequem à sua pesquisa. Com a disponibilidade de um sensor digital cujo módulo de compatibilização tenha sido implementado, o sistema permitirá adicionar ao experimento quaisquer parâmetros que os pesquisadores julguem necessários à melhor compreensão do processo estudado.

## 5. BIBLIOGRAFIA


[1] A. S. Morris, "Principles of Measurement," in *Measurement and Instrumentation Principles*, Woburn, Butterworth-Heinemann, 2001, pp. 3-11.

[2] A. S. Morris, "Instrument Types and Performance Characteristics," in *Measurement and Instrumentation Principles*, Woburn, Butterworth-Heneimann, 2001, pp. 12-31.

[3] R. Kamal, "Introduction to Embedded Systems," in *Embedded Systems: Architeture, Programming and Design*, New Delhi, McGraw-Hill, 2008, pp. 1-61.

[4] P. Horowits and W. Hill, "Microprocessors," in *The Art of Eletronics*, Cambridge, Cambridge University Press, 1989, pp. 743-826.

[5] P. Horowits and W. Hill, "Low Power Design," in *The Art of Electronics*, Cambridge, Cambridge University Press, 1989, pp. 917-986.

[6] S. R. Ball, "Hardware Design," in *Embedded Microprocessor Systems*, Woburn, Butterworth-Heinemann, 2000, pp. 27-88.

[7] J. D. Day and H. Zimmermann, "The OSI Reference Model," *Proceedings of IEEE,* vol. 71, pp. 1334-1340, 1983.

[8] ISO, *ISO 7498:Basic Reference Model for Open Systems Interconnection,* ISO, 1983.

[9] H. Zimmermann, "OSI Reference Model - The ISO Model of Architeture for Open System Interconnection," *IEEE Transactions on Communications,* vol. 28, pp. 425-432, 1980.

[10] A. S. Morris, "Instrumentation/Computer Networks," in *Measurement and Instrumentation Principles*, Woburn, Butterworth-Heneimann, 2001, pp. 187-199.

[11] S. R. Ball, "Multiprocessor Systems," in *Embedded Microprocessor Systems: Real World Design*, Woburn, Butterworth-Heinemann, 2000, pp. 167-196.

[12] NXP Semiconductors, *I2C-bus specification and user manual, Rev. 4,* Eindhoven, 2012.

[13] Hydrolab, *Quanta Water Monitoring System,* Loveland, 2002.

[14] S. Ambiental, "Conheça a Squitter Ambiental," Squitter® Ambiental, 2010. [Online]. Available: http://www.squitter.com.br/portugues/conheca-a-squitter-ambiental/. [Accessed 15 03 2013].

[15] ATMEL, *8 bit Microcontroler with 128K Bytes In-System Programable Flash,* San Jose, 2009.

[16] Texas Instruments, *3V to 5.5V Multichannel RS-232 Line Driver/Receiver with+-15kV ESD Protection,* Dallas: Texas Instruments, 2004.

[17] P. Horowits and W. Hill, "Microcomputers," in *The Art of Electronics*, Cambridge, Cambridge University Press, 1989, pp. 673-741.

[18] SD Group, *SD Specifications Part 1: Physical Layer Simplified Specification,* San Ramon: SD Card Association, 2010.

[19] IEEE Standards Association, *IEEE Std 802.11™-2012,* New York: IEEE, 2012.

[20] Telecontrolli, *RXQ2 GFSK Multichannel Radio Transceiver Datasheet,* Casoria: Telecontrolli, 2012.

[21] Freewave, *Free Wave Spread Spectrum Wireless Data Transceiver,* Boulder: Freewave, 2000.

[22] Roving Networks, *RN-171 802.11 b/g Wireless LAN Module,* Los Gatos: Roving Networks, 2012.

[23] A. S. Morris, "Display, Recording and Presentation of Measurement Data," in *Measurement and Instrumentation Principles*, Woburn, Butterworth-Heinemann, 2001, pp. 200-223.

[24] S. Lai, "Non-volatile memory technologies: The quest for ever lower cost," in *Electron Devices Meeting*, Saratoga, 2008.

[25] R. N. Handcock, D. L. Swain, G. J. Bishop-Hurley, K. P. Patison, T. Wark, P. Valencia, P. Corke and C. J. O'Neill, "Monitoring Animal Behaviour and Environmental Interactions Using Wireless Sensor Networks, GPS Collars and Satellite Remote Sensing," *Sensors,* vol. 9, no. 5, pp. 3586-3603, 2009.

[26] Y. Liu, Y. Chen, Y. Zhang and J. Tong, "Research Based on Real Time Monitoring System of Digitized Agricultural Water Supply," in *The 6th International Conference on Computer Science & Education*, Singapore, 2011.

[27] M. Ji-hua, "A Global Crop Growth Monitoring System Based on Remote Sensing," in *IEEE International Conference on Geoscience and Remote Sensing Symposium* , Beijing, 2006.

[28] A. Gil, D. Belver, P. Cabanelas, E. Castro, J. Díaz, J. A. Garzón, D. Gonzalez-Diaz, B. W. Kolb, M. Traxler, R. Trebacz and Z. P, "Control and Monitoring System for the HADES RPC Detector," in *16th IEEE-NPSS Real Time Conference*, Valencia, 2009.

[29] X. Zhang and B. Chang, "Research of Temperature and Humidity Monitoring System Based on WSN and Fuzzy Control," in *International Conference on Electronics and Optoelectronics*, Huai'an, 2011.

[30] W. Xiaohong and L. Liu, "The Design of Remote Medical Monitoring System Based on Sensors and GPRS," in *International Forum on Information Technology and Applications*, Jinan, 2009.

[31] J. Xiao, P. Liu, L. Jiao, H. Zhu and Y. Du, "Design of PV Power Station Remote Monitoring System Data Acquisition Device," in *International Conference on Advanced Mechatronic Systems*, Zhengzhou, 2011.

[32] F. Shen, C. Wei, Z. Cao, C. Zhou, D. Xu and W. Zhang, "Water Quality Monitoring System Based on Robotic Dolphin," in *World Congress on Intelligent Control and Automation*, Taipei, 2011.

[33] A. Anvari, J. D. Reyes, E. Esmaeilzadeh, A. Jarvandi and N. Langley, "Designing an Automated Water Quality Monitoring System for West and Rhode Rivers," in *IEEE Systems and Information Engineering Design Symposium*, Charlottesville, 2009.

[34] C. C. Lisboa, K. BUTTERBACH-BAHL, M. Mauder and R. Kiese, "Bioethanol production from sugarcane and emissions of greenhouse gases – known and unknowns," *GCB Bioenergy,* pp. 277-292, Agosto 2011.

[35] L. Soares, B. Alves, S. Urquiaga and R. Boddey, *Mitigação das emissões de gases efeito estufa pelo uso de etanol da cana-de-açúcar produzido no Brasil,* Seropédica: Embrapa, 2009.

[36] E. Kebreab, K. Clark, C. Wagner-Riddle and J. France, "Methane and nitrous oxide

emissions from Canadian animal agriculture: A review," *Canadian Journal of Animal Science,* pp. 135-157, 2006.

## ANEXO A: VALORES ESTATÍSTICOS – VINHAÇA DE CANA DE AÇÚCAR

Valores estatísticos para o ponto 2.

| | N total | Mean | SE of mean |
|---|---|---|---|
| Temperatura | 18 | 40.88778 | 0.07427 |
| pH | 18 | 4.45722 | 0.00211 |
| Condutividade Especifica | 18 | 20.45056 | 0.1176 |
| Salinidade | 18 | 12.41111 | 0.07724 |
| DO % | 18 | 20.05556 | 2.95217 |
| DO | 18 | 0.91611 | 0.14563 |
| ORP | 18 | 26 | 0.31311 |

Valores estatísticos para o ponto 3.

| | N total | Mean | SE of mean |
|---|---|---|---|
| Temperatura | 20 | 35.7665 | 0.05176 |
| pH | 20 | 4.498 | 0.00258 |
| Condutividade Especifica | 20 | 20.845 | 0.0578 |
| Salinidade | 20 | 12.605 | 0.03782 |
| DO % | 20 | 11.93 | 2.28303 |
| DO | 20 | 0.633 | 0.12019 |
| ORP | 20 | -93.4 | 0.92452 |

## ANEXO B: TABELA DE ESPECIFICAÇÕES TÉCNICAS

| Especificações Técnicas | |
|---|---|
| Tensão de operação | 12 V |
| **Consumo** | |
| Sem módulos acoplados | 20 mA |
| Sonda SDI-12 e comunicação via rádio (aplicação EMBRAPA) | 135 mA (medições)<br>720 mA (transmissão) |
| **Armazenamento** | |
| Cartão de memória 1 GB | > $2x10^6$ medidas |
| Cartão de memória 2 GB | > $4x10^6$ medidas |
| **Sensores** | |
| Máximo de sensores no barramento | 10 |
| Interfaces | RS-232<br>SDI-12 |
| **Comunicação RS-232 com sistema supervisório** | |
| *Baud Rate* | 9600 bps |
| *Bits de Dados* | 8 |
| *Flow control* | None |
| *Stop Bit* | 1 bit |

## ANEXO C: FIRMWARE MICROCONTROLADOR MESTRE

```
#define F_CPU 4000000

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "C:\Users\Brultra\Desktop\Programa caipora\AVR programas bruno\BIBLIOTECAS_BRUNO\fifobuffer.h"

#include "C:\Users\Brultra\Desktop\Programa caipora\AVR programas bruno\BIBLIOTECAS_BRUNO\USART0_BRUNO.h"

#include "C:\Users\Brultra\Desktop\Programa caipora\AVR programas bruno\BIBLIOTECAS_BRUNO\USART1_BRUNO.h"

#include "C:\Users\Brultra\Desktop\Programa caipora\AVR programas bruno\BIBLIOTECAS_BRUNO\SDCARD_BRUNO.h"

#define MODO_OPERACAO_DEFAULT 3

////  Modo 0 = sem SDCard e MC Mestre e MC leitura em estado de wait comando Central

////  Modo 1 = sem SDcard e MC Mestre e MC leitura em estado de comando de botão de aquisição

////  Modo 2 = sem SDcard e MC Mestre e MC leitura em estado de Aquisição e guarda medida na memoria e envia o bloco

////  Modo 3 = com SDCard e MC Mestre e MC leitura em estado de wait comando Central

////  Modo 4 = com SDcard e MC Mestre e MC leitura em estado de comando de botão de aquisição e guarda medida na memoria e envia

////  Modo 5 = com SDcard e MC Mestre e MC leitura em estado de Aquisição e guarda medida na memoria e envia o bloco

////  Modo 6 = Modo economico com SDcard e MC Mestre e MC leitura em estado de Aquisição e guarda medida na memoria e envia bloco economico


#define TEMPO_AQUISICAO_DEFAULT  13000          // POR ENQUANTO EM MILISEGUNDOS

#define TIMEOUT_DEFAULT  5                      // POR ENQUANTO 5 * TEMPO_AQUISICAO (range de 0 a 9)

#define SIZE_BLOCO_CONFIGURACAO 512     //TAMANHO DO BLOCO DE CONFIGURAÇÃO DOS PERIFÉRICOS

#define SIZE_BLOCO_LEITURA 512      //TAMANHO DO BLOCO A LER E GRAVAR NO SD/RADIO

#define SIZE_MAX_COMANDO_LEITURA 10

#define SIZE_MAX_FORMATO_LEITURA 20
```

```
#define TIME_WAIT_DEFAULT 7000       //Em ms, TEMPO DE ESPERA DE LEITURA DO PERIFÉRICO MAIS LENTO

#define TIME_WAIT_INIT_DEFAULT 13000       //Em ms

#define N_MAX_SENSORES 15

#define N_MAX_PARAMETROS 15

////////////////////////////////////////////////////////////////////////////////////////////          config

        char modo_operacao;

        int tempo_aquisicao;

        int n_bloco_envio, bloco_envio_atual;

        char n_timeout;

        char n_dispositivos_i2c;

        char serial_port_en=0;

        char serial_port_config;

        char serial_port_n_param;

        int time_wait;

        int time_wait_init;

        unsigned long int endereco_atual_sd_card;

        char *serial_port_cmd_init; //configurar para n parametros

        char *serial_port_fmt_init;

        char *serial_port_cmd_leit[N_MAX_PARAMETROS];

        char *serial_port_fmt_leit[N_MAX_PARAMETROS];

    char *protocolo;   //transforma em matriz de tamanho n_dispositivos(1 protocolo para cada dispositivo) (criar regras para cada protocolo)

    int *config_protocolo;  //transforma em matriz de tamanho n_dispositivos (criar regras para cada protocolo)

    char *endereco_i2c;  //transforma em matriz de tamanho n_dispositivos (formato xxxxxxx0, pois ultimo bit = R+W)

    char **cod_sensor[3]; //matriz n_dispositivos x n_param_a_medir x 3 (exemplo: SD0(xxxxx), "SD0" é o mnemonico do cod_sensor)

    char **cmd_init;     //transforma em matriz de tamanho n_dispositivos x tamanho dos comandos de inicialização

    char **fmt_init;     //transforma em matriz de tamanho n_dispositivos x tamanho dos formatos de inicialização

    char *n_param_a_medir; //quantos parâmetros requerer do mesmo endereço (tamanho n_dispositivos)

    char **fmt_leitura;   //transforma em matriz de tamanho do somatório dos n_param_a_medir dos n_dispositivos

    char **cmd_leitura;  //transforma em matriz de tamanho do somatório dos n_param_a_medir dos n_dispositivos
```

```
	char *bloco_leitura,*sd_adr_ptr;//sd_adr_ptr = ponteiro para endereço do sd card
	char *dummyint[N_MAX_SENSORES+2];

//Função que envia as configurações para o Microcontrolador Leitura
//Parametros = void
//Retorna = void
void configura_mc_leitura()
{
	char *bloco_config;
	char buff_string1[80];
	int to_count=0;
	bloco_config=malloc(SIZE_BLOCO_CONFIGURACAO);
	USART0_Transmit('C');
	USART0_Clr_Rxbuff();
	_delay_ms(DELAY_USART0);
	while (USART0_Chars_In_Buff()==0)
	{
		_delay_ms(DELAY_USART0);
		to_count++;
		if (to_count*DELAY_USART0>10000)
		{
			sprintf(buff_string1,"Timeout Config MC Leitura1\r\n");
			USART1_Transmit_Block(buff_string1,strlen(buff_string1));
			break;
		}
	}
	to_count=0;
	if (USART0_Receive()=='c')
	{
		for (int i=0; i<n_dispositivos_i2c+2; i++)
		{
		Comando17_open(0,0,2*i,0);// Verifica parâmetros SDcard
		for (int j=0;j<SIZE_BLOCO_CONFIGURACAO;j++)
		{
```

```
                recebe_char_sdcard(bloco_config+j);
            }
            USART0_Transmit_Block(bloco_config,SIZE_BLOCO_CONFIGURACAO);
            USART0_Clr_Rxbuff();
            while (USART0_Chars_In_Buff()==0)
            {
                _delay_ms(DELAY_USART0);
                to_count++;
                if (to_count*DELAY_USART0>10000)
                {
                    sprintf(buff_string1,"Timeout Config MC Leitura2\r\n");
                    USART1_Transmit_Block(buff_string1,strlen(buff_string1));
                    break;
                }
            }
            _delay_ms(DELAY_USART0);
            if (USART0_Receive()!='c')break;
            }
        }
free(bloco_config);
USART0_Clr_Rxbuff();
}

//Função que inicializa as variáveis de acordo com os dados do SD Card
//Parametros = void
//Retorna = void
void leitura_configuracao_sdcard()
{
    char* bloco_lido;
    char *bufferitoa;
    modo_operacao = MODO_OPERACAO_DEFAULT;
    tempo_aquisicao = TEMPO_AQUISICAO_DEFAULT;
    n_bloco_envio = 0;                  // Nao tem bloco para enviar
    n_timeout =  TIMEOUT_DEFAULT;
```

```
	bloco_lido=malloc(512);

	bufferitoa=malloc(10);

	Comando17_open(0,0,0,0);// Verifica parâmetros SDcard

	for (int i=0;i<512;i++)

	{

		recebe_char_sdcard(bloco_lido+i);

	}

	if(sscanf(bloco_lido,"BLC(%u)MOD(%d)TAQ(%d)BEN(%d)TOU(%d)DIS(%d)SPE(%d)SPC(%d)S
PP(%d)TWI(%d)TWM(%d)",&endereco_atual_sd_card,&modo_operacao,&tempo_aquisicao,&n_bloco_en
vio,&n_timeout,&n_dispositivos_i2c,&serial_port_en,&serial_port_config,&serial_port_n_param,&time_wait
_init,&time_wait)<=0)

//se não tiver parâmetros, reformatar cartão e começar do endereço do bloco 1

	{

		n_dispositivos_i2c=0;

		Comando24_open(*(sd_adr_ptr+3),*(sd_adr_ptr+2),*(sd_adr_ptr+1),*(sd_adr_ptr+0));

		envia_char_sdcard('B');

		envia_char_sdcard('L');

		envia_char_sdcard('C');

		envia_char_sdcard('(');

		envia_char_sdcard('0');

		envia_char_sdcard('0');

		envia_char_sdcard('0');

		envia_char_sdcard('0');

		envia_char_sdcard('0');

		envia_char_sdcard('0');

		envia_char_sdcard('0');

		envia_char_sdcard('0');

		envia_char_sdcard('5');

		envia_char_sdcard('1');

		envia_char_sdcard('2');

		envia_char_sdcard(')');

		envia_char_sdcard('M');

		envia_char_sdcard('O');

		envia_char_sdcard('D');

		envia_char_sdcard('(');

		envia_char_sdcard(modo_operacao+48);
```

```
                envia_char_sdcard(')');


                for (int i=22; i<512; i++)
                {
                        envia_char_sdcard('?');
                }
endereco_atual_sd_card=512*(N_MAX_SENSORES+3);//primeiro bloco depois do bloco de configuração
        }
        free (bloco_lido);
        free (bufferitoa);
}


//Função recebe a tabela de configuração através do Sistema Supervisório e guarda no SD Card
//Parametros = void
//Retorna = void
void leitura_configuracao_central()
{
        char dummystr[SIZE_BLOCO_CONFIGURACAO];
        USART1_Transmit('c');//USART pergunta à central os itens da tabela
        while (USART1_Chars_In_Buff()==0)
        {
                _delay_ms(DELAY_USART1*5);
        }
        USART1_Receive_Block(&dummystr,SIZE_BLOCO_CONFIGURACAO); //recebe tabela de
configuração pela USART1 (central)
        Comando24_open(0,0,0,0);
        for (int i=0; i<SIZE_BLOCO_CONFIGURACAO; i++)
        {
                envia_char_sdcard(dummystr[i]);
        }
        dummyint[0]=strtok(dummystr,")");
        dummyint[1]=strtok(NULL,")");
        dummyint[2]=strtok(NULL,")");
        dummyint[3]=strtok(NULL,")");
```

```
dummyint[4]=strtok(NULL,")");
dummyint[5]=strtok(NULL,")");
dummyint[6]=strtok(NULL,")");
dummyint[7]=strtok(NULL,")");
dummyint[8]=strtok(NULL,")");
dummyint[9]=strtok(NULL,")");
dummyint[10]=strtok(NULL,")");
sscanf(dummyint[1],"MOD(%d", &modo_operacao);
sscanf(dummyint[2],"TAQ(%d", &tempo_aquisicao);
sscanf(dummyint[3],"BEN(%d", &n_bloco_envio);
sscanf(dummyint[4],"TOU(%d", &n_timeout);
sscanf(dummyint[5],"DIS(%d", &n_dispositivos_i2c);
sscanf(dummyint[6],"SPE(%d", &serial_port_en);
sscanf(dummyint[7],"SPC(%d", &serial_port_config);
sscanf(dummyint[8],"SPP(%d",&serial_port_n_param);
sscanf(dummyint[9],"TWI(%d",&time_wait_init);
sscanf(dummyint[10],"TWM(%d",&time_wait);
USART1_Transmit('c');//USART pergunta à central os itens do bloco 2
USART1_Clr_Rxbuff();
while (USART1_Chars_In_Buff()==0)
{
        _delay_ms(DELAY_USART1*5);
}
USART1_Receive_Block(&dummystr,SIZE_BLOCO_CONFIGURACAO); //recebe tabela de configuração pela USART1 (central)
Comando24_open(0,0,2,0);
        for (int i=0; i<SIZE_BLOCO_CONFIGURACAO; i++)
{
        envia_char_sdcard(dummystr[i]);
}
for (int i=0; i<n_dispositivos_i2c; i++)
{
        USART1_Transmit('c');//USART pergunta à central os itens do bloco 2
        USART1_Clr_Rxbuff();
```

```
                while (USART1_Chars_In_Buff()==0)
                {
                        _delay_ms(DELAY_USART1*5);
                }
                USART1_Receive_Block(&dummystr,SIZE_BLOCO_CONFIGURACAO); //recebe tabela de configuração pela USART1 (central)
                Comando24_open(0,0,2*(i+2),0);
                for (int j=0; j<SIZE_BLOCO_CONFIGURACAO; j++)
                {
                        envia_char_sdcard(dummystr[j]);
                }
        }
}

//Função que envia as configurações como informação para o Sistema Supervisório
//Parametros = void
//Retorna = void
void mostra_config()
{
        char *bloco_lido;
        char buff_string1[80];
        bloco_lido=malloc(512);
        Comando17_open(0,0,0,0);
        for(int i=0; i<512; i++) recebe_char_sdcard(bloco_lido+i);
        USART1_Transmit_Block(bloco_lido,512);
        Comando17_open(0,0,2,0);
        for(int i=0; i<512; i++) recebe_char_sdcard(bloco_lido+i);
        USART1_Transmit_Block(bloco_lido,512);

        for (int j=0; j<n_dispositivos_i2c; j++)
        {
                Comando17_open(0,0,2*(j+2),0);
                for(int i=0; i<512; i++) recebe_char_sdcard(bloco_lido+i);
                USART1_Transmit_Block(bloco_lido,512);
```

```
	}
	free (bloco_lido);
}


//Função que envia todos os dados do SD Card para o Sistema Supervisório
//Parametros = void
//Retorna = void
void descarrega_sd_card_serial()
{
	char *bloco_lido;
	unsigned long int j;
	j=endereco_atual_sd_card;
	bloco_lido=malloc(512);
	for(endereco_atual_sd_card=0;endereco_atual_sd_card<j;endereco_atual_sd_card=endereco_at
ual_sd_card+512)
	{
		Comando17_open(*(sd_adr_ptr+3),*(sd_adr_ptr+2),*(sd_adr_ptr+1),*(sd_adr_ptr+0));


		for (int i=0;i<512;i++)
		{
			recebe_char_sdcard(bloco_lido+i);
		}
		USART1_Transmit_Block(bloco_lido,512);
		_delay_ms(DELAY_USART1);
	}
	free(bloco_lido);
	endereco_atual_sd_card=j;
}


//Função que grava um bloco no cartão SD
//Parametros = void
//Retorna = void
void armazena_bloco_sd()
{
```

```
char *ul_endereco_sd;
char *bufferitoa;
int lenght_ul_endereco_sd=0;
ul_endereco_sd=malloc(10);
bufferitoa=malloc(10);

Comando24_open(*(sd_adr_ptr+3),*(sd_adr_ptr+2),*(sd_adr_ptr+1),*(sd_adr_ptr+0));
for (int i=0; i<512; i++)
{
        envia_char_sdcard(*(bloco_leitura+i));
}
endereco_atual_sd_card+=512;
lenght_ul_endereco_sd=strlen(ultoa(endereco_atual_sd_card,ul_endereco_sd,10));
Comando24_open(0,0,0,0);//rotina para atualizar o endereço no bloco de configuração
envia_char_sdcard('B');
envia_char_sdcard('L');
envia_char_sdcard('C');
envia_char_sdcard('(');
for (int i=0; i<(10-lenght_ul_endereco_sd); i++)
{
        envia_char_sdcard('0');
}
for (int i=0; i<lenght_ul_endereco_sd; i++)
{
        envia_char_sdcard(*(ul_endereco_sd+i));
}
envia_char_sdcard(')');

        envia_char_sdcard('M');
        envia_char_sdcard('O');
        envia_char_sdcard('D');
        envia_char_sdcard('(');
        envia_char_sdcard(modo_operacao+48);
        envia_char_sdcard(')');
```

```
envia_char_sdcard('T');
envia_char_sdcard('A');
envia_char_sdcard('Q');
envia_char_sdcard('(');
itoa(tempo_aquisicao,bufferitoa,10);
for (int i=0; i<10-strlen(bufferitoa); i++)
{
        envia_char_sdcard('0');
}
for (int i=0; i<strlen(bufferitoa); i++)
{
        envia_char_sdcard(*(bufferitoa+i));
}
envia_char_sdcard(')');
envia_char_sdcard('B');
envia_char_sdcard('E');
envia_char_sdcard('N');
envia_char_sdcard('(');
itoa(n_bloco_envio,bufferitoa,10);
for (int i=0; i<10-strlen(bufferitoa); i++)
{
        envia_char_sdcard('0');
}
for (int i=0; i<strlen(bufferitoa); i++)
{
        envia_char_sdcard(*(bufferitoa+i));
}
envia_char_sdcard(')');
envia_char_sdcard('T');
envia_char_sdcard('O');
envia_char_sdcard('U');
envia_char_sdcard('(');
envia_char_sdcard(n_timeout+48);
envia_char_sdcard(')');
```

```
envia_char_sdcard('D');
envia_char_sdcard('I');
envia_char_sdcard('S');
envia_char_sdcard('(');
itoa(n_dispositivos_i2c,bufferitoa,10);
for (int i=0; i<3-strlen(bufferitoa); i++)
{
        envia_char_sdcard('0');
}
for (int i=0; i<strlen(bufferitoa); i++)
{
        envia_char_sdcard(*(bufferitoa+i));
}
envia_char_sdcard(')');
envia_char_sdcard('S');
envia_char_sdcard('P');
envia_char_sdcard('E');
envia_char_sdcard('(');
envia_char_sdcard(serial_port_en+48);
envia_char_sdcard(')');
envia_char_sdcard('S');
envia_char_sdcard('P');
envia_char_sdcard('C');
envia_char_sdcard('(');
itoa(serial_port_config,bufferitoa,10);
for (int i=0; i<3-strlen(bufferitoa); i++)
{
        envia_char_sdcard('0');
}
for (int i=0; i<strlen(bufferitoa); i++)
{
        envia_char_sdcard(*(bufferitoa+i));
}
envia_char_sdcard(')');
```

```
envia_char_sdcard('S');
envia_char_sdcard('P');
envia_char_sdcard('P');
envia_char_sdcard('(');
itoa(serial_port_n_param,bufferitoa,10);
for (int i=0; i<3-strlen(bufferitoa); i++)
{
        envia_char_sdcard('0');
}
for (int i=0; i<strlen(bufferitoa); i++)
{
        envia_char_sdcard(*(bufferitoa+i));
}
envia_char_sdcard(')');
envia_char_sdcard('T');
envia_char_sdcard('W');
envia_char_sdcard('I');
envia_char_sdcard('(');
itoa(time_wait_init,bufferitoa,10);
for (int i=0; i<10-strlen(bufferitoa); i++)
{
        envia_char_sdcard('0');
}
for (int i=0; i<strlen(bufferitoa); i++)
{
        envia_char_sdcard(*(bufferitoa+i));
}
envia_char_sdcard(')');
envia_char_sdcard('T');
envia_char_sdcard('W');
envia_char_sdcard('M');
envia_char_sdcard('(');
itoa(time_wait,bufferitoa,10);
for (int i=0; i<10-strlen(bufferitoa); i++)
```

```
                {
                        envia_char_sdcard('0');
                }
                for (int i=0; i<strlen(bufferitoa); i++)
                {
                        envia_char_sdcard(*(bufferitoa+i));
                }

                envia_char_sdcard(')');

        for (int i=117; i<512; i++)
        {
                envia_char_sdcard('?');
        }
        free(ul_endereco_sd);
        free(bufferitoa);
}

//Função que verifica se o modo de operação mudou durante a última medida
//Parametros = void
//Retorna = void
void verifica_modo_operacao()
{
        char buff_string[STRING_MAXIMO_USART0];
        if (USART1_Chars_In_Buff()>0)
                {
                        switch (USART1_Receive()) //check_comando_central();
                        {
                case  'C':      //   Comando de envio de configuração pela Central para o MC Mestre
                                        leitura_configuracao_central();
                                        USART1_Clr_Rxbuff();
                                break;
                case  'c':              //   Comando de questionamento de configuração do MC Mestre
                                        mostra_config();
```

```
                                        break;


                        case 'Y':
                                                descarrega_sd_card_serial();
                                        break;
                                }
                        }
}


//Função que envia o bloco de medidas para o Sistema Supervisório
//Parametros = void
//Retorna = void
void envia_bloco_serial()
{
                        USART1_Transmit_Block(bloco_leitura,SIZE_BLOCO_LEITURA);
}


//Função que, no modo econômico, verifica se chegou a hora de enviar os blocos para o Sistema
//Supervisório
//Parametros = void
//Retorna = void
void verifica_enviar_bloco()
{
        bloco_envio_atual++;
        if (bloco_envio_atual>=n_bloco_envio)
        {
                envia_bloco_serial();
        }
}


//Função que lê o bloco enviado pelo Microcontrolador Leitura
//Parametros = void
//Retorna = void
int le_bloco()
```

```
{
                        bloco_leitura=calloc(sizeof(char),SIZE_BLOCO_LEITURA);
                        USART0_Clr_Rxbuff();
                        _delay_ms(DELAY_USART0);
                        USART0_Transmit('M');
                        for (int i=0; i<time_wait;i++)
                        {
                                _delay_ms(1);
                        }
                        USART0_Transmit('t');
                        _delay_ms(1000);
                        USART0_Receive_Block(bloco_leitura,SIZE_BLOCO_LEITURA);
                        _delay_ms(DELAY_USART0);
                        if (modo_operacao>=3)
                        {
                                armazena_bloco_sd(); //envia bloco para sd card
                                if (modo_operacao==6)
                                {
                                        verifica_enviar_bloco();
                                        return 0;
                                }
                        }
                                envia_bloco_serial(); //falta tratar erro
                                free(bloco_leitura);
                                return 0;
}

//Função referente ao modo de operação 2, 5 ou 6 (modo contínuo e modo econômico)
//Parametros = void
//Retorna = void
void loop_remoto()
{
        int erro_leitura;
        erro_leitura=le_bloco();
```

```
	for (int i=0;i<(tempo_aquisicao-time_wait);i++)
	{
		_delay_ms(1);
	}
	verifica_modo_operacao();
	USART1_Clr_Rxbuff();

}

//Função referente ao modo de operação 0 ou 3 (modo central)
//Parametros = void
//Retorna = void
void loop_central()
{
	char buff_string[80];
	int erro_leitura;

	if (USART1_Chars_In_Buff()>0)
		{
		switch (USART1_Receive()) //check_comando_central();
			{
		case  'C':	//  Comando de envio de configuração pela Central para o MC Mestre
					leitura_configuracao_central();
					USART1_Clr_Rxbuff();
				break;
		case  'c':		//  Comando de questionamento de configuração do MC Mestre
					mostra_config();
				break;

		case 'L':
		case 'l':
					erro_leitura=le_bloco();
					USART0_Clr_Rxbuff();
				break;
```

```
                case 'Y':
                                descarrega_sd_card_serial();
                                USART1_Clr_Rxbuff();
                        break;

                }
        }
}

//Função referente ao modo de operação 1 ou 4 (modo local)
//Parametros = void
//Retorna = void
void loop_botao()
{
        int erro_leitura;

                if (USART0_Chars_In_Buff()==SIZE_BLOCO_LEITURA)
                {
                        erro_leitura=le_bloco();
                }
                verifica_modo_operacao();
                USART1_Clr_Rxbuff();
}

//Função que escolhe qual modo seguir
//Parametros = void
//Retorna = void
void tipo_loop()
{
        switch (modo_operacao)
        {
        case 0:
                loop_central();
```

```
                break;
        case 1:
                loop_botao();
                break;
        case 2:
                loop_remoto();
                break;
        case 3:
                loop_central();
                break;
        case 4:
                loop_botao();
                break;
        case 5:
                loop_remoto();
                break;
        case 6:
                loop_remoto();
                break;
        }
}
int main(void)
{
char buff_string1[80];
char data;
int erro_sd;
CLKPR= 0x80;
CLKPR= 0x01;//define clock 4MHz
USART0_Init(26);//initialize usart and define baud rate 9600
USART1_Init(26);
SPI_MasterInit();
sd_adr_ptr=&endereco_atual_sd_card;
sprintf(buff_string1,"Iniciando SD\r\n");
USART1_Transmit_Block(buff_string1,strlen(buff_string1));
```

```
inicializa_sdcard();
sprintf(buff_string1,"Espere inicializacao dos sensores\r\n");
USART1_Transmit_Block(buff_string1,strlen(buff_string1));
 leitura_configuracao_sdcard();
 configura_mc_leitura();
 for (int i=0;i<time_wait_init; i++)
 {
              _delay_ms(1);
 }
sprintf(buff_string1,"Pronto para aquisição\r\n");
USART1_Transmit_Block(buff_string1,strlen(buff_string1));
USART1_Clr_Rxbuff();
USART0_Clr_Rxbuff();
              while (1)
              {
                            tipo_loop();
              }
}
```

## ANEXO D: FIRMWARE MICROCONTROLADOR LEITURA

```
#define F_CPU 4000000

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "C:\Users\Brultra\Desktop\Programa caipora\AVR programas bruno\BIBLIOTECAS_BRUNO\fifobuffer.h"

#include "C:\Users\Brultra\Desktop\Programa caipora\AVR programas bruno\BIBLIOTECAS_BRUNO\USART0_BRUNO.h"

#include "C:\Users\Brultra\Desktop\Programa caipora\AVR programas bruno\BIBLIOTECAS_BRUNO\USART1_BRUNO.h"

#include "C:\Users\Brultra\Desktop\Programa caipora\AVR programas bruno\BIBLIOTECAS_BRUNO\TWI_BRUNO.h"


#define SIZE_BLOCO_CONFIGURACAO 512 //BYTES NA TABELA DE CONFIGURAÇÃO

#define TEMPO_AQUISICAO_DEFAULT  30 // POR ENQUANTO EM SEGUNDOS

#define TIMEOUT_DEFAULT  5 // POR ENQUANTO 5 * TEMPO_AQUISICAO

#define SIZE_BLOCO_LEITURA 512 //BYTES NO BLOCO DE LEITURA

#define MODO_OPERACAO_DEFAULT 0

#define SIZE_MEDIDA_AD_HC08 20

#define SIZE_MEDIDA_NOMAD 31

#define SIZE_MEDIDA_SONDA 60

#define N_MAX_PARAMETROS 15

#define N_MAX_SENSORES 15

#define N_BYTES_CMD 40


char modo_operacao;

int tempo_aquisicao;

int n_bloco_envio, bloco_envio_atual;

char n_timeout;

char n_dispositivos_i2c;

char serial_port_en=0;
```

```
char serial_port_config;

char serial_port_n_param;

int time_wait;

int time_wait_init;


char *serial_port_cmd_init; //configurar para n parametros

char *serial_port_fmt_init;

char serial_port_cmd_leit[N_MAX_PARAMETROS][N_BYTES_CMD];

int serial_port_fmt_leit[N_MAX_PARAMETROS];


char protocolo[N_MAX_SENSORES];  //transforma em matriz de tamanho n_dispositivos(1 protocolo para cada dispositivo) (criar regras para cada protocolo)

int config_protocolo[N_MAX_SENSORES];  //transforma em matriz de tamanho n_dispositivos (criar regras para cada protocolo)

char endereco_i2c[N_MAX_SENSORES];  //transforma em matriz de tamanho n_dispositivos (formato xxxxxxx0, pois ultimo bit = R+W)

char cod_sensor[N_MAX_SENSORES][N_MAX_PARAMETROS][3]; //matriz n_dispositivos x n_param_a_medir x 3 (exemplo: SD0(xxxxx), "SD0" é o mnemonico do cod_sensor)

char *cmd_init[N_MAX_SENSORES];  //transforma em matriz de tamanho n_dispositivos x tamanho dos comandos de inicialização

char *fmt_init[N_MAX_SENSORES];  //transforma em matriz de tamanho n_dispositivos x tamanho dos formatos de inicialização

char n_param_a_medir[N_MAX_SENSORES]; //quantos parâmetros requerer do mesmo endereço (tamanho n_dispositivos)

int fmt_leitura[N_MAX_SENSORES][N_MAX_PARAMETROS];  //transforma em matriz de tamanho do somatório dos n_param_a_medir dos n_dispositivos

char cmd_leitura[N_MAX_SENSORES][N_MAX_PARAMETROS][N_BYTES_CMD];  //transforma em matriz de tamanho do somatório dos n_param_a_medir dos n_dispositivos


char *bloco_leitura;

char *serial_port_measure[N_MAX_PARAMETROS];

int n_medida=0;

int n_bloco=0;

int size_i2c[N_MAX_SENSORES];

int size_total_i2c=0;


//Função que recebe as configurações do Microcontrolador Mestre

//Parametros = void
```

```
//Retorna = void
void leitura_configuracao()
{

	char *dummyint[N_MAX_PARAMETROS+2];
	char *string_configuracao;
	char *printconfig;
	int serial_port_size=0;
	char buff_string[STRING_MAXIMO_USART1]="lixo0";

	printconfig=malloc(80);
	string_configuracao=malloc(SIZE_BLOCO_CONFIGURACAO);
	USART0_Clr_Rxbuff();
	while (USART0_Chars_In_Buff()==0)_delay_ms(DELAY_USART0);
	_delay_ms(DELAY_USART0);
	if (USART0_Receive()=='C')
	{
		_delay_ms(DELAY_USART0);
		USART0_Clr_Rxbuff();
		USART0_Transmit('c');//USART pergunta à central os itens da tabela

	}

	_delay_ms(DELAY_USART0);
	while (USART0_Chars_In_Buff()==0)_delay_ms(DELAY_USART0*5);
	USART0_Receive_Block(string_configuracao,SIZE_BLOCO_CONFIGURACAO); //recebe tabela
//de configuração pela USART0 (mc_mestre)
	USART0_Clr_Rxbuff();
	_delay_ms(DELAY_USART0);
	dummyint[0]=strtok(string_configuracao,")");
	dummyint[1]=strtok(NULL,")");
	dummyint[2]=strtok(NULL,")");
	dummyint[3]=strtok(NULL,")");
	dummyint[4]=strtok(NULL,")");
```

```
dummyint[5]=strtok(NULL,")");
dummyint[6]=strtok(NULL,")");
dummyint[7]=strtok(NULL,")");
dummyint[8]=strtok(NULL,")");
dummyint[9]=strtok(NULL,")");
dummyint[10]=strtok(NULL,")");

sscanf(dummyint[1],"MOD(%d", &modo_operacao);
sscanf(dummyint[2],"TAQ(%d", &tempo_aquisicao);
sscanf(dummyint[3],"BEN(%d", &n_bloco_envio);
sscanf(dummyint[4],"TOU(%d", &n_timeout);
sscanf(dummyint[5],"DIS(%d", &n_dispositivos_i2c);
sscanf(dummyint[6],"SPE(%d", &serial_port_en);
sscanf(dummyint[7],"SPC(%d", &serial_port_config);
sscanf(dummyint[8],"SPP(%d",&serial_port_n_param);
sscanf(dummyint[9],"TWI(%d",&time_wait_init);
sscanf(dummyint[10],"TWM(%d",&time_wait);
USART0_Transmit('c');//Requisita próximo bloco de config
_delay_ms(DELAY_USART0);
while (USART0_Chars_In_Buff()==0)_delay_ms(DELAY_USART0*5);
USART0_Receive_Block(string_configuracao,SIZE_BLOCO_CONFIGURACAO);  //recebe  bloco
//2 da tabela de configuração pela USART0 (mc_mestre)
USART0_Clr_Rxbuff();
dummyint[0]=strtok(string_configuracao,")");
dummyint[1]=strtok(NULL,")");
for (int i=0; i<serial_port_n_param; i++)
{
        dummyint[2*i+2]=strtok(NULL,")");
        dummyint[2*i+3]=strtok(NULL,")");
}
sscanf(dummyint[0],"SCI(%s",&serial_port_cmd_init);
sscanf(dummyint[1],"SFI(%s",&serial_port_fmt_init);
for (int i=0; i<serial_port_n_param; i++)
{
```

```
                for (int j=0; j<strlen(dummyint[2*j+2]+4); j++)
                {
                        serial_port_cmd_leit[i][j]=dummyint[2*i+2][j+4];
                }

                serial_port_fmt_leit[i]=atoi(dummyint[2*i+3]+4);
        }
        ////////////////////////////////////////////////////////////////configura I2C
        for (int j=0; j<n_dispositivos_i2c; j++)
        {
                USART0_Transmit('c');//Requisita próximo bloco de config
                _delay_ms(DELAY_USART0);
                while (USART0_Chars_In_Buff()==0)_delay_ms(DELAY_USART0*5);
                USART0_Receive_Block(string_configuracao,SIZE_BLOCO_CONFIGURACAO);
//recebe bloco da tabela de configuração I2C pela USART0 (mc_mestre)
                USART0_Clr_Rxbuff();
                dummyint[0]=strtok(string_configuracao,")");
                sscanf(dummyint[0],"INP(%d",&n_param_a_medir[j]);
                dummyint[1]=strtok(NULL,")");
                dummyint[2]=strtok(NULL,")");
                dummyint[3]=strtok(NULL,")");
                dummyint[4]=strtok(NULL,")");
                dummyint[5]=strtok(NULL,")");
                for (int i=0; i<n_param_a_medir[j]; i++)
                {
                        dummyint[3*i+6]=strtok(NULL,")");
                        dummyint[3*i+7]=strtok(NULL,")");
                        dummyint[3*i+8]=strtok(NULL,")");
                }

                sscanf(dummyint[1],"IPR(%d",&protocolo[j]);
                sscanf(dummyint[2],"ICP(%d",&config_protocolo[j]);
                sscanf(dummyint[3],"IAD(%d",&endereco_i2c[j]);
                sscanf(dummyint[4],"ICI(%s",&cmd_init[j]);
```

```
            sscanf(dummyint[5],"IFI(%d",&fmt_init[j]);
            for (int i=0; i<n_param_a_medir[j]; i++)
            {
                        for (int k=0;k<3;k++)
                        {
                        cod_sensor[j][i][k]=dummyint[3*i+6][4+k];
                        }
                        for (int k=0; k<strlen(dummyint[3*i+8]+4); k++)
                        {
                                    cmd_leitura[j][i][k]=dummyint[3*i+8][k+4];
                        }
                        fmt_leitura[j][i]=atoi(dummyint[(3*i)+7]+4);
            }
}
for (int i=0; i<n_dispositivos_i2c;i++)
{
            for (int j=0; j<n_param_a_medir; j++)
            {
                        size_i2c[i]+=fmt_leitura[i][j];
            }
            size_total_i2c+=size_i2c[i];
}
_delay_ms(DELAY_USART0);
USART0_Transmit('x');//Termina config
_delay_ms(DELAY_USART0);
free(printconfig);
free(string_configuracao);
            if (serial_port_en=1)
{
            for (int i=0;i<serial_port_n_param; i++)
            {
                        serial_port_size+=serial_port_fmt_leit[i];
            }
}
```

```
	USART0_Clr_Rxbuff();
}


//Função que mantém funcionalidades básicas do Microcontrolador Leitura mesmo sem uma configuração
//válida

//Parametros = void

//Retorna = void

void configura_parametros_default()

{


	modo_operacao = MODO_OPERACAO_DEFAULT;

	tempo_aquisicao = TEMPO_AQUISICAO_DEFAULT;

	n_bloco_envio = 0;
	// Nao tem bloco para enviar

	n_timeout = TIMEOUT_DEFAULT;


}


//Função responsável por montar os blocos de medidas

//Parametros = Vetor com o conteúdo do bloco e tamanho do bloco

//Retorna = void

void forma_bloco(char *medida, int size_medida)

{

	char init_blc_vect[27];

	char dum;

	int size_serial=0;

	n_bloco++;

	n_medida=1; //trocar quando estiver pronta a rotina de incrementar medida

	sprintf(init_blc_vect,"MED(%04d),BLC(%010d),",n_medida,n_bloco);

	size_serial=0;


	for (int i=0;i<26;i++)

	{

		*(bloco_leitura+i)=init_blc_vect[i];

	}
```

```
		//////////////////////serial
		for (int j=0; j<serial_port_n_param; j++)
		{

				*(bloco_leitura+size_serial+26)='S';
				*(bloco_leitura+size_serial+27)='P';
				dum=j+49;
				*(bloco_leitura+size_serial+28)=dum;
				*(bloco_leitura+size_serial+29)='(';

			size_serial+=4;
		for (int i=0;i<serial_port_fmt_leit[j];i++)
		{
			*(bloco_leitura+size_serial+26)=serial_port_measure[j][i];
			size_serial++;
		}
		*(bloco_leitura+size_serial+26)=')';
		size_serial++;
		}
		for (int i=26+size_serial; i<26+size_serial+size_medida; i++)
		{
			*(bloco_leitura+i)=*(medida+i-(26+size_serial));

		}
		for (int i=26+size_serial+size_medida; i<SIZE_BLOCO_LEITURA;i++)
		{
			*(bloco_leitura+i)='?';
		}
}

//Função que verifica se algum comando foi enviado pelo Microcontrolador Mestre
//Parametros = void
//Retorna = void
void check_comando_central()
```

```
{
        char data=0;
        char buff_string[STRING_MAXIMO_USART1]="lixo0";
        char *size_measure;
        int to_count=0;
        if (USART0_Chars_In_Buff()>0)
                                        switch (USART0_Receive())
                                {

                                        case 'M':

                                        for (int i=0; i<n_dispositivos_i2c; i++)
                                        {
                                        i2c_start(endereco_i2c[i]+I2C_WRITE);
                                        i2c_write(191);
                                        i2c_stop();
                                        _delay_ms(50);
                                        }
                                        break;

                                        case 't':

                                        bloco_leitura=calloc(sizeof(char),SIZE_BLOCO_LEITURA);

//size_measure=calloc(sizeof(char),SIZE_MEDIDA_NOMAD+SIZE_MEDIDA_AD_HC08);
                                        size_measure=calloc(sizeof(char),SIZE_MEDIDA_SONDA);
                                                // write 191d to HC08 address 16 (Byte Write)
                                                for (int i=0; i<n_dispositivos_i2c; i++)
                                                {

                        i2c_start(endereco_i2c[i]+I2C_WRITE);     // set device address and write mode
                        i2c_write(193);                   // write
                        i2c_stop();
```

```
                    _delay_ms(100);
                    i2c_get_string(size_measure,size_i2c[i],endereco_i2c[i]);
                    _delay_ms(DELAY_USART1);

                    }

                        if (serial_port_en=1)
                        {
                            for (int i=0; i<serial_port_n_param; i++)
                            {
USART1_Transmit_Block(serial_port_cmd_leit[i],strlen(serial_port_cmd_leit[i]));

                                while (USART1_Chars_In_Buff()==0)
                                {
                                    _delay_ms(DELAY_USART0);
                                    to_count++;

                                }
to_count=0;
_delay_ms(DELAY_USART1*5);
USART1_Receive_Block(&serial_port_measure[i][0],serial_port_fmt_leit[i]);
_delay_ms(DELAY_USART1*10);
USART1_Clr_Rxbuff();
                            }
                        }

                            forma_bloco(size_measure,SIZE_MEDIDA_SONDA);
USART0_Transmit_Block(bloco_leitura,SIZE_BLOCO_LEITURA);
                            free (size_measure);
                            free(bloco_leitura);
                            USART0_Clr_Rxbuff();
```

```
                                        break;
                                        case 'C':
                                        //leitura_configuracao();
                                        break;
                                }
}
int main(void)
{
 CLKPR= 0x80;
CLKPR= 0x01;//define clock 4MHz
USART0_Init(26); //initialize usart 0 and define baud rate 9600
USART1_Init(26); //initialize usart 1 and define baud rate 9600
i2c_init();

int status_perifericos_ativos;
char buff_string[STRING_MAXIMO_USART1]="lixo0";
        configura_parametros_default();
        leitura_configuracao();
        _delay_ms(DELAY_USART0);
        for (int i=0; i<serial_port_n_param; i++)
        {
                serial_port_measure[i]=malloc(serial_port_fmt_leit[i]);
        }
   while(1)
        {
          check_comando_central();
        }
}
```