UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

RAFAEL SILVA MONTES

Automação de um frontend de RMN para controle de periféricos de baixa velocidade

São Carlos
2023

RAFAEL SILVA MONTES

Automação de um frontend de RMN para controle de periféricos de baixa velocidade

Dissertação apresentada ao Programa de Pós-Graduação em Física do Instituto de Física de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências.

Área de concentração: Física Aplicada
Opção: Física Computacional
Orientador: Prof. Dr. Alberto Tannús

Versão original

São Carlos
2023

Montes, Rafael
Automação de um frontend de RMN para controle de periféricos de baixa velocidade / Rafael Montes; orientador Alberto Tannús -- São Carlos, 2023.
88 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Física Aplicada Computacional) -- Instituto de Física de São Carlos, Universidade de São Paulo, 2023.

1. Automação. 2. Espectrômetro. 3. Ressonância Magnética. I. Tannús, Alberto, orient. II. Título.

FOLHA DE APROVAÇÃO

Rafael Silva Montes

Dissertação apresentada ao Instituto de Física de São Carlos da Universidade de São Paulo para obtenção do título de Mestre em Ciências. Área de Concentração: Física Teórica e Experimental.

Aprovado (a) em: 31/08/2023

Comissão Julgadora

Dr(a).: Alberto Tannús

Instituição: (IFSC/USP)

Dr(a).: Kalinka Regina Lucas Jaquie Castelo Branco

Instituição: (ICMC/USP)

Dr(a).: Paulo Estevão Cruvinel

Instituição: (Embrapa/São Carlos)

À minha família pelo apoio e incentivo ao longo
período de elaboração deste trabalho.

AGRADECIMENTOS

A oportunidade de desenvolver este trabalho trouxe resultados que vão além do que está exposto neste trabalho. Este trabalho me levou para um laboratório ao qual me ofereceu inúmeras oportunidades de aprendizado e de desenvolvimento profissional. Desta forma eu agradeço ao professor Dr. Alberto Tannús, de me dar a oportunidade de poder contribuir com esse grupo de pesquisa e pelos ensinamentos, paciência e compreensão demonstradas durante todo o meu período de atuação.

Agradeço também ao Mateus José Martins e ao Edson Vidoto, referências como excelentes profissionais de engenharia, pelos ensinamentos passados a mim no decorrer do meu mestrado.

Aos meus pais, expresso minha total gratidão pela vida que me proporcionaram, o cuidado e principalmente por fazer da minha educação a prioridade de suas vidas. Por toda essa dedicação em toda a minha existência, o mérito pela realização deste trabalho torna-se mais deles do que o meu. E, por fim, agradeço às minhas irmãs pela parceria e incentivo. Elas são exemplos aos quais eu sempre me mirei.

“O presente trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) Código de financiamento 001”.

"O estudo, a busca da verdade e da beleza são domínios em que nos é consentido sermos crianças por toda a vida."

Albert Einstein

RESUMO

MONTES, R. S. **Automação de um FrontEnd de RMN para controle de periféricos de baixa velocidade**. 2023. 88 p. Dissertação (Mestrado em Ciências) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2023.

O desenvolvimento de técnicas de ressonância magnética tem se mostrado fundamental pois estas têm grande aplicabilidade na indústria, em análises clínicas, no estudo morfológico da estrutura do material, entre outros. Assim, o desenvolvimento de um espectrômetro com ampla aplicabilidade, como o desenvolvido pelo grupo CIERMag, tem relevância. Tal espectrômetro necessita de circuitos para intermediar as transmissões e recepções dos sinais captados e gerados. Ao conjunto destes, é dado o reconhecimento como o FrontEnd desse espectrômetro. A este deve ser associado um controle que o prepare para os experimentos que forem almejados no sistema ao qual forem inseridos. Este trabalho propõe a automação deste FrontEnd por meio de uma biblioteca desenvolvida em Python 3, sendo que esta prevê a utilização do módulo Raspberry Pi como o meio para realizar este controle. Esta biblioteca contém classes e métodos específicos para as partes do FrontEnd e para a comunicação entre o módulo Raspberry Pi e um terminal cliente que que deve ser utilizado para operar o FrontEnd. A dinâmica entre a Raspberry Pi e o FrontEnd ocorre por meio do protocolo SPI e foi desenvolvido um soquete TCP/IP e uma biblioteca para a manipulação dos métodos de controle do FrontEnd contidos na Raspberry Pi.

Palavras-chave: Automação. FrontEnd de um espectrômetro. Ressonância magnética nuclear.

ABSTRACT

MONTES, R. S. **Automation of an NMR FrontEnd to control low-speed peripheral**. 2023. 88 p. Dissertation (Master in Science) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2023.

The development of magnetic resonance techniques has been shown to be fundamental, as they have great applicability in industry, in clinical analysis, in the morphological study of the structure of the material, among others. Thus, the development of a spectrometer with wide applicability, such as the one developed by the CIERMag group, is relevant. Such a spectrometer needs circuits to mediate the transmissions and receptions of captured and generated signals. The set of these is recognized as the FrontEnd of this spectrometer. This must be associated with a control that prepares it for the experiments that are desired in the system to which they are inserted. This work proposes the automation of this FrontEnd through a library developed in Python 3, which foresees the use of the Raspberry Pi module as the means to carry out this control. This library contains specific classes and methods for the FrontEnd parts and for the communication between the Raspberry Pi module and a client terminal that must be used to operate the FrontEnd. The dynamic between the Raspberry Pi and the FrontEnd occurs through the SPI protocol and a TCP/IP socket and a library were developed for handling the FrontEnd control methods contained in the Raspberry Pi.

Keywords: Automation. FrontEnd of a spectrometer. Nuclear magnetic resonance.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CIERMag	Centro de Imagens e Espectroscopia in vivo por Ressonância Magnética
FPGA	Field Programable Gate Arrays
RMN	Ressonância Magnética Nuclear
SS	Slave Select
GND	Ground
MOSI	Master Output Slave Input
MISO	Master Input Slave Output
SCLK	Serial Clock
VGA	Variable Gain Amplifier
OSI	Open System Interconnection
ISO	International Organization for Standardization
TCP	Transmission Control Protocol
ICMP	Internet Control Message
MTU	Maximum Transmission Unit
UDP	User Datagram Protocol
SYN	Synchronize Sequence Numbers
ACK	Acknowledgement)
SPI	Serial Peripheral Interface
GPIO	General Purpose Input and Output
CSI	Camera Serial Interface
JSON	JavaScript Object Notation

LISTA DE SÍMBOLOS

dB	decibel
dBm	decibel miliwatt
V	Volts

SUMÁRIO

1 INTRODUÇÃO

O estudo referente ao fenômeno da ressonância magnética nuclear teve seus registros iniciais disponíveis para a comunidade científica em 1946 em pesquisas independentes. O primeiro estudo foi realizado pela Universidade de Harvard pelos pesquisadores Purcell, Torrey e Pound. A análise do fenômeno relatado por tais pesquisadores tinha como objeto de estudo uma estrutura sólida. A publicação paralela se relacionava à uma substância líquida, esta foi realizada na Universidade de Stanford por Bloch, Hansen e Packard. (1-2) Atualmente as técnicas que se utilizam desse fenômeno para obter informações de alguma matéria tem ampla utilização, sendo que aplicações são encontradas na medicina, indústria do petróleo, alimentícia entre outras.

Para observar o fenômeno da ressonância magnética nuclear, deve-se escolher um elemento da tabela periódica que o manifeste. Entre esses elementos podemos citar o Hidrogênio, Carbono, Flúor, entre outros. A matéria escolhida deve ser exposta a um campo magnético constante o que o induz os núcleos do elemento que manifeste o fenômeno citado a ter a direção e o sentido do momento magnético relativo a eles a estar alinhado com o do campo constante ao qual estão expostos. Ao dispor esses núcleos a um outro campo magnético, entretanto variável e perpendicular ao campo magnético contínuo, o momento relativo tende a deslocar a sua direção de um ângulo θ. Após um pulso de campo variável o momento tende a se alinhar novamente com a direção do contínuo. Essa variação das componentes desse momento é captada por bobinas de recepção que, ao serem dispostas a essa variação, conduzem uma corrente elétrica que representa a resposta desse momento ao pulso do campo magnético. (1)

Note que esse momento magnético é referente ao somatório dos momentos individuais de diversos átomos do mesmo núcleo. Um fator importante para a observação dessa ocorrência é a frequência do campo magnético variável. Cada núcleo manifesta o deslocamento do momento quando a perturbação magnética for de uma frequência específica para aquele elemento, o que chamamos de frequência de Larmor. (3) Esta frequência, ω, está diretamente relacionada ao campo magnético constante, B, e a razão giromagnética, γ, que é uma constante característica de cada núcleo, conforme Equação 1.

$$\omega = \gamma \, . \, B \qquad (1)$$

Percebe-se que a captação do fenômeno tem direta relação com o campo magnético variável, deste modo, o comportamento desse campo magnético deve ser projetado para que as

características do núcleo que se deseja obter sejam recolhidas. Assim, dentro do estudo da ressonância magnética o desenvolvimento de sequências de pulsos tem alta relevância, tanto para o que se objetiva com o uso da técnica, como para buscar o aperfeiçoamento da tecnologia.

Os espectrômetros de ressonância magnética são elementos que auxiliam na produção e captação do fenômeno da ressonância magnética. É a partir dele que são gerados os pulsos de radiofrequência para a perturbação dos núcleos. As respostas a essas perturbações geradas por esses pulsos e captadas pelas bobinas de recepção são recebidas e interpretadas pelo espectrômetro. Existem diversas marcas que oferecem espectrômetros de ressonância magnética nuclear, entretanto, além de serem de custo elevado, o hardware e software nem sempre apresenta os recursos necessários para a implementação desejada, assim é necessário a colaboração da empresa associada ao espectrômetro, o que dificulta o estudo e aprimoramento de sequências de pulsos. (4) Tal dificuldade foi fator determinante para que o grupo de pesquisas CIERMag decidisse desenvolver o seu próprio espectrômetro de ressonância magnética nuclear. O objetivo central era obter um espectrômetro que fosse de caráter geral, isto é, pudesse ser adaptado para qualquer tipo de aplicação. A facilidade de utilização do espectrômetro também é fator de grande interesse para o grupo, visto que o público-alvo que fará uso do mesmo são pesquisadores do fenômeno que não necessariamente são peritos em programação e hábeis para lidar com o hardware dele.

O espectrômetro do CIERMag foi desenvolvido com a *Field Programable Gate Arrays*, FPGA. Tal decisão permite que durante o desenvolvimento desse espectrômetro as alterações da estrutura de hardware não necessitam da substituição do aparato físico, é necessário apenas a atualização do software da FPGA. (5) Para dar suporte às operações do CIERMag uma série de recursos foram criadas. Dessa forma o desenvolvimento do espectrômetro foi realizado em conjunto com o das ferramentas: Console, IDE, linguagem F, Editor de Sequências Gráfico.

Os sinais gerados pelo espectrômetro e por ele captados passa por um FrontEnd, hardware responsável por adaptar esses sinais para o amplificador e pré-amplificador dos transdutores usados no equipamento de Ressonância Magnética. O espectrômetro desenvolvido pelo CIERMag faz o uso de mais de um canal de comunicação, afinal nele opera técnicas que incluem múltiplos canais de recepção e de transmissão.

O FrontEnd do espectrômetro de ressonância magnética do CIERMag tem como elemento base a TX-RX-Box-Controller, que é uma placa de circuito impresso a qual podem ser conectadas até nove transmissores ou receptores de sinais de radiofrequência. Esses sinais de radiofrequência transmitidos se referem às sequências de pulsos responsáveis por produzir o fenômeno da ressonância magnética nuclear e os de recepção são referentes a resposta à

perturbação gerada pelos sinais de transmissão que, em suma, trata-se da variação do momento magnético resultante captada por bobinas de recepção e que expressam o mesmo fenômeno. O Tx-RX-Box-Controller pode ser endereçado com valores de 0 até 7, por meio de um circuito integrado demultiplex. Desta forma podem ser trabalhados um conjunto de 8 dessas placas por um mesmo controlador, que no caso deste trabalho será uma Raspberry Pi. As interfaces de comunicação com os transmissores ou receptores são os denominados slots, sendo que 8 são considerados os principais e o nono é chamado de auxiliar. Assim, cada slot principal é endereçado dentro de um mesmo TX-Rx-Box-Controller como valores de 0 até 7, sendo que este endereçamento também se dá por um demultiplex. O slot auxiliar é acionado pelo endereço da própria Tx-Rx-Box-Controller. A comunicação dos transmissores e dos receptores para com a TX-Rx-Box-Controller, bem como desta para com a Raspberry Pi, se dá por meio do protocolo de comunicação serial SPI (Serial Peripheral Interface). Perceba que para esse projeto são utilizados dois canais de seleção (SS - Slave Select), sendo que um deles é para o manejo dos slots principais e o outro é para a seleção da Tx-RX-Box-Controller que se deseja trabalhar e seleção do slot auxiliar, quando for requerido o trabalho com ele. Ao Tx-Rx-Box-Controller são conectados 4 canais de recepção e transmissão para com o espectrômetro. A distribuição desses canais em relação a ordem dos transmissores e receptores conectados segue uma determinação do projeto do espectrômetro e seus softwares correspondentes. Aos receptores cabe dois estados de configuração principal por meio do Tx-Rx-Box-Controller, que são os estados de ganho alto ou baixo. Eles também possuem uma VGA que permite mais uma configuração adicional de ganho para os sinais que chegam a esses receptores. Os transmissores podem ser atenuados em até 63 dB em degraus de 0,5 dB. A configuração desses estados é realizada pelo controlador, a Raspberry Pi conectada.

A implementação dos recursos principais do espectrômetro desenvolvido pelo CIERMag foi realizada dentro de uma FPGA, ela é muito recomendada para aplicações em que desempenho e tempo real são de grande importância, deliberou-se utilizá-la exclusivamente para este tipo de operação. Afinal, com o desenvolvimento contínuo do espectrômetro pelo CIERMag, liberar a FPGA do controle de periféricos que não precisam dos recursos real time abriria espaço para o desenvolvimento e acréscimos de funcionalidades do espectrômetro. Desta forma pensou em utilizar Raspberry Pi para a realização do controle de periféricos de baixa velocidade, o denominado FrontEnd do espectrômetro de RMN. O objetivo geral descrito dessa dissertação é a estruturação de uma arquitetura de software que possibilite o controle do FrontEnd para periféricos em baixa velocidade de um espectrômetro de ressonância magnética nuclear através de um conjunto de Raspberries Pi.

Com o intuito de atender a esse objetivo o entendimento da topologia do FrontEnd, no que se refere ao hardware, do espectrômetro e o entendimento da lógica digital que se resulta pela disposição dos dispositivos digitais dele se fez necessário. O contexto do FrontEnd para o espectrômetro, isto é, a função deste para com as operações do espectrômetro também se fez necessária, uma vez que as funcionalidades e as respostas que o aquele deve fornecer devem satisfazer as necessidades e as especificações que esta precisa. Uma vez que definiu-se utilizar uma Raspberry Pi para o controle de periféricos lentos, ao desenvolvedor coube adquirir o conhecimento e a habilidade para realizar o manejo dela. O entendimento da topologia permitiu a construção de bibliotecas específicas para a operação de cada hardware que compõe o FrontEnd. Estas contêm métodos específicos para operação dos mesmos e desenvolvidas utilizando o paradigma de programação orientada a objeto, para que a manutenção do software seja simplificada. Uma prévia e superficial análise do projeto do FrontEnd mostra a necessidade de compreender o protocolo de comunicação serial SPI, Serial Peripheral Interface, já que este é utilizado para a troca de informações entre uma Raspberry Pi e o FrontEnd.

O software desenvolvido de controle deve prevê a comunicação da Raspberry com um terminal computador. Foi estipulada um protocolo para essa comunicação de forma que o módulo Raspberry consiga interpretar e responder ao que for solicitado pelo terminal. O terminal também é gerido por uma biblioteca desenvolvida em Python 3 que interpreta as mensagens recebidas da Raspberry, bem como gerar e enviar as solicitações do operador que o controla.

A estruturação desta dissertação será dividida em mais seis partes, além desta introdução. Na primeira, o autor explicita a metodologia utilizada para o desenvolvimento do projeto. Em seguida, uma necessária revisão bibliográfica com os tópicos técnicos de interesse do projeto será apresentada. Os materiais, tanto físicos como softwares de terceiros, são apresentados no terceiro capítulo. No quarto capítulo o leitor encontrará a descrição do desenvolvimento do projeto. E, por fim, nos últimos dois capítulos são apresentados os resultados e conclusões, sendo que neste último serão abordadas projeções futuras para o andamento desta pesquisa.

2 METODOLOGIA

A proposta deste trabalho é consequência do desenvolvimento do Espectrômetro de Ressonância Magnética Nuclear do CIERMag. Após anos de trabalho o espectrômetro comportava diversas funcionalidades, implementadas na FPGA, às quais havia a projeção de adicionar outras mais. Sendo a FPGA um recurso de hardware, limitações são encontradas. No caso específico a memória é um fator limitante. Dessa forma, a ideia de delegar funções a outro tipo de hardware que não necessitam das características que a FPGA é capaz de fornecer, como aplicações que precisam ser abordadas em tempo real. Considerou-se então o uso módulo Raspberry Pi para o controle de periféricos de baixa velocidade, que no caso são os associados ao FrontEnd do espectrômetro de RMN. Para tanto, este trabalho surge com o objetivo geral de desenvolver um software que possibilite o controle do FrontEnd, por meio de um módulo Raspberry, do espectrômetro de ressonância magnética do CIERMag .

Em se tratando de um desenvolvimento científico, uma revisão de literatura se fez necessária. Esta é essencial para qualquer projeto de pesquisa, uma vez que o desenvolvimento de um conhecimento se dá através de um referencial teórico existente. (6) Segundo HANT (7), a revisão de literatura contribui para a capacitação teórica e técnica do pesquisador, uma vez que o engaja a uma atitude investigadora, que o levará a uma pesquisa mais aprofundada além de o fazer questionar mais sobre o tema proposto. O mesmo autor, (7), define a revisão de literatura como:

> “A seleção de documentos disponíveis (publicados e não publicados) sobre o tema, que contenham informações, ideias, dados e evidências escritas a partir de um determinado ponto de vista para cumprir certos objetivos ou expressar certas opiniões sobre a natureza do tema e como ele deve ser investigado, e a avaliação efetiva desses documentos em relação às pesquisas propostas.” (HANT, 1998, p.13)

A revisão de literatura realizada teve como foco a identificação de teorias, práticas e aplicações de um conceito a partir de uma perspectiva neutra. Buscou-se então sintetizar a literatura existente e organizá-las por conceito e realizar a contextualização histórica. A partir da revisão da literatura a escrita do referencial teórico é realizada. Referencial teórico, pode ser definido como (6):

> “É um texto estruturado escrito com redação acadêmica formal, resultante da busca, recuperação e organização da literatura sobre um tema a ser pesquisado, proporcionando uma base teórica consistente para o desenvolvimento de um trabalho de pesquisa”.

Uma vez que se tem definido o objetivo geral da pesquisa e foi-se realizado uma investigação de literatura adequada ao tema, a definição dos objetivos específicos, aqueles que direcionam o pesquisador para a conclusão do objetivo geral, se torna mais assertiva. Para essa pesquisa os objetivos específicos se resumem a:

1. Entender a topologia do FrontEnd do espectrômetro de ressonância magnética e como configurá-lo.
2. Compreender a função do FrontEnd para a operação do espectrômetro de ressonância magnética.
3. Dominar a topologia da Raspberry Pi no que se mostrar necessário para o controle do FrontEnd.
4. Entender o protocolo de comunicação serial SPI, pois este é fundamental para a comunicação entre o FrontEnd e as Raspberry Pi.
5. Elaborar uma biblioteca para o controle das partes específicas do FrontEnd.
6. Definir como será realizada a comunicação entre a Raspberry Pi e um terminal computador.
7. Buscar o entendimento da ferramenta a ser utilizada para construir a comunicação entre as Raspberry Pi e os Usuários.
8. Desenvolver a comunicação entre a Raspberry Pi e os usuários.
9. Unir as partes referentes ao controle do FrontEnd e a comunicação com o terminal.

Verifica-se que o cumprimento dos objetivos 1, 2 e 3 são necessários para poder atender os demais. Mas os objetivos 4 e 5 podem ser desenvolvidos em uma frente de trabalho diferente dos 6, 7, cabendo ao objetivo 8 realizar a conexão entre as partes. Dessa forma a abordagem de desenvolvimento do projeto pode ser esquematizada em duas frentes que são dependentes dos mesmos pré-requisitos, pois são direcionados a interagir com o mesmo objeto, o FrontEnd do Espectrômetro de RMN do CIERMag. Dessa forma a metodologia proposta para o desenvolvimento segue a forma da Figura 1.

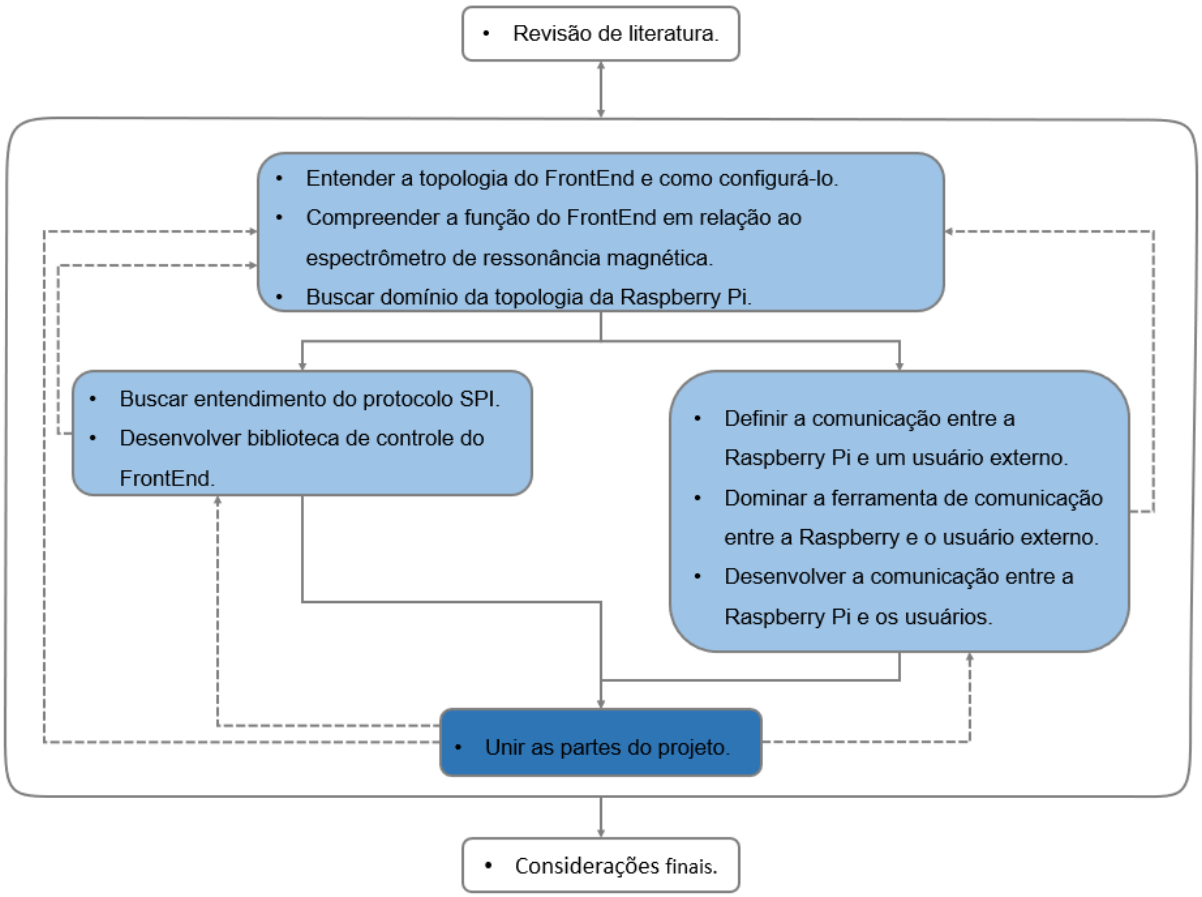

Figura 1 – Representação da metodologia de desenvolvimento
Fonte: Elaborada pelo autor

Repare que das frentes de trabalhos há setas pontilhadas apontando para as etapas anteriores. Tal representação representa a verificação se elas estão atendendo aos requisitos que os primeiros objetivos oferecem. Pode-se compreender dela, também, a necessidade de retornar para a realização das primeiras etapas para o caso de se chegar a conclusão durante o desenvolvimento de que os requisitos não são cabíveis, necessitando novas análises relacionadas ao FrontEnd ou ao seu contexto de aplicação no espectrômetro de RMN.

Da seta que sai do quadro que contém os últimos objetivos específicos e chega ao primeiro quadro, podemos ter o mesmo entendimento. Entretanto, soma-se a ele que ao final de uma primeira versão funcional espera-se um processo de otimização do projeto e que deve se atentar ao cumprimento dos requisitos e o aprimoramento das técnicas obtidas pela realização dos primeiros objetivos específicos. Nota-se também setas deste último quadro para os das frentes paralelas. Tais setas dizem respeito a necessidade de possíveis mudanças que sejam necessárias para que possibilite o cumprimento dos últimos objetivos específicos. Outro ponto de destaque na metodologia seguida é que durante o desenvolvimento do trabalho não é

dispensável o processo de revisão de literatura. Este foi importante para direcionar o desenvolvimento do trabalho, para fundamentar os objetivos específicos, e continuou sendo relevante durante o desenvolvimento, sempre que o pesquisador achou necessário um aprimoramento teórico.

3 REVISÃO TEÓRICA

3.1 O espectrômetro do CIERMag

O estudo da sequência de pulsos de radiofrequência é essencial para a evolução das técnicas de ressonância magnética nuclear. Para que ele seja possível, pesquisadores necessitam de um espectrômetro que lhes dê a possibilidade de configuração de hardware e software a fim de adaptá-lo para a aplicação que se desejar. Apesar de existirem no mercado ofertas de espectrômetros, eles possuem software e hardware proprietários, o que torna a característica de adaptabilidade, em termos de software e hardware, difícil e lenta, pois há dependência direta com o fabricante. (5) Neste contexto, o desenvolvimento de um espectrômetro de ressonância magnética nuclear que forneça a tão almejada, por pesquisadores de técnicas de ressonância magnética nuclear, característica de versatilidade se torna imprescindível. Tamanha necessidade motivou o grupo de pesquisa CIERMag a desenvolver o seu próprio espectrômetro de ressonância magnética nuclear.

Nota-se que a característica de mobilidade de hardware tornaria a adaptação do espectrômetro para atender o uso que se requisitar é desejável. Tendo isso em vista, o CIERMag optou por desenvolver o seu equipamento em uma FPGA. As FPGAs são “dispositivos de silício pré-fabricados que podem ser programados eletricamente para se tornarem quase qualquer tipo de circuito ou sistema digital." (5) De acordo com (4) um desenho de hardware desenvolvido em uma FPGA pode ser programado e testado imediatamente e a mesma FPGA pode ser utilizada em diferentes projetos. Tais características atendem as necessidades requeridas pelo CIERMag, afinal esta permite a adaptabilidade do hardware para acompanhar a aplicação que se desejar, seja espectroscopia ou geração de imagens. Outro fator interessante é que o espectrômetro pode ser aprimorado sem a necessidade de substituição física, já que basta inserir a nova configuração de hardware na FPGA.

O espectrômetro desenvolvido necessita de um suporte de software para a sua operação. Sendo assim, um conjunto de ferramentas foram desenvolvidas para darem operabilidade ao espectrômetro, que são: Console, IDE, linguagem F, Editor de Sequências Gráfico. (5) A console é responsável pela operação do espectrômetro por um usuário que necessita conduzir um experimento. A linguagem F e o Editor de Sequências Gráfico são utilizados para a geração de sequências de pulsos, sendo que a segunda é uma implementação gráfica. E por fim, a IDE, que é utilizada para o gerenciamento dos parâmetros de um método ou do próprio sistema. (4)

3.2 FronEnd do espectrômetro do CIERMag

Neste projeto o equipamento de controle do FrontEnd, uma Raspberry Pi, é responsável pelo controle dos sinais de entrada e saída do espectrômetro. A recepção e transmissão desses sinais necessitam de módulos intermediários chamados de receptores e transmissores. Esses,

atuam atribuindo atenuações para os sinais que por eles transcorrem para que eles sejam emitidos ou recebidos em um valor ao qual possam corresponder, no caso dos transmissores, as necessidades de um experimento com o espectrômetro ou, se tratando dos receptores, serem percebidos e interpretados. O FrontEnd é o elemento que corresponde ao hardware, desenvolvido por pesquisadores do CIERMag, que dispõe desses módulos. Este hardware é caracterizado por três tipos diferentes de placas eletrônicas que são: Tx-Rx-Box-Controller, Receiver-Box, Transmitter-Box.

3.2.1 Tx-Rx-Box-Controller

A primeira delas é a base de comunicação entre o equipamento de controle, a Raspberry Pi, e os módulos de transmissão e recepção. É uma placa, com alimentações de 5 e 12 volts, ao qual os transmissores e receptores são conectados. A entrada dela corresponde a dois conectores paralelos idênticos em quantidade de pinos e no significado de cada um deles, isto é, os pinos que se encontram na mesma posição em cada conector possuem o mesmo valor. Tal característica permite que sejam conectadas diversas placas desse mesmo tipo em paralelo. Para que os transmissores e receptores possam ser identificados de maneira individual, esta possui demultiplexadores de 8 canais que permitem que sejam endereçadas até 8 placas destas e 8 transmissores ou receptores. Assim, esses módulos podem ser identificados exclusivamente pelo endereçamento da Tx-Rx-Box-Controller que está conectada e pelo seu endereço nesta.

A comunicação pela Raspberry Pi é realizada com o protocolo de comunicação serial SPI, sendo que a entrada do sinal SS, Slave Select, se dá pela programação do multiplex. Tal procedimento só é possível pelo fato de que cada canal de saída do multiplex é seguido por uma interrupção, um jumper, entre ela e a parte seguinte do circuito. Ao soldar, conexão entre as partes, em uma saída correspondente ao endereço da TX-RX-Box-Controller, o sinal SS só percorrerá essa placa caso o endereço no multiplex corresponder a essa porta.

Nesta placa são encontrados mais 9 conectores. Estes correspondem aos slots em que serão conectados os transmissores e receptores, sendo que 8 são considerados principais e um auxiliar. Os slots principais utilizam o mesmo sinal de SS, e o auxiliar tem o seu específico. Percebe-se que o equipamento deve possuir uma interface SPI com dois canais de seleção SS, condição essa correspondida pela Raspberry Pi utilizada por este projeto. Cada um desses slots tem seu endereço associado a sua posição na placa, sendo que este é caracterizado por três pinos em que se chega alimentação ou terra para corresponder a posição em binário. Por exemplo, o slot correspondente ao slot 0, terá os 3 pinos conectados ao GND, caracterizando assim o

endereço digital 000. Já o slot 5, terá o primeiro e o último pinos de endereço alimentados com 5V e o do meio ligado ao GND, caracterizando o endereço lógico 101, que corresponde ao número 5 em binário. O slot auxiliar sempre terá seu endereço lógico interno em 000, ou seja, seus pinos de endereço são conectados ao referencial terra, GND.

O espectrômetro do CIERMag possui quatro canais de transmissão e de recepção, em vista disto, no Tx-Rx Box Controller encontram-se 4 dutos para os canais de transmissão e 4 para os de recepção. A atribuição deles para cada um dos circuitos conectados, transmissores ou receptores, nos slots principais deve seguir a regra indicada nas Tabelas 1 e 2. Há dois dutos nesta placa que se iniciam nos conectores de entrada e se encerram nos conectores dos slots que serão utilizados para a configuração do canal de transmissão ou recepção o circuito conectado irá reconhecer.

Tabela 1 – Atribuição do canal para os transmissores em relação à sua ordem de identificação

Nº do Tx	Canal
1	TxGATE1
2	TxGATE2
3	TxGATE3
4	TxGATE4
> 4	TxGATE1

Fonte: Elaborada pelo autor.

Tabela 2 – Atribuição do canal para os receptores em relação à sua ordem de identificação

Nº do Rx	Canal
1	RxGATE1
2	RxGATE2
3	RxGATE3
4	RxGATE4
> 4	RxGATE1

Fonte: Elaborada pelo autor.

3.2.2 Transmitter-Box

Estes dispositivos correspondem às placas que transmitem sinais de radiofrequência para o espectrômetro de RMN. Nele encontram-se filtros característicos, amplificadores e outros componentes que o caracterizam em relação ao núcleo ao qual vai se operar e ao campo magnético do magneto que irá se utilizar para integrar o conjunto de equipamentos para se realizar um experimento de RMN. Esses transmissores podem ser configurados para atenuar o sinal de entrada em até 63 dB, em degraus de 0,5 dB, além da atenuação característica das condições de hardware do próprio transmissor, isto é, as atenuações impostas por cada elemento que compõe o transmissor. Em seu conector de entrada, esse transmissor recebe os canais dos receptores e transmissores, sinais para a seleção do canal de transmissão, o endereço que ele

corresponde naquela Rx-Tx-Box-Controller, alimentação de 5V e de 12V, sinal MOSI vindo da Raspberry pi e sinal de seleção. E, por meio desse mesmo conector informa que é um transmissor, envia o sinal MISO para a Raspberry e o sinal digital que é utilizado para ligar um led na Tx-Rx-Box-Controller para dar indicar que o transmissor está operável.

3.2.3 Receiver-Box

Este dispositivo corresponde ao meio, hardware, de envio dos sinais resultados do fenômeno de ressonância magnética nuclear para o espectrômetro do CIERMag. Ele pode ser configurado para amplificar de duas maneiras diferentes. A primeira delas corresponde a uma sequência de dois amplificadores operacionais, aos quais o operador pode selecionar se os dois vão atuar na amplificação do sinal ou se será apenas um deles. A outra maneira é com a programação da VGA presente neste componente. O objetivo dessas amplificações é elevar o sinal captado para uma faixa que possa ser identificada pelo espectrômetro. Ele basicamente contém as mesmas funcionalidades em seu conector utilizado para agregá-lo em uma Tx-Rx-Box-Controle. A diferença é que ele informa que é um receptor.

3.3 Modelo de referência OSI

O modelo OSI (Open Systems Interconnection) é uma proposta desenvolvida pela ISO (International Standards Organization) com a intenção de padronizar, internacionalmente, protocolos empregados em diversas camadas.

Este modelo de rede é organizado em sete camadas com atribuições específicas. É importante notar que esta referência apenas delimita o que cada camada deve realizar, não o como, ou seja, não é determinado os serviços e protocolos específicos para cada camada.

De acordo com TANENBAUM (8), a elaboração do modelo de referência OSI, no que diz respeito à definição das camadas, foi elaborado respeitando princípios relativos à necessidade de separar abstrações de diferentes contextos, ter funções que estejam de acordo com protocolos padronizados internacionalmente, minimizar os fluxos de informações entre interfaces, possuir número de camadas em quantidade suficiente para que cada uma tenha funções diferentes contando que não é desejável a criação de um grande número de camadas para não tornar o modelos deveras complexo.

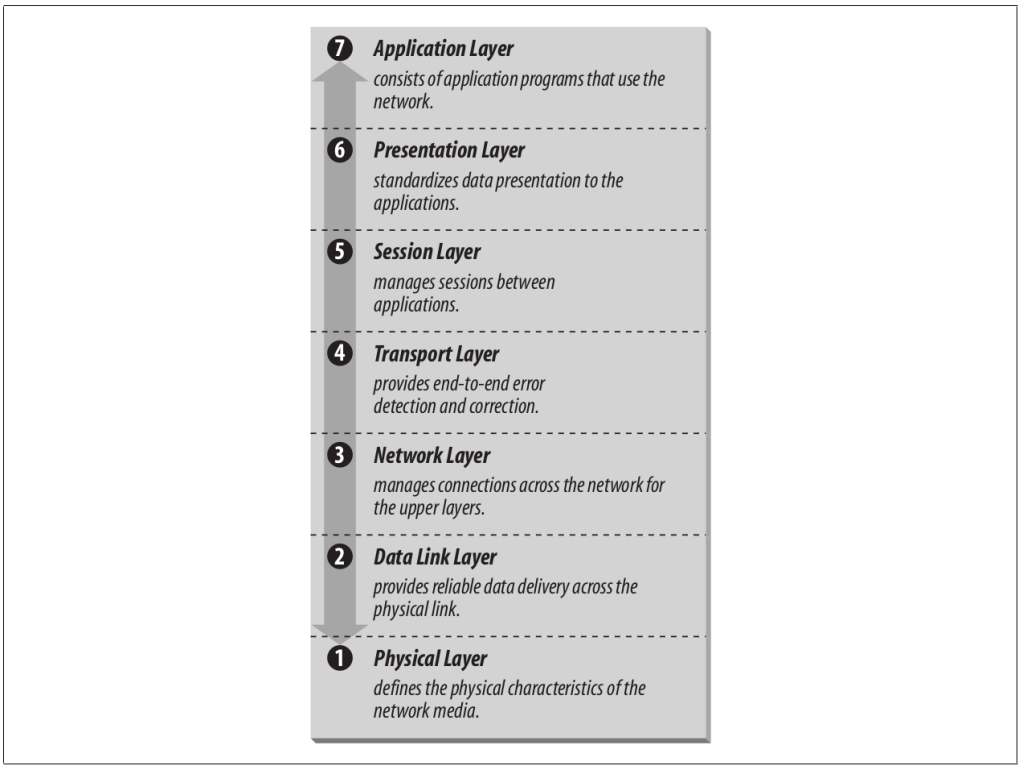

Figura 2 – O Modelo de Referência OSI
Fonte: HUNT. (9)

3.3.1 Camada física

A camada física é a responsável por garantir que os bits enviados por um lado da comunicação sejam recebidos da forma como foram enviados pela parte a qual a comunicação foi direcionada. De acordo com TANENBAUM (8), o projeto da camada física “deve garantir que, quando um lado enviar um bit 1, o outro lado o receberá como um bit 1, não como um bit 0”. De acordo com o mesmo autor, a preocupação do projeto da camada física está relacionada a questões relacionadas ao meio físico de transmissão, sincronização, interfaces físicas e elétrica. Dessa forma, exemplos de questões que são tratadas por essa camada são a duração do bit, quantidade e finalidade dos pinos dos conectores de rede, se terá simultaneidade da transmissão nos dois sentidos, como será realizada a conexão inicial, como finalizá-la e os valores de tensão relativos ao nível lógico alto e baixo, representados pelos bits 1 e 0, respectivamente

3.3.2 Camada de enlace de dados

A principal tarefa desta camada é realizar a divisão dos dados a serem transmitidos em quadros de dados, formado por centenas ou milhares de bytes, e os envia sequencialmente e, no

caso de serviços confiáveis, haverá a confirmação do receptor acerca da recepção correta do quadro. (9) O objetivo é tornar a linha de transmissão livre de erros não detectados.

Outras atribuições que estão, de certa forma, relacionadas ao objetivo principal desta camada são realizar o manejo de acesso compartilhado e o controle de tráfego, essencial em casos em que o transmissor envia dados numa taxa maior que o receptor consegue lidar.

3.3.3 Camada de rede

Esta camada atua no roteamento dos dados da origem até o destino. O roteamento pode ser orientado por uma tabela, definida no início de cada transmissão ou atribuída de forma dinâmica. (9)

É responsabilidade desta camada realizar o controle de congestionamento causado por múltiplos pacotes na sub rede no mesmo instante. A qualidade do serviço dessa camada é associada ao retardo na transmissão, tempo em trânsito dos dados e possíveis instabilidades. Outros problemas que podem acontecer neste nível são a incompatibilidade do formato de endereços entre origem e destino, aceitação dos pacotes enviados devido ao tamanho dele, diferença nos protocolos dessa camada. Esta camada deve ser projetada para lidar com todos esses problemas, a fim de ter uma abordagem genérica o suficiente para ser possível conectar redes heterogêneas.

3.3.4 Camada de transporte

Esta camada tem como principal atribuição receber os dados das camadas superiores e encaminhar para as inferiores. A implementação desta camada deve ser implementada de maneira que, além de ser eficiente, as camadas superiores fiquem isoladas das inferiores para que possíveis mudanças na tecnologia de hardwares não as afete. (9) Ao admitir os dados das camadas superiores esta camada é responsável por dividi-los, caso seja necessário, em unidades menores e garantir que os dados cheguem na outra extremidade.

Nesta camada é estabelecida o tipo de serviço que deve ser fornecido para a camada de rede e, por consequência, para o usuário. Os tipos de serviços podem estar relacionados a garantia da entrega ou não dos dados na ordem em que foram enviados, como por exemplo as do tipo ponto a ponto, as mensagens isoladas, ou quando se deseja enviar mensagens para muitos destinos. (9) A determinação do tipo de serviço é realizada quando a conexão for estabelecida.

A camada de transporte é uma camada fim a fim, ou seja, ela mantém comunicação constante com a parte equivalente em outra máquina, para tanto faz o uso de cabeçalhos de mensagens e mensagens de controle, para controlar a comunicação entre as partes.

3.3.5 Camada de sessão

Esta camada permite o estabelecimento de sessões entre usuários de diferentes máquinas. (9) A camada de sessão oferece controle da comunicação entre as partes, realiza o gerenciamento das operações críticas de forma que não sejam realizadas duas ao mesmo tempo por partes diferentes e atua na sincronização da transmissão.

3.3.6 Camada de apresentação

Esta camada trata da semântica e da sintaxe, ela é responsável por apresentar as informações da maneira que seja pertinente para a aplicação. Segundo HUNT (9), deve haver concordância entre as partes, aplicativos que se comunicam, em relação a maneira como os dados são apresentados. Esta camada fornece as rotinas apropriadas para a apresentação dos dados padrões.

3.3.7 Camada de aplicação

Representa a aplicação finalidade do usuário. De acordo com HUNT (9), “a camada de aplicativo é o nível da hierarquia de protocolo onde residem os processos de rede acessados pelo usuário”. A camada de aplicativo inclui todos os processos que interage de alguma forma com o usuário ou que esteja nesse mesmo nível de abstração.

3.4 TCP IP

De acordo com HUNT (9), os protocolos básicos de TCP/IP foram desenvolvidos dentro da Agência de Projetos de Pesquisa Avançada com a continuidade do desenvolvimento da rede ARPAnet. Ao ser adotada como *Militarry Standards* - Mil STD, em 1983, foi necessário que todos os hosts fossem alterados para abrangê-los. Diante deste cenário, TCP/IP foi implementado no sistema operacional Berkeley Unix. Atualmente o TCP/IP é um padrão em redes de computadores.

HUNT, (9), enfatiza que a popularidade associada a este protocolo está associada ao fato deste oferecer a capacidade de comunicação em escala mundial e por possuir padrões de

protocolos abertos e gratuitos, poder ser utilizado independente do hardware de computador, da rede física e do sistema operacional, possibilita endereçamento exclusivo dos dispositivos da rede e pelo motivo de seu protocolo ser padronizado.

Sendo o modelo OSI, o de referência definido pela ISO, o TCP/IP pode ser abstraído tomando-o como referência. De acordo com HUNT (9), a camada de aplicação é definida como todos os processos que ocorrem acima da camada de transporte. A camada de Apresentação não é distinguida como uma camada única, sendo que as funções desta camada são tratadas pelos aplicativos TCP/IP. A camada de sessão também não é identificável dentro do TCP/IP, suas funções podem ser visualizadas na camada de transporte, entretanto o termo sessão não é abordado, mas está relacionado com os termos “soquete” e “porta”. A camada de transporte é bem caracterizada no modelo TCP/IP, assim sendo a garantia de que todos os dados sejam recebidos exatamente como foram enviados é executada pelo TCP. A camada de rede é distinguida dentro do protocolo, assim, a responsabilidade de isolar os protocolos das camadas superiores dos detalhes referentes a rede subjacente é de responsabilidade do IP, que também lida com o endereçamento e a entrega dos dados. A camada de enlace não é identificada neste protocolo, sendo que as suas atribuições são englobadas pelo IP. E, por fim, a camada física não é definida nesse protocolo, isto é, este protocolo não define as características do hardware necessário para a transmissão, ele utiliza o que estiver disponível.

Apesar de ser possível descrever esse protocolo a partir do modelo OSI, a sua realização não oferece pleno entendimento da arquitetura do protocolo TCP/IP. Por esse motivo, será apresentado um modelo de quatro camadas que é relatado em HUNT. (9) Este modelo é expresso pelas camadas de: acesso à rede, internet, transporte e de aplicação. A seguir é detalhado cada uma delas.

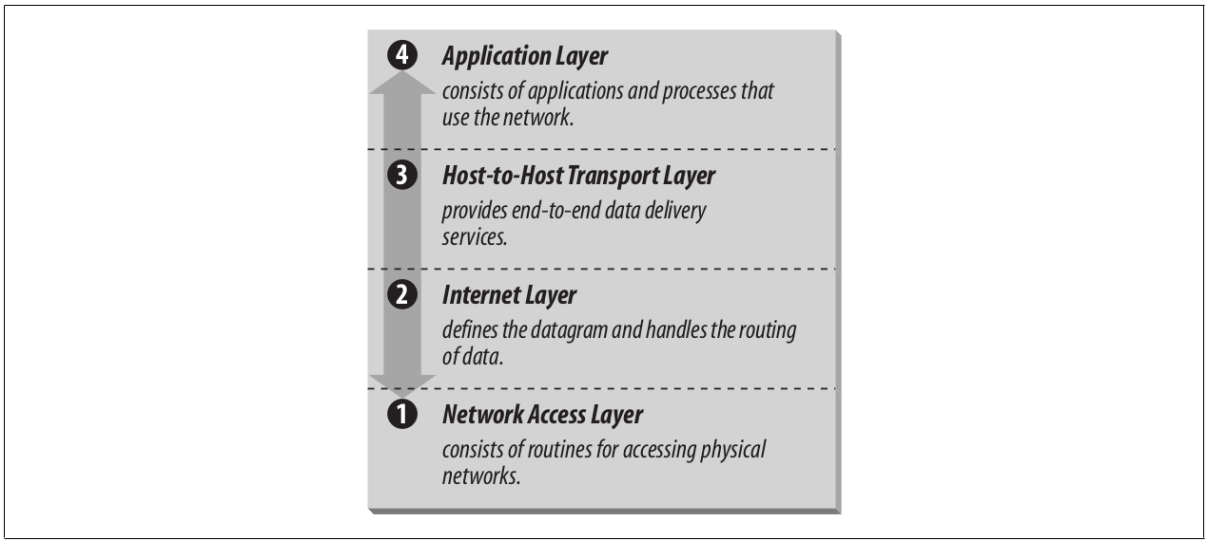

Figura 3 – A Arquitetura TCP/IP
Fonte: HUNT. (9)

De acordo com HUNT (9), as informações seguem o mesmo direcionamento do modelo OSI, ou seja, quando o host deseja enviar dados, eles descem a pilha de camadas do protocolo, ao serem recebidos eles sobem a mesma. Essa estrutura visualmente se justifica na maneira como os dados são lidados à medida que percorrem as parcelas envolvidas na transmissão. A clareza desse fato está na verificação da dinâmica dos acréscimos de cabeçalho aos dados que se deseja enviar quando eles saem de uma camada superior para um inferior e o decréscimo, no sentido contrário, quando são recebidos. Esses cabeçalhos contêm as informações pertinentes para que a camada correspondente no dispositivo que receber as informações saiba como lidar com elas no que cabe às suas responsabilidades. É importante enfatizar que uma camada não sabe discernir as informações adicionadas pelas outras, elas entendem toda a informação que recebem como um bloco de dados a ser transmitidos. Assim uma camada só se preocupa em adicionar as informações de controle da camada quando se objetiva a transmissão de dados e recolher as informações quando há recepção. A única preocupação que uma camada deve ter em relação a outra é como enviar e receber dados. Essa técnica de adicionar cabeçalhos na frente de dados se chama encapsulamento.

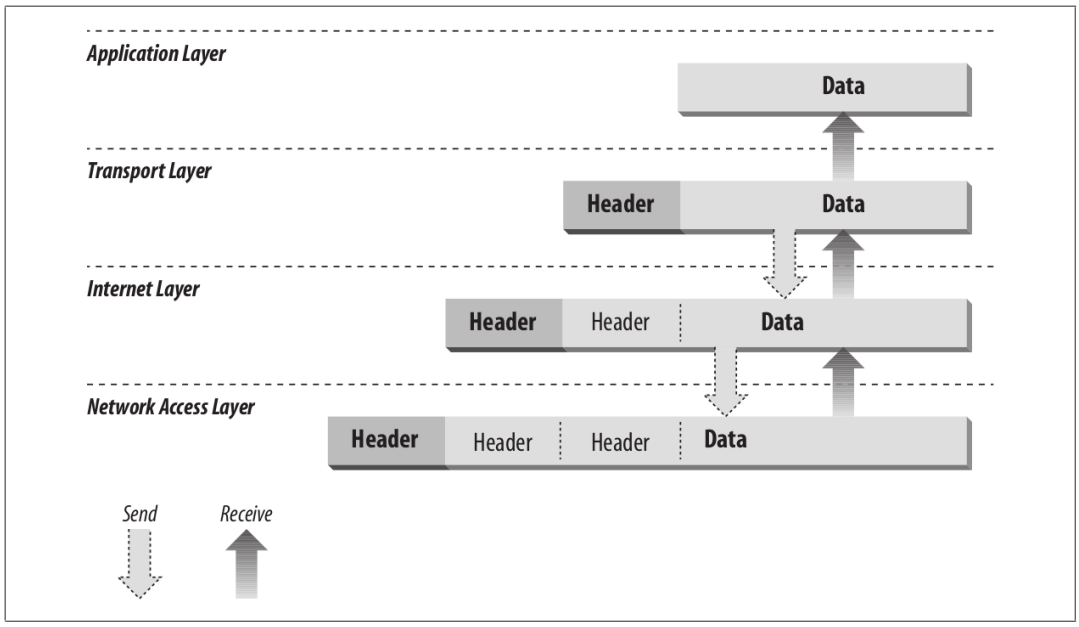

Figura 4 – Encapsulamento de dados do protocolo TCP/IP
Fonte: HUNT. (9)

3.4.1 Camada de acesso à rede

Esta camada corresponde as três camadas mais inferiores do modelo OSI, que são as camadas de rede, enlace de dados e a física. É a responsável por enviar as informações para os outros dispositivos conectados à rede. De acordo com HUNT, (9):

> Esta camada define como usar a rede para transmitir um datagrama IP. Ao contrário dos protocolos de nível superior, os protocolos da camada de acesso à rede devem conhecer os detalhes da rede subjacente (sua estrutura de pacote, endereçamento etc.) para formatar corretamente os dados que estão sendo transmitidos para cumprir as restrições da rede

Percebe-se então que é função desta camada é realizar o encapsulamento de datagramas IP. De acordo com o mesmo autor, outra atividade desta camada é a realização do mapeamento de endereços IP para os endereços físicos usados na rede.

3.4.2 Camada de Internet

Esta camada, segundo HUNT (9), é responsável por definir o datagrama, que é a unidade básica de transmissão da internet, e controlar o roteamento dos dados. O protocolo IP é o elemento central desta camada. Este protocolo é associado de maneira usual com a sua versão IPV4 em redes comercial, entretanto a sua versão IPV6 é considerada um padrão e surgiu por

implementar uma capacidade de endereçamento expandida, pois com a difusão da computação exigiu-se tal necessidade.

HUNT (9), delimita que as funções relacionadas ao protocolo IP são: a definição do datagrama e do esquema de endereçamento da internet, movimentar dados entre as camadas de acesso à rede e a de transporte, roteamento de datagramas para hosts remotos, executar fragmentação e remontagem de datagramas. Este protocolo não é orientado a conexão, isto é, ele não realiza nenhuma abordagem para verificar se o receptor está disponível para receber dados. Dessa forma, para que o TCP/IP seja orientado a conexão, há a dependência de protocolos que fornecem esse tipo de serviço em outras camadas. Outra dependência que esta camada tem em relação as demais é a necessidade de um mecanismo de detecção e recuperação de erros, pois o protocolo IP não possui nenhuma resolução para isto.

A Figura 5 é uma representação do formato de um datagrama IP. As definições relativas aos conteúdos específicos desse datagrama IP está na Tabela 3. Este protocolo também usa este datagrama para enviar suas próprias mensagens que estão relacionadas ao controle. A parte do protocolo relacionado a essas mensagens é o ICMP, *Internet Control Message*. As informações relacionadas ao ICMP se resumem a mensagens de controle de fluxo, detecção de destinos inacessíveis, redirecionamento de rotas e verificação de *hosts* remotos.

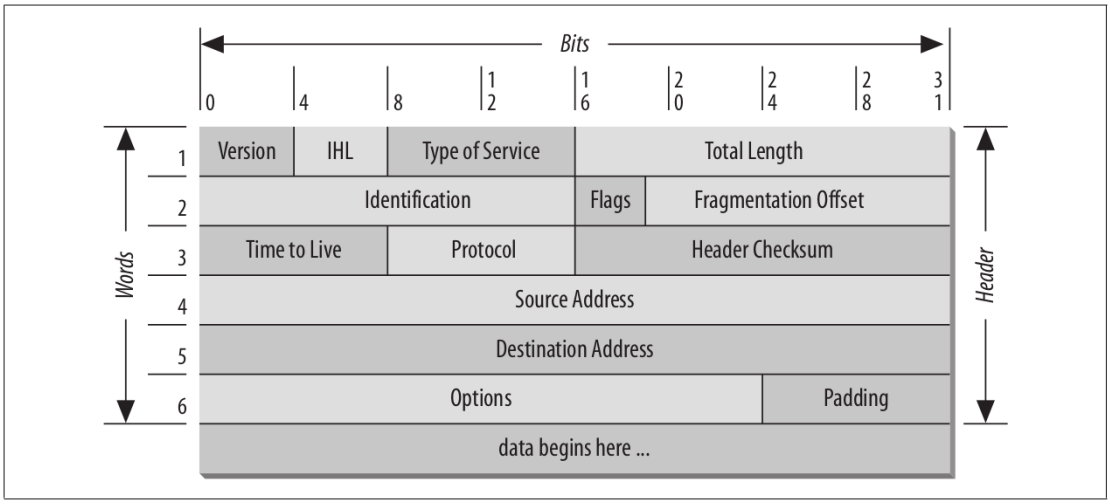

Figura 5 – O formato do datagrama IP
Fonte: HUNT. (9)

Tabela 3 – Descrição dos campos do datagrama IP

Campo	Descrição
Version	Indica o formato da versão do cabeçalho. No caso, esta tabela descreve um cabeçalho IPV4.
IHL	Mostra o comprimento do cabeçalho em palavras. Sendo que a palavra contém 32 bits.
Type of Service	Determina a qualidade de serviço que se deseja.
Total Length	Indica o tamanho do datagrama, cabeçalho e dados, em *bytes*.
Identification	É utilizado para indicar a qual datagrama o fragmento pertence. Este campo é fundamental quando o cabeçalho precisa ser fragmentado por conta da quantidade máximas de dados por unidade de transmissão que a rede dá suporte.
Flags	É um *bit* que indica se há mais fragmentos do datagrama a ser reunido.
Fragmentation Offset	Informa qual é a parte que o fragmento indicado pertence em relação ao todo do datagrama.
Time to Live	Indica o tempo máximo que o cabeçalho deve ser disponibilizado na *internet*.
Protocol	Indica qual é o protocolo da camada de transporte para o qual os dados do datagrama serão passados.
Header Checksum	Este campo é utilizado para realizar a verificação do cabeçalho.
Source Address	Indica o endereço IP do host que está enviando dados.
Destination Address	Indica o endereço IP do host ao qual os dados se destinam.
Options	Fornece informações para a realização de teste e depuração da rede.
Padding	Campo utilizado para a realização de ajuste de tamanho de cabeçalho.
Data	Representada os dados da transmissão.

Fonte: HUNT. (9)

A entrega do datagrama para o *host* de destino ocorre pela interpretação do campo referente ao endereço de destino expresso no mesmo. Caso o host ao qual a informação é enviada esteja na mesma rede de quem o envio, a entrega é imediata. Em situação contrária, os *gateways*, que atuam na comunicação entre redes físicas diferentes, envia o datagrama para a rede que ele interconecta para que ele chegue ao seu host de destino. Sendo que o host de destino pode estar nessa rede interconectada ou há, nessa rede, um outro gateway que seja um caminho viável para o host de destino.

Pode ser que aconteça de as redes interconectadas por gateways tenham características físicas diferentes. Para esses casos, é possível que seja realizada a fragmentação dos datagramas em partes que respeitam o máximo de dados, em bytes, que pode ser enviado por uma

determinada rede, MTU (*Maximum Transmission Unit*). Os campos *Identification*, *Flags* e Fragmentation Offset auxiliam na desfragmentação do datagrama para que não ocorra a perda de dados e para que eles sejam ordenados corretamente, conforme exposto na Tabela 3.

3.4.3 Camada de transporte

A camada de transporte, segundo HUNT (9), é responsável por fornecer um serviço de entrega de dados ponta-a-ponta e fornece dados entre as camadas de Aplicativo e a de Internet. Os protocolos mais conhecidos dessa camada são o UDP (User Datagram Protocol) e o TCP (Transmission Control Protocol).

O protocolo UDP (9), fornece acesso direto a um serviço de entrega de datagramas. As características do protocolo UDP é que ele não é orientado a conexão, ou seja, não estabelece um método de confirmação para saber se o host de destino está pronto para receber dados, e não é confiável, isto é, não estabelece uma maneira para verificar se os dados enviados foram de fato recebidos.

A Figura 6 é uma representação do formato de uma mensagem UDP. As definições relativas ao conteúdo específico dessa mensagem estão na Tabela 4. Este protocolo pode ser recomendado apenas quando a quantidade de dados for pequena, o que torna a possibilidade de ter erros na transmissão reduzida e quando o reenvio de dados ter sobrecarga menor que para a criação de uma conexão que garanta entrega confiável deles.

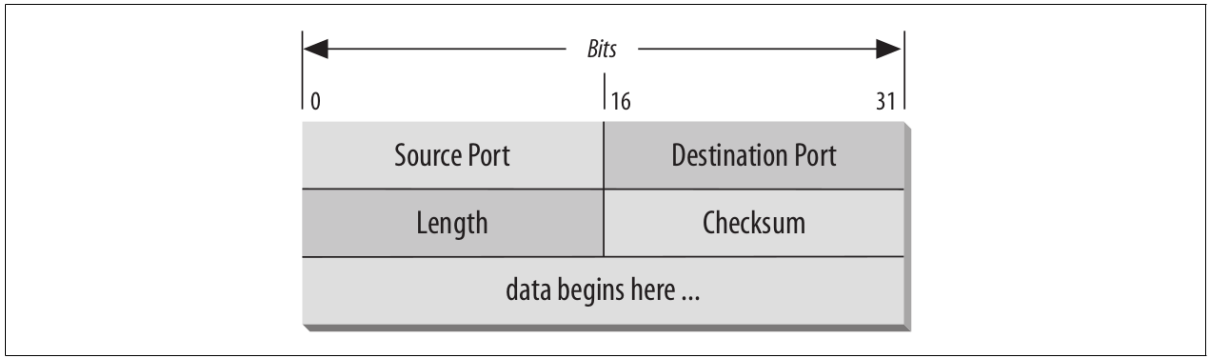

Figura 6 – Formato da mensagem UDP
Fonte: HUNT. (9)

Tabela 4 – Descrição dos campos do cabeçalho UDP

Campo	**Descrição**
Source Port	Indica a porta que está sendo utilizada pelo *host* que está enviando dados.
Destination Port	Indica a porta que está sendo utilizada pelo *host* ao qual os dados se destinam.
Length	Indica o Comprimento da mensagem
Checksum	É utilizado para verificação de erros no cabeçalho ou dados.
Data	Representada os dados da transmissão.

Fonte: HUNT. (9)

O protocolo TCP, (9), é considerado confiável pois ele realiza a verificação se as entregas realizadas estão corretas e se foram recebidas na ordem em que foram enviadas. Uma notação importante para futuras discussões acerca deste protocolo é o nome atribuído para a unidade de dados trocados por módulos TCP, a qual é conhecida por segmento.

Quando um segmento é enviado por um módulo TCP, a verificação da integridade deles é realizada no módulo de destino pela verificação da soma de verificação, que é uma informação que está disponível no segmento. Caso seja verificado que o segmento está íntegro é enviado para o módulo remetente uma confirmação de recebimento. Se não for recebido, por quem enviou o segmento, uma confirmação de recebimento este irá enviar novamente os dados. Na situação em que o destinatário verificou que as informações não estão corretas ele apenas descarta o segmento e aguarda que ele seja enviado de novo.

Este protocolo é orientado à conexão, ou seja, existe um método de verificação do estabelecimento da conexão. No caso do TCP este método é o *handshake* de 3 vias. Este método inicia com módulo remetente ao enviar para o destinatário um segmento com o SYN (*Synchronize Sequence Numbers*), tal procedimento inicial tem o objetivo de estabelecer a conexão e informar o número de sequência que será utilizado como o de início de seus segmentos. O destinatário confirma o recebimento do SYN enviando o ACK (*Acknowledgement*) e o número que será utilizado como inicial para a sequência recebida, isto é, o seu SYN. Repare que cada módulo tem a autonomia de escolher o seu ponto de partida. O remetente confirma o recebimento dessas informações e então retorna com a confirmação de recebimento, ACK, juntamente com os dados iniciais. Para encerrar a conexão um outro *Handshake* de três vias é executado contendo o bit de finalização, FIN, que informa que não há mais dados no remetente para seres enviados.

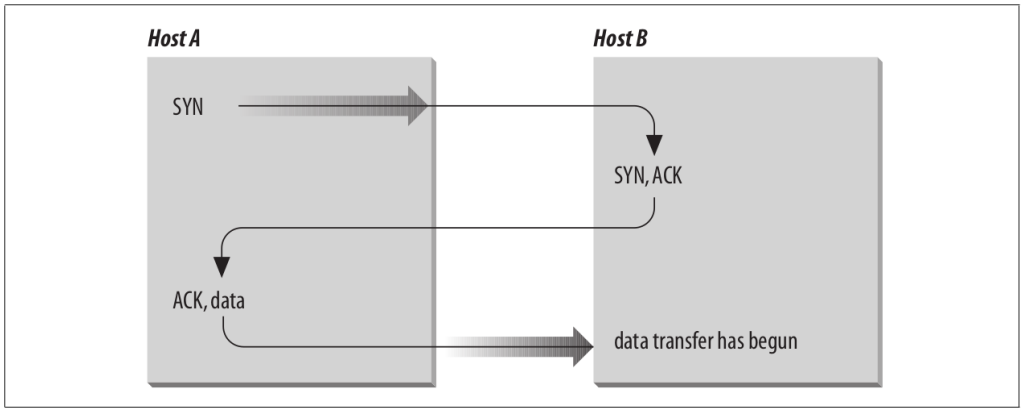

Figura 7 – Three-way Handshake
Fonte: HUNT. (9)

A Figura 8 é uma representação do formato de uma mensagem TCP. As definições relativas ao conteúdo específico dessa mensagem estão na Tabela 5.

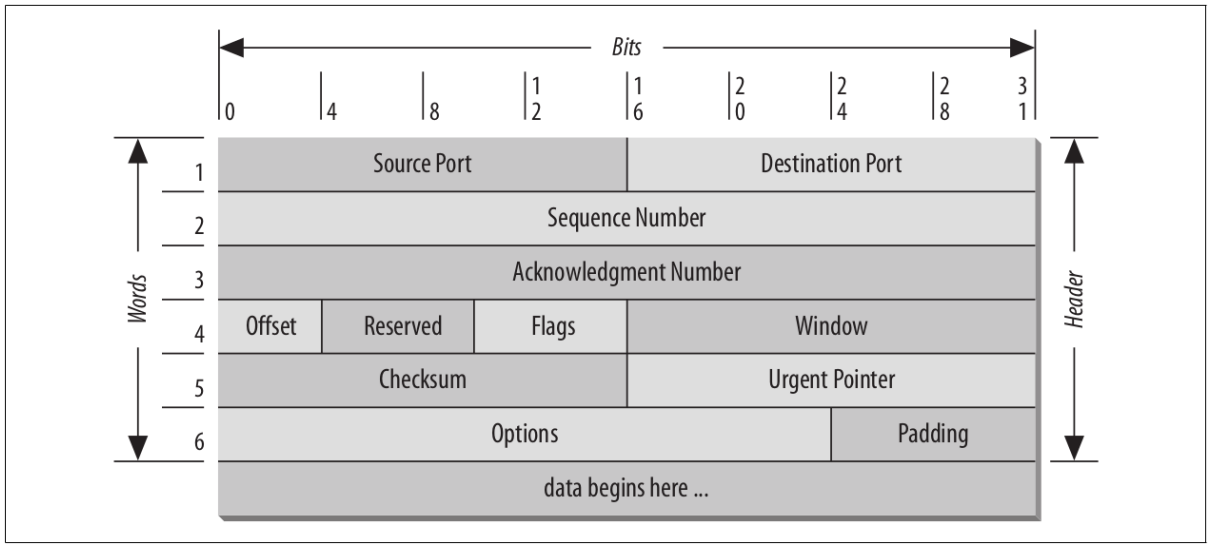

Figura 8 – Formato do cabeçalho TCP
Fonte: HUNT. (9)

Tabela 5 – Descrição dos campos do cabeçalho TCP

Campo	Descrição
Source Port	Informa qual é a porta utilizada pelo host remetente para a realização da comunicação.
Destination Port	Informa qual é a porta utilizada pelo host destinatário para a realização da comunicação.
Sequence Number	Informa o número de sequência do primeiro *byte* de dados no segmento, exceto quando SYN está presente. Quando SYN estiver presente, o número de sequência é o número de sequência inicial (ISN) e o primeiro octeto de dados é ISN + 1.
Acknowledgement Number	Se o bit de controle ACK estiver definido, este campo conterá o valor do próximo número de sequência que o remetente do segmento espera receber.
Offset	Este campo indica onde os dados começam
Reserved	Espaço reservado para uso futuro. Costuma ser zero.
Flags	Representa seis *bits* de controle, que são: URG, ACK, PSH, RST, SYN, FIN
Window	Informa a quantidade de *Bytes* que o *host* de destino dos dados pode aceitar.
Checksum	Este campo é utilizado para realizar a verificação do cabeçalho.
Urgent Pointer	É um ponteiro que aponta para o número de sequência do *byte* após os dados urgentes.
Options	Contempla informações extras que não são especificadas no restante do cabeçalho.
Padding	Campo utilizado para a realização de ajuste de tamanho de cabeçalho.
Data	Representa os dados a serem enviados

Fonte: HUNT. (9)

Os principais campos do formato apresentado para o TCP que dão entendimento sobre a dinâmica de funcionamento do protocolo, mais especificamente o porquê que há segurança em relação a integridade e a ordem de envio, são os campos "*Sequence Number*", "*Acknowledgement Number*" e "*Window*". Segundo HUNT (9), o campo "*Sequence Number*" contém, quando utilizado para enviar o segmento SYN, o ISN (*Initial Sequence Number*), que indica a referência de início no módulo remetente. Nos demais segmentos, ele representa o início dos dados que ele está enviando. O campo "*Acknowledgement Number*", segundo o mesmo autor, executa a confirmação positiva e o controle de fluxo. Ele informa o número de sequência do próximo byte que o receptor tem a expectativa de receber. O campo "*Window*" é fundamental, porque ele informa a quantidade de Bytes que o host de destino dos dados pode aceitar.

Com a interpretação desses campos o host que deseja enviar os dados tem a informação de quantos dados ele pode enviar e quais ele tem a confirmação de recebimento, dessa forma ele vai entregando os dados, se passar um tempo sem confirmação de entrega de algum

segmento, ele o envia novamente. Na parte do destinatário, quando ele receber um pacote que esteja em uma posição que não corresponde a que ele deveria receber para que os dados estejam em ordem, ele o descarta e espera o recebimento dos corretos.

É também obrigação do TCP enviar para o aplicativo correto as informações (9), é por isso que os campos "*Source Port*" e "*Destination Port*" estão presentes no formato do segmento TCP.

3.4.4 Camada de aplicação

Nesta camada, de acordo com Hunt (9), estão presentes todos os processos que fazem o uso de dados advindos da camada de transporte. Está relacionada aos serviços prestados ao usuário, mesmo que ele não tenha a percepção disso. Existem vários protocolos de aplicativos, como por exemplos: Telnet, FTP, SMTP e HTTP.

3.5 Comunicação Serial SPI

O *Serial Peripheral Interface* (SPI) é um protocolo de comunicação serial *duplex* e síncrono entre o dispositivo mestre e escravo. Segundo Leens, (10), "O SPI foi apresentado com o primeiro microcontrolador derivado da mesma arquitetura do popular microprocessador Motorola 68000 anunciado em 1979". O SPI é composto, em resumo, por registradores de controle, dados e status, lógica de deslocamento, gerador de taxa de transmissão, lógica de controle mestre/escravo e lógica para controle das portas. (11)

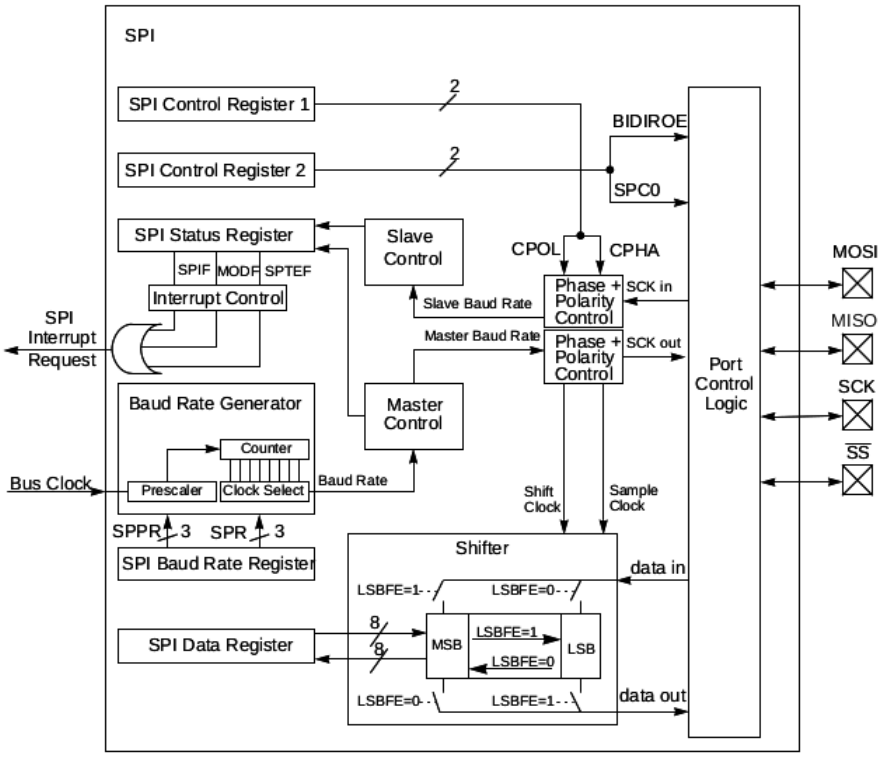

Figura 9 – Diagrama de Blocos do SPI
Fonte: MOTOROLA. (11)

A conexão com um dispositivo que permite uma comunicação pelo protocolo SPI é realizada por meio de quatro portas externas ao dispositivo. São elas: MOSI, MISO, SS e SCK.

A porta MOSI, é utilizada para a transmissão de dados para fora do módulo SPI quando ele é configurado como mestre e recepção de dados quando ele é disposto como escravo. (11) A porta MISO tem função inversa, isto é, o módulo transmite dados quando é atribuído como escravo e recebe dados quando é mestre. (11)

A porta SS também tem funções diferentes caso o dispositivo seja mestre ou escravo. Quando for mestre esta porta seleciona o dispositivo escravo que ela deseja se comunicar, já o módulo SPI escravo utiliza esse pino para saber se a comunicação é direcionada a ele, apenas no caso afirmativo que este irá responder a solicitação do mestre, como especificado por Motorola (11):

> A entrada SS também controla o pino de saída de dados seriais, se SS for alto (não selecionado), o pino de saída de dados seriais é de alta impedância e, se SS for baixo, o primeiro bit no Registro de Dados SPI é expulso dos dados seriais pino de saída. Além disso, se o escravo não for selecionado (SS é alto), a entrada SCK será ignorada e nenhum deslocamento interno do registro de deslocamento SPI ocorrerá.

O pino SCK é utilizado para a sincronização, o *clock*, dos dados enviados e recebidos por este protocolo. No caso, quem envia o sinal de *clock* é o dispositivo caracterizado como

mestre. Existem quatro modos de comunicação que estão relacionados a diferenças na fase e na polaridade do clock e a configuração do modo é definida pelos parâmetros CPOL, relacionado a polaridade, e CPHA, que se refere à fase. (10)

A operação de um dispositivo como mestre se dá quando o bit MSTR do registrador SPI Control Register 1 é definido. Nesse modo, como especificado em Motorola (11), é o mestre que seleciona o dispositivo escravo colocando em nível baixo a porta SS. Ao fazê-lo ocorre o envio de dados para o escravo pela porta MOSI e o recebimento de dados pela porta MISO, sendo que essas transações são sincronizadas com o sinal de clock emitido pela porta SCK.

No escravo ao perceber em sua porta SS um nível digital baixo ele começa a enviar, ação essa sincronizada pelo sinal de clock recebia em SCK, dado para o mestre pela porta MISO e receber pela porta MOSI. A transmissão dos dados entre mestre e escravos são organizados em 8 bits, conforme estipulado por Motorola (11):

> O registro de dados de 8 bits no mestre e o registro de dados de 8 bits no escravo são vinculados pelos pinos MOSI e MISO para formar um registro distribuído de 16 bits. Quando uma operação de transferência de dados é realizada, este registro de 16 bits é deslocado em série nas posições de oito bits pelo S-clock do mestre, de modo que os dados são trocados entre o mestre e o escravo. Os dados gravados no registrador de dados SPI mestre tornam-se os dados de saída para o escravo, e os dados lidos do registrador de dados SPI mestre após uma operação de transferência são os dados de entrada do escravo.

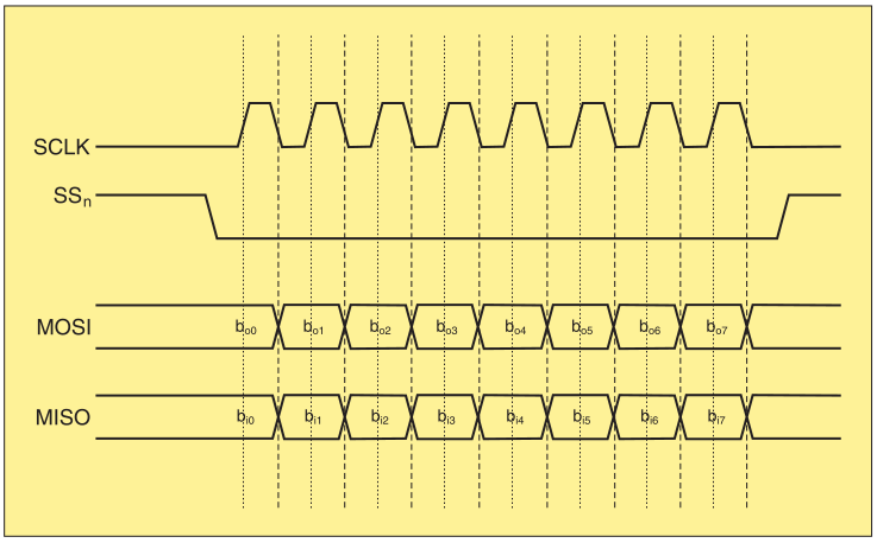

Figura 10 – Representação do diagrama de tempo do SPI
Fonte: LEENS. (10)

Na Figura 11 encontra-se a representação da disposição dos canais de comunicação entre módulos SPI mestres e escravos e o direcionamento dos sinais emitidos por eles.

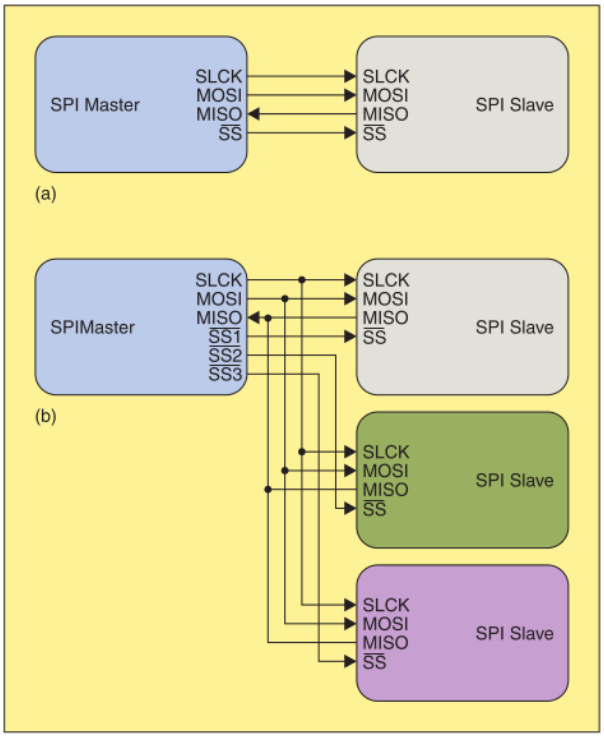

Figura 11 – Representação do SPI considerando (a) um único escravo e (b) vários dispositivos atuando como escravos.

Fonte: LEENS (10)

4 MATERIAIS

4.1 Raspberry Pi

Raspberry Pi são os conhecidos microcomputadores de chip único e de pequenas dimensões. Elas são elaboradas pela Raspberry Pi Foundation (12), que é uma instituição que busca tornar o poder da computação e a produção digital mais acessível.

Neste trabalho foi utilizada a Raspberry Pi modelo B. Nesta versão (13), são encontrados 26 pinos GPIO (General Purpose Input and Output), 2 portas USB 2.0, entrada HDMI, conector Ethernet RJ45, conector de vídeo CSI (*Camera Serial Interface*), conector para display DSI (Display Serial Interface) e leitor de cartão de memória SD.

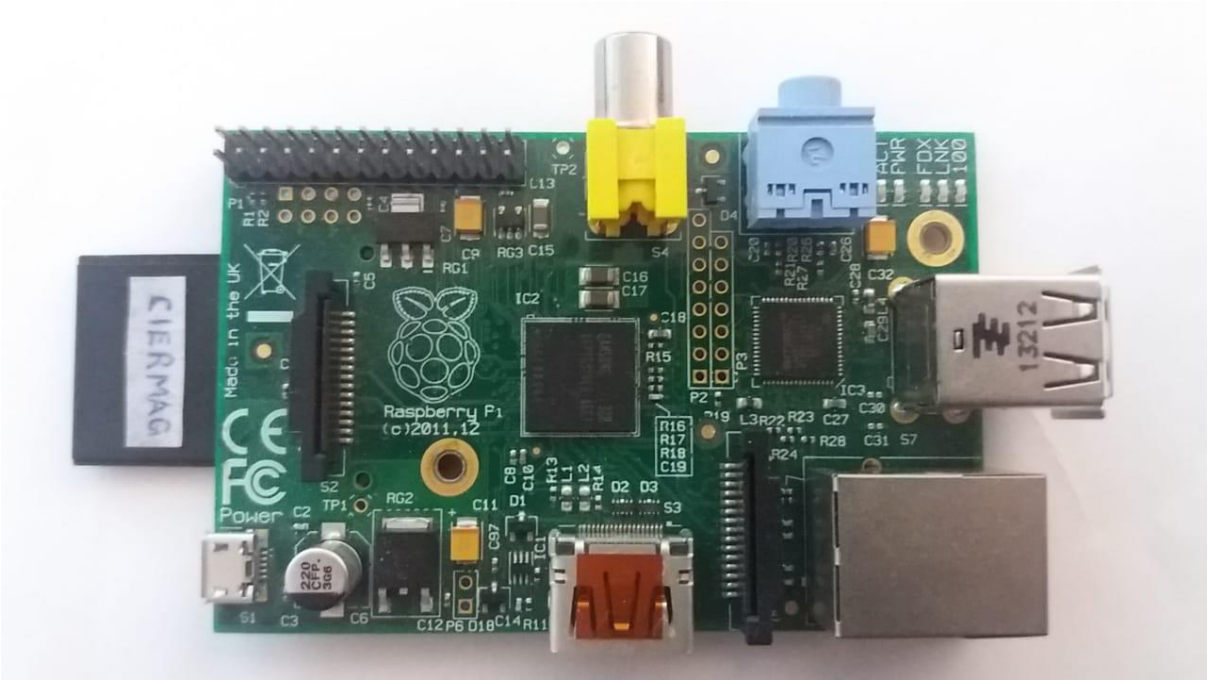

Figura 12 – Raspberry Pi, modelo B
Fonte: Elaborada pelo autor

O processador desse modelo é o Broadcom BCM2835 constituída por CPU e GPU integrados. A arquitetura do processador é a ARM11 e opera a 700 MHz. A memória SDRAM, deste modelo, é um chip separado de 512 MB, montada sobre o processador.

Para operar uma Raspberry Pi, basta instalar um sistema operacional em um cartão SD. Na página web da Raspberry Pi Foundation (14) encontra-se disponível a Raspberry Pi OS, o anteriormente chamado de Raspbian, que é o sistema operacional oficial para todos os modelos Raspberry.

4.2 Python 3

A linguagem de programação Python, (15), possui uma estrutura de dados de alto nível, tem uma abordagem simples e eficaz para orientação à objetos e possui natureza interpretada. A disponibilização gratuita e os novos avanços da linguagem são de responsabilidade da Python Software Foundation. Além dos métodos incorporados na biblioteca padrão, no site da fundação, é possível encontrar indicação de módulos de terceiros e por ser uma linguagem altamente utilizada, o programador Python tem facilidade de encontrar bibliografia apropriada e uma comunidade atuante para auxiliá-lo.

4.2.1 RPi.GPIO

Este módulo contém a classe que representa o GPIO da Raspberry Pi. Assim é possível realizar o controle do GPIO da Raspberry PI através de seus métodos. (16) Nesse módulo não há métodos relativos a implementações SPI, I2C, PWM e implementações relativas a funções seriais. Para cobrir essas necessidades é necessário a implementação dos métodos específicos para cada um deles ou a utilizar método de terceiros. O RPi.GPIO não é recomendado para aplicações que necessita de precisão relativas ao tempo presente.

4.2.2 Spidev

É um módulo, Python, que possibilita, por meio do driver do kernel linux spidev, a interface com dispositivos SPI do espaço de usuário. (17) Dessa forma, ele pode ser empregado para a utilização desse protocolo na Raspberry PI.

4.2.3 Json

É um módulo que permite a documentação e outras interações das informações no formato JSON (JavaScript Object Notation). JSON é um formato de intercâmbio de dados que tem como característica o fato de ser leve. O seu formato é baseado na sintaxe literal do objeto JavaScript. (18)

4.2.4 Socket

Este módulo provê acesso a interface de socket BSD (19), sendo que pode ser aplicado em sistemas operacionais UNIX, Windows, MacOs e outros. Foi desenvolvido para adotar o

paradigma orientado a objetos. Dessa forma a função socket retorna um objeto socket, ao qual os métodos implementam as chamadas características de um sistema de socket.

4.2.5 Time

Este módulo possui diversas implementações de métodos que operam com o tempo. Por este motivo, este módulo é recomendado quando é desejado consultar o horário e data, realizar pausa por um período específico, entre outras funções que podem ser necessárias do ponto de vista do programador. Alguns recursos podem não ser funcionais em todas as plataformas devido ao fato de que algumas funções utilizam métodos em C da plataforma, em que opera, que podem estar implementados de forma diferente, portanto, deve ser consultada as documentações das plataformas destinadas a aplicação antes da utilização desse módulo. (20)

5 DESENVOLVIMENTO

Conforme expresso no capítulo referente a metodologia, para a realização deste trabalho uma oportuna revisão bibliográfica relacionadas a tópicos, teóricos e instrumentais, de interesse para este trabalho, somada ao entendimento do contexto ao qual ele se insere, proporcionou as perspectivas necessárias para dar-se o processo de desenvolvimento do trabalho. A automação do FrontEnd é dependente tanto da programação de métodos, aos quais há grande dependência em relação a sua arquitetura e funcionalidades, como do condicionamento de uma comunicação entre os meios que o controla e que interligam as suas partes. No caso esses meios seriam o módulo Raspberry Pi e possíveis terminais usuários.

O desenvolvimento dessas duas partes se deu de forma paralela, cabendo uma integração entre as partes após as suas finalizações. Por este motivo, a descrição deste capítulo tratará individualmente sobre os desenvolvimentos de cada uma das frentes de trabalho, em seguida será tratada como se obteve a concordância operacional entre elas. Dessa forma, a primeira parte a ser tratada será a criação de uma biblioteca para o controle do FrontEnd, depois será relatado o desenvolvimento da comunicação da Raspberry Pi, com um meio externo, podendo este ser um terminal computador.

5.1 FrontEnd

O desenvolvimento da biblioteca relacionada ao FrontEnd, foi escrita em Python 3. O principal motivo para a escolha dessa linguagem está na possibilidade da construção de software com o a paradigma de programação orientado a objeto. O fato de essa linguagem ser amplamente utilizada e possuir uma comunidade bastante atuante a tornou mais atrativa para o desenvolvimento desse projeto, uma vez que material referente de suporte para implementação com esta linguagem é facilmente encontrado.

Se tratando de orientação a objeto, uma construção de classes referentes a objetos de interesse e a atribuição de característica e métodos a estas se faz necessária. Assim, representou-se, por meio de um diagrama, as interligações das partes de interesse do projeto e a seguir foram definidas quais classes seriam criadas. Neste momento de desenvolvimento, para o desenvolvimento da biblioteca FrontEnd-MR, não se visou a utilização de mais de uma Raspbery Pi, como pode ser observado no referido diagrama, Figura 13.

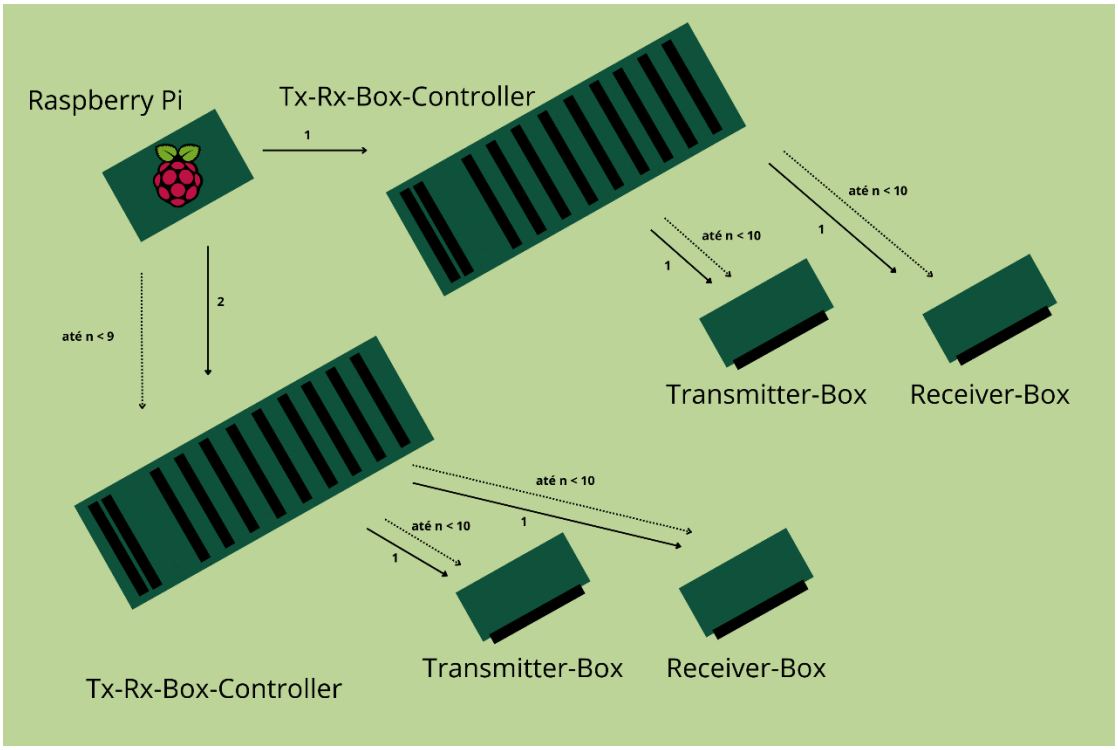

Figura 13 – Representação das conexões físicas do FrontEnd com apenas uma Raspberry Pi.
Fonte: Elaborada pelo autor.

Esta representação deixa claro a variedade de periféricos relacionados ao FrontEnd que a Raspberry Pi pode operar. É possível observar, também, a quantidade de elementos que são parecidos. Diante disso, é imediata a associação de cada um desses elementos a uma classe específica, isto é, com seus próprios atributos e métodos. Ao descrever o FrontEnd no capítulo de revisão bibliográfica, deixou-se claro, que tantos os transmissores como os receptores são formados pelo extensor de pinos de entradas e saídas MCP23S17. Dessa forma a este também foi associada uma classe específica. A expressão dessa associação é apresentada na Tabela 6.

Tabela 6 – Correspondência entre os elementos que compõe o FrontEnd com as suas classes desenvolvidas em Python 3.

Elemento	Classe que o representa
Raspberry Pi	ControllerMotherboard
Tx-RX-Box Controller	TxRxBoxController
Transmitter-Box	TransmitterBox
Receiver-Box	ReceiverBox
Eeprom	Eeprom
MCP23S17	MCP23S17
Memória	TxRxRecord

Fonte: Elaborada pelo autor.

Considerando apenas os elementos centrais que compõem o FrontEnd, a mesma associação pode ser entendida como a relação exibida na Tabela 7.

Tabela 7 – Correspondência entre os elementos centrais que compõe o FrontEnd com as classes desenvolvidas em Python 3.

Elemento central do FrontEnd	Classe que o representa
Raspberry Pi	ControllerMotherboard
	TxRxRecord
Tx-RX-Box Controller	TxRxBoxController
Transmitter-Box	TransmitterBox
	MCP23S17
	Eeprom
Receiver-Box	ReceiverBox
	MCP23S17
	Eeprom

Fonte: Elaborada pelo autor.

Repare, a partir da Tabela 1, que as classes MCP23S17 e EEPROM estão relacionadas tanto aos Transmiter-Box quanto aos Receivers-Box. Cada uma dessas representa um dispositivo eletrônico que está presente nestes dois elementos. Este fato, foi o incentivo para a construção dessas classes, pois tal decisão eliminaria a repetição desnecessária de códigos nas classes dos transmissores e receptores.

A arquitetura que representa a relação entre essas classes desenvolvidas pode ser visualizada na Figura 14. Nos tópicos, a seguir, serão abordados como se deu a construção de cada uma dessas classes. Com o objetivo de oferecer uma melhor didática, serão abordados os

temas mais as extremidades desta arquitetura, até se chegar na classe principal de controle, a ControllerMotherboards. A ordem em que serão realizadas a descrição das classes segue o que está relatado na Tabela 8.

Tabela 8 – Ordem, seguida pelo autor, de explicação das classes correspondentes ao FrontEnd.

Ordem	Classe que o representa
1	MCP23S17
2	EEPROM
3	TransmitterBox
4	ReceiverBox
5	TxRxController
6	TxRxRecord
7	ControllerMotherboards

Fonte: Elaborada pelo autor.

As duas primeiras classes a serem abordadas, correspondem a elementos eletrônicos que estão presentes dentro dos projetos de hardware dos transmissores e dos receptores. Assim podemos dispor eles numa mesma categoria de importância para o projeto. Desta forma, neste trabalho, eles são classificados como classes auxiliares. Os demais métodos serão chamados de classes principais. Eles serão abordados em tópicos distintos, para que a explicação da sua atuação conjunta seja mais didática.

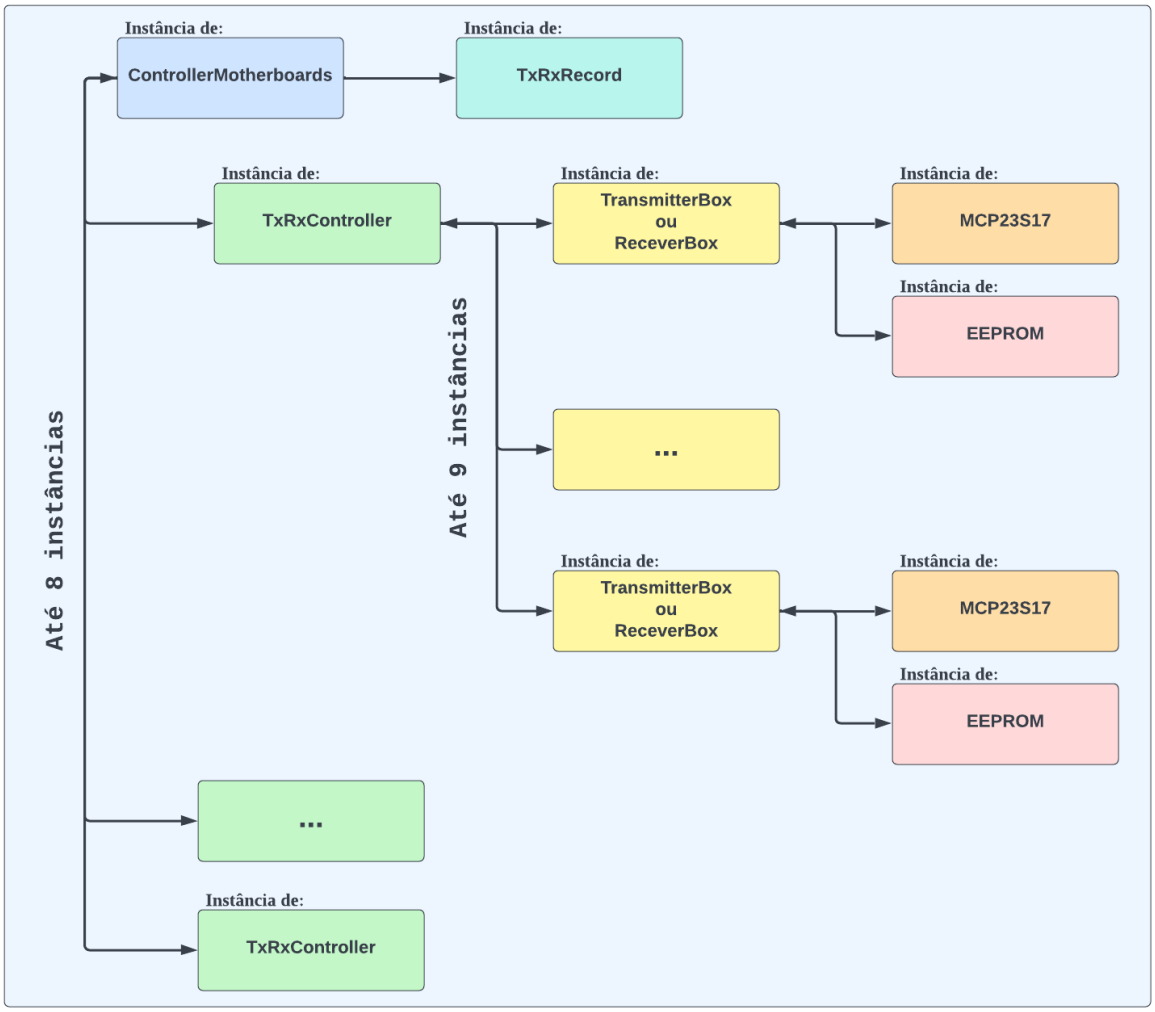

Figura 14 – Representação das conexões físicas do FrontEnd com apenas uma Raspberry Pi.
Fonte: Elaborada pelo autor.

5.1.1 Classes auxiliares

MCP23S17

O MCP23S17 é um expansor de I/Os, isto é, de pinos de inputs (sinais de entrada) e de outputs (sinais de saída). Esta classe é composta pelos seguintes métodos:

- output
- pullup
- iocon
- start_commands
- read_laches

A comunicação com o componente MCP23S17 ocorre por meio da comunicação serial SPI. Desta forma o componente tem uma porta ao qual ele recebe os dados (SDI), uma outra na qual ele envia dados (SDO). O que difere a comunicação desse dispositivo com o protocolo SPI padrão é o fato de a seleção ser realizada de uma maneira diferente. Nele há três pinos de endereçamento, que são representados, em termos de software com 3 bits. Se este componente estiver com as suas portas de endereço alimentadas com 5 volts, o seu endereço será o número 7, representado por 111 em binário. Caso esteja com apenas a porta do meio alimentada o seu endereço corresponderá, em binário, ao número 010, que significa 2 no sistema de numeração decimal.

Para se comunicar com o dispositivo, basta serem enviadas três informações em sequência. São elas: o endereço do módulo MCP23S17, o registrador ao qual a comunicação se destina e, por fim, o valor a ser atribuído para este registrador. Assim, para cada um dos métodos elaborados para este dispositivo, a exceção do start_comands, consiste na montagem de frames com estas três informações.

Para iniciar a comunicação com este dispositivo, deve ser realizado um reset nos dispositivos. Isto é, deve ser variada a entrada com um pulso em nível baixo. Após isso, são enviados os frames de configuração do dispositivo para que ele opere da forma como queremos no projeto. Os métodos de configuração adicionados até o momento nesta classe são: output, pullup e iocon. O MCP23S17 tem 22 registradores que podem ser configurados para que o dispositivo atue de uma forma única. Para este projeto foram configurados apenas os IODIRA, GPPUA e IOCON que estão associados aos métodos de configuração desenvolvidos para este projeto. Isso significa que foi necessário modificar o valor padrão destes. Entenda como valor padrão aquele que o dispositivo assume quando for resetado. Os valores padrões de cada registrador podem ser consultados no datasheet do MCP23S17.

O método output é responsável por definir os pinos associados a porta A do MCP23S17 como saída de sinais (informação). Para tanto ele faz o uso de registrador IODIRA e envia o valor 0 para cada um dos bits associados a esta porta.

Já o pullup realiza o controle do registrador GPPU. Cada bit desse registrador, está associado a um pino específico. Quando este está configurado como input, recebedor de sinais, e algum bit do registrador for indicado como 1 quer dizer que um resistor de 100 kΩ, interno ao MCP23S17, é associado ao pino relacionado.

E, por fim, o método iocon atua na configuração do registrador IOCON. Cada bit deste registrador tem uma função específica. O valor padrão dos bits deste registrador é zero. No caso deste projeto, viu-se a necessidade de setar o bit 3, isto é, colocar ele para corresponder ao nível alto, indicado por 1. Este bit, quando em 1, habilita os pinos de endereçamento do MCP23S17. Tal decisão foi tomada, pois o hardware desenvolvido para este projeto prevê o endereçamento dos transmissores e receptores por meio do componente MCP23S17.

O método start_comands foi criado para simplificar a chamada dos métodos de configuração mencionados anteriormente. Assim, basta o usuário chamar ele, caso deseje que todos eles sejam chamados.

O último método da lista, o readlaches, é utilizado para a leitura do nível das portas que são destinadas a entradas de sinal no dispositivo MCP23S17.

EEPROM

Esta classe corresponde ao dispositivo AT25010, que corresponde a um elemento de memória do tipo EEPROM. A comunicação com este dispositivo é realizada por meio do protocolo de comunicação serial SPI. Os métodos associados a esta classe correspondem a apenas 4 métodos. São eles:

- ready_eeprom
- write_eeprom
- interpret_ready
- msg_write

Os dois primeiros métodos são, respectivamente, os de leitura e de escrita das informações na EEPROM. O método interpret_ready é utilizado para interpretar o que foi lido da EEPROM. E, por fim, o msg_write é utilizado par preparar a mensagem a ser gravada na EEPROM.

A utilização da EEPROM neste projeto é informar para o usuário, que o opera, qual é a faixa de frequência que o transmitir ou o receptor opera.

Os participantes da comunicação com a EEPROM são três. O primeiro deles é a Raspberry PI que recebe a solicitação, por um terminal computador, de programar a informação ou mesmo a de ler a informação da frequência da operação de um transmissor ou receptor. O dispositivo presente nos transmissores e receptores que faz o papel de interface para a comunicação com a

EEPROM é o MCP23S17. E o último participante é a própria EEPROM. Sendo o MCP23S17 o elemento intermediário, a comunicação entre a EEPROM e o usuário que controla a Raspberry Pi ocorre pela manipulação dos pinos que estão ligados diretamente a EEPROM. Estes pinos estão localizados na porta B do MCP23S17.

5.1.2 Classes Principais

Antes de iniciarmos a explicação das classes principais, é conveniente dar uma explicação de como que o terminal console espera que seja dado o funcionamento do controle dos periféricos lentos do espectrômetro de ressonância magnética, para que o leitor se contextualize sobre o porquê das tomadas de decisão do desenvolvimento deste trabalho.

O FrontEnd é formado por uma placa principal, denominada Tx-Rx Box Controller, podendo ser adicionadas mais 7 placas auxiliares do mesmo tipo. Em cada uma dessas placas é possível a conexão de circuitos projetados para filtrar sinais de transmissão, Transmitter Box, e de recepção, Receiver Box, advindos do espectrômetro. O Tx-Rx Box Controller permite a conexão de 9 placas de transmissão ou de recepção, não importando a distribuição das mesmas e a proporção entre os tipos (receptor ou transmissor), sendo que oito são endereçadas de zero até sete e a nona placa, dita auxiliar, é endereçada como zero, ela acessada por um duto distinto dos oito anteriores.

Dessa forma, o FrontEnd, conectado a uma Raspberry Pi, pode conter o número limite de setenta e dois circuitos de transmissão ou recepção de sinais, distribuídos em 8 placas Tx-Rx Box Controller, sendo que em cada umas dessas há a possibilita de conexão de 9 circuitos de transmissão, 8 localizadas no duto principal e 1 no auxiliar. O sistema de controle deve percorrer cada uma das Tx-Rx Box Controller, cada uma delas é endereçada com um valor único de zero até 7, e identificar os tipos de filtros, receptores ou transmissores, conectados a elas. Como uma medida de segurança o sistema de controle, ao ser ligado, deve programar em todos os receptores ou transmissores a atenuação mais baixa possível, de acordo com as suas características.

O espectrômetro desenvolvido pelo CIERMag possui quatro canais de comunicação do tipo transmissão e 4 do tipo recepção. Dessa forma é um requisito para o projeto que sejam feitas as seleções dos canais de transmissão ou de recepção de acordo com a quantidade de cada um desses tipos em uma Tx-Rx-Box-Controller, no caso dos filtros do duto principal. As

atribuições de qual canal gates deve ser selecionado deve seguir a regra estabelecida pelas Tabelas 9 e 10.

Tabela 9 – Atribuições dos canais do Espectrômetro de Ressonância Magnética referente aos transmissores.

Número do Transmitter-Box	Canal selecionado
1	TXGATE1
2	TXGATE2
3	TXGATE3
4	TXGATE4
5 ou mais	TXGATE1

Fonte: Elaborada pelo autor.

Tabela 10 – Atribuições dos canais do Espectrômetro de Ressonância Magnética referente aos Receptores.

Número do Receiver-Box	Canal selecionado
1	RXGATE1
2	RXGATE2
3	RXGATE3
4	RXGATE4
5 ou mais	RXGATE1

Fonte: Elaborada pelo autor.

TransmitterBox

A classe TransmitterBox foi desenvolvida para o controle do transmissor, Transmitter-Box, desenvolvido pelo grupo de pesquisa CIERMag. O referido dispositivo permite a programação de atenuações dos sinais por eles transmitidos de até 63 dB em passos de 0,5 dB. Tal variação só é possível pois como parte integrante deste dispositivo são encontrados dois atenuadores digitais, SKY12347. Os métodos associados ao transmissor são apenas dois. São eles:

- informs_attenuation_Tx
- interpret_attenuation

O primeiro deles é o responsável por programar a atenuação do transmissor. Este método basicamente envia para o MCP23S17 os sinais que os pinos da sua porta B devem assumir ao longo do tempo para que eles sejam programados. A comunicação entre o SKY12347 e o

MCP23S17 ocorre por meio da comunicação serial SPI em que o pino que corresponde ao SDI varia de acordo com um sinal de *clock* que é gerado por um pino vizinho. Ainda na porta B, podem ser encontrados os pinos de seleção que indica, quando em nível alto, qual o atenuador digital está sendo programado. O sinal de saída do SKY, que corresponde ao SDO no protocolo SPI é encontrado na porta A do MCP23S17, afinal ela está programada como *input*, entrada de sinais. Este sinal é a repetição dos dados enviados para os SKY12347, defasados por 6 ciclos de *clock*.

O Método interpret_attenuation é um método que converte a atenuação solicitada por um usuário em uma lista de binários que correspondem ao valor dos bits que devem ser enviadas aos SKY12347, para que este seja programado com a atenuação informada. Os valores desses bits são calculados por este mesmo método.

ReceiverBox

A classe ReceiverBox foi desenvolvida para que se possa realizar o controle dos receptores do FrontEnd do espectrômetro de ressonância magnética desenvolvido pelo CIERMag. A programação associada aos receptores tem relação direta com o ganho que eles irão fornecer aos sinais por ele recebidos. No circuito deste, existem dois tipos de elementos que podem ser programados para adicionar ganho ao receptor.

O primeiro deles são amplificadores operacionais que podem ou não serem utilizados para atenuar o circuito e o segundo é o VGA - *Variable Gain Amplifier*. O primeiro é programado por meio de dois pinos da porta A do MCP23S17 que, conjuntamente podem assumir as configurações binárias "01" e "10", sendo que a primeira permite a participação do amplificador na composição do sinal do receptor e a segunda significa o oposto.

A comunicação com o VGA é realizada por meio do protocolo SPI, que assim como aconteceu com o SKY12347 pertencentes aos transmissores, a comunicação será realizada por intermédio do MCP23S17. Este dispositivo possui duas faixas de ganho que podem ser selecionadas e estas coincidem em uma parte de suas faixas. A variação de ganho almejada pelo CIERMag é contemplada pelas duas faixas caso estas sejam usadas em conjunto. Essas faixas são caracterizadas pelo datasheet como modos de ganho, no qual o modo *Low Gain* possui menor distorção do sinal, entretanto o *High Gain* possui maior estabilidade do sinal. Toda a configuração e seleção das faixas de ganhos seguem as diretrizes indicadas pelo datasheet.

Esta classe é composta pelos seguintes métodos:

- informs_attenuation_Rx
- interpret_gain_with_MSB
- select_msb
- inform_gain_vga
- VGA_controller
- recebe_ganho

O método recebe_ganho, recebe o valor de ganho, em dB, solicitado pelo usuário e o converte para V/V. O método MSB é utilizado para a seleção do tipo de ganho a ser utilizado, se vai ser *Low Gain* (ganho baixo) ou *High Gain* (ganho alto).

O método VGA_Controller organiza os demais métodos do circuito relacionados ao VGA para que ocorra a programação do mesmo em conjunto com a seleção do aplificador operacional. Este método chama os métodos interpret_gain_with_MSB e o inform_gain_vga. O primeiro deles realiza de fato a programação do ganho solicitado para o receptor. Já o segundo monta uma lista de valores binários, incluindo o MSB - seleção do modo de ganho (high gain ou low gain), que será utilizada para a programação dos ganhos por intermédio do MCP23S17.

O método informs_attenuation_Rx é ulizado apenas para alterar a atuação do amplificador operacional no receptor. Isto é, se a sua atuação irá ou não acrescentar ao sinal passado pelo transmissor.

TxRxController

Esta classe representa a placa Tx-Rx-Controller. A função desta placa é fazer a inteface entre a Raspberry Pi e um conjunto de transmissores e receptores. Assim, cabem como métodos dessa classe aqueles que objetivam a identificação das placas conectadas e o gerenciamento delas como conjunto. Os métodos principais associados a esta classe são:

- select_motherboard
- interprets_motherboard_address
- circuit_identification_routines
- circuit_identification
- start_MCP23S17

- fabricate_ABs_Rx
- fabricate_ABs_Tx
- initial_attenuation_decision
- initial_attenuation_decision_slot_AUX

O método select_motherboard é utilizado quando se deseja operar com a placa instanciada. Quando a classe é instanciada deve ser fornecido o seu endereço como parâmetro. Assim, ao chamar o método select_motherboard em uma instância desta classe, os pinos da Raspberry que são utilizados para o endereçamento das placas deste tipo assumirão os valores, em binário, da Tx-Rx-Controler que representa a tal instância. Este método atua em conjunto com o seu auxiliar, interprets_motherboard_address, este transforma o valor do endereço em decimal em uma lista de binários que corresponde aos valores dos pinos da Raspberry Pi que são utilizados para selecionar as Tx-Rx-Controller.

O método circuit_identification_routines realiza os procedimentos para identificar se há placas conectadas nos slots da Tx-Rx-Controller e, caso tenha, identificar se é um circuito transmissor ou receptor. Este método pede a leitura dos pinos de cada slot correspondentes a portas dos MCP23S17 que atua como saída de dados e envia o resultado para o seu método auxiliar. O método circuit_identification é o que atua em conjunto com o anterior, é ele que de fato faz a distinção do tipo de circuito conectado. Caso a leitura dos pinos corresponda a 2, 11 em binário, o circuito será identificado como um receptor, caso seja 1, 01 em binário, ele será identificado com um transmissor. Qualquer valor diferente disso o slot será interpretado como sem circuito.

O start_MCP23S17 chama os métodos de inicialização do MCP23S17 para que possa ser iniciada a comunicação com os receptores e transmissores. Este dispositivo está presente nestes dois tipos de circuitos, fato este que faz ser conveniente a sua implementação nesta classe.

Os métodos fabricate_ABs_Rx e fabricate_ABs_Tx são os responsáveis por atribuir aos circuitos conectados o canal transmissor ou receptor adequado. A atribuição do canal segue a regra apresentada nas tabelas 8 e 9. A primeira se refere aos transmissores e a segunda aos receptores.

Por fim, os métodos initial_attenuation_decision e initial_attenuation_decision_slot_AUX são responsáveis por garantir a programação dos transmissores e receptores quando o módulo

Raspberry é ligado. Eles atribuem para os receptores o menor ganho possível e aos transmissores, a menor atenuação possível.

TxRxRecord

Esta classe representa a memória do sistema de hardware que compõe o FrontEnd. Este projeto deve ser aplicado em experimentos com o espectrômetro de ressonância magnética. Esses experimentos podem ocorrer por vários dias ou em dias diferentes. Este fato, mostra a importância de se ter uma forma de verificar se houve alguma alteração na composição dos transmissores e receptores que compõe o projeto. Pois caso tenha alguma ocorrência de alteração haverá alterações no resultado esperado pelos operadores.

Como os transmissores e receptores são endereçados é possível identificar as suas posições relativas. Assim a solução para esta demanda é o armazenamento das informações referente as disposições dos transmissores e receptores em um arquivo json.

Toda vez que for utilizado o controlador do FrontEnd ele irá verificar se este arquivo possui a mesma descrição da disposição dos elementos que o compõe em relação com a identificada ao ligar o controlador do FrontEnd, a Raspberry Pi. Caso seja identificada uma mudança a Raspberry fornecerá um relatório com as mudanças identificadas e usuário deverá decidir se mantém o sistema com as mudanças identificadas.

Os métodos pertencentes a esta classe são:

- structure_the_memory
- write_json_file
- compare_with_memory
- map_difference
- interpret_errors

O método structure_the_memory monta a estrutura do arquivo json que conterá a memória de hardware. O método write_json_file, como o próprio nome diz, realiza a escrita da composição do hardware do FrontEnd no arquivo json. O compare_with_memory compara a informação recebida com a contida na memória. A função map_differença mapeia as mudanças que aconteceram entre a nova leitura de composição de hardware e a registrada na memória. Com os dados obtidos com o mapeamento das diferenças o método interpret_errors informa as diferenças entre o status atual e o armazenado em memória.

ControlerMotherboard

Esta classe representa a atuação da Raspberry Pi para o controle das demais partes associadas ao FrontEnd. Sua atuação está diretamente relacionada aos dispositivos nomeados como TX-RX-Box Controller, como mostra a Figura 15.

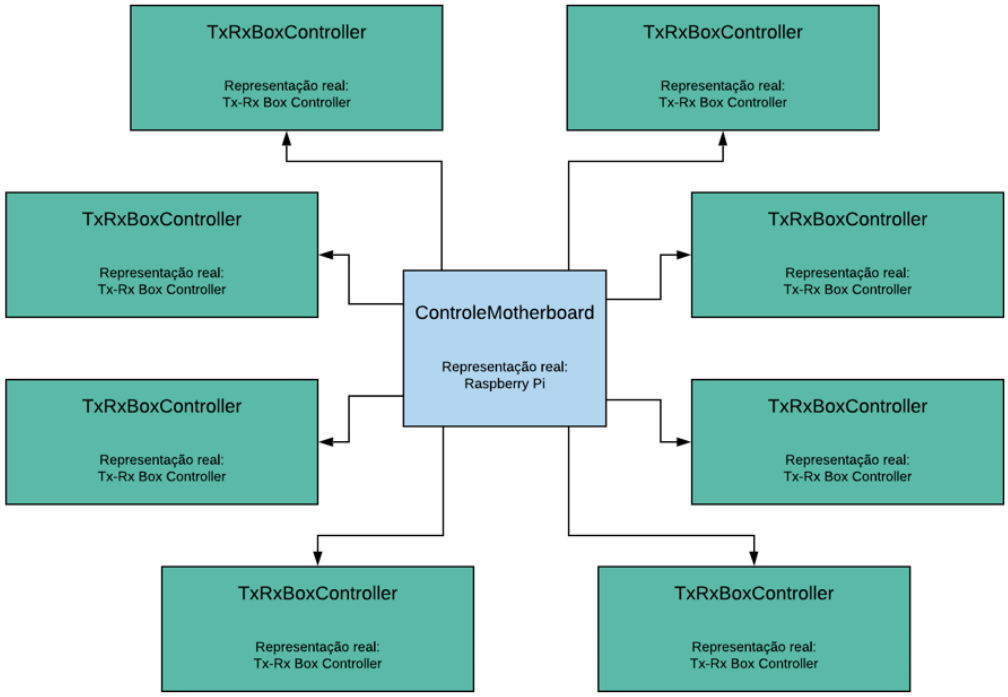

Figura 15 - Esquema da atuação direta da classe assoviada a Raspberry Pi e a classe TxRxBoxController de forma a especificar a representação da disposição dos mesmos no FrontEnd do espectrômetro de RMN do CIERMag.

Fonte: Elaborada pelo autor

Percebe-se, pela Figura 18, que há um número predeterminado de TxRxBoxControler que esta classe pode instanciar. Assim essa classe limita o número de vezes que esta pode ser chamada a 8. Apesar de não ter relação direta com os transmissores e receptores, a comunicação entre esses e o equipamento de controle ocorrem por meio do TxRxBoxController, afinal a função dele é permitir a programação e o uso de 9 transmissores ou receptores. Assim, os seus métodos estão relacionados a obter a lista de dispositivos conectados a cada uma das TxRxBoxController, gerenciar a comunicação com cada um deles e atribuir o estado inicial de operação de cada um deles. Os principais métodos dessa classe foram divididos em métodos operacionais e métodos de usuário. Os métodos operacionais são:

- reset_MCP23S17
- start_spi_0

- start_spi_1
- orders
- get_list_for_operations
- calls_initial_attenuation
- find_and_make_changes
- how_to_change_attenuation
- method_selected
- interpret_method
- main_controller

A função de resetar dispositivos conectados ao objeto de controle, no caso a Raspberry Pi, é destinado ao método reset_MCP23S17. O reset se dá alterando o estado lógico para um nível baixo por um pequeno período. Isso é realizado com a identificação do pino 22 como um pino em que recebe uma programação para a saída. Em seguida altera saída com um nível alto, baixo e alto nessa sequência. Esse processo é suficiente para realizar o reset.

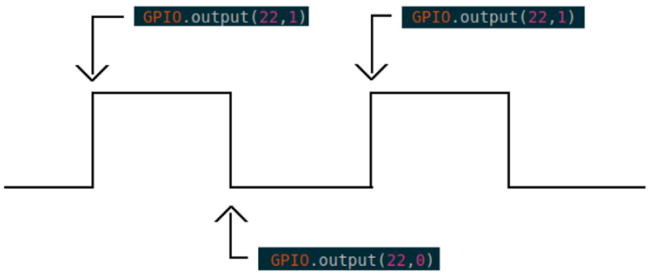

Figura 16 - Pulso de reset programado no pino 22 da Raspberry Pi.
Fonte: Elaborada pelo autor

Conforme mostrado no capítulo de materiais, a Raspberry Pi possui dos canais de *slave select* para o protocolo de comunicação serial SPI. Neste projeto será utilizado os dois. O método start_spi_0 seleciona inicia o protocolo de comunicação SPI que tem o pino 24 da Raspberry Pi como o canal de seleção. Este pino está associado a comunicação com os 8 slots principais da placa que ela está associada.

Já o método start_spi_1 seleciona inicia o protocolo de comunicação SPI tendo o pino 26 da Raspberry Pi como o canal de seleção. Ele é dedicado para a comunicação do slot auxiliar que as placas Tx-Rx-Box Controller possuem.

O método find_and_make_changes é utilizado para o usuário informar qual é o Tx-Rx-Box-Controller e o slot deste com o qual ele deseja trabalhar para encaminhar para os métodos necessários para que as mudanças de configuração possam ser realizadas.

O get_list_for_operations é o método que acumula as informações necessárias para a operação do sistema. Informações como o tipo e a seleção do canal são obtidas por esse método. Enquanto que o método calls_initial_attenuation identifica e atribui atenuação baixa a todos os receptores e atenuação de 0 dB a todos os transmissores.

O método how_to_change_attenuation é utilizado para selecionar o slot e a mudança em atenuação ou o tipo de ganho que ele deseja executar, em transmissores ou receptores.

A diferença dos métodos para usuários em relação aos operacionais é que os do usuário podem ser selecionados por meio de *strings*, tipo de variável em linguagem de programação que representam uma combinação de caracteres, que são interpretadas pelo método interpret_method. Este método decodifica essa *string* para identificar o método solicitado e as variáveis de entrada, caso houver, e chama o método method_selected. Esse, por meio de uma estrutura de condicionais, encaminha as variáveis para o método solicitado realizar a sua tarefa.

A Tabela 11 contêm o nome dos métodos para usuários desenvolvidos com as suas respectivas descrições.

Tabela 11 – Relação dos métodos para usuário desenvolvidos acompanhados com as suas descrições.

Método	**Descrição**
how_many	Informa para o usuário quantos transmissores e receptores estão sendo utilizados no sistema.
receiver	Este método recebe o número de identificação do receptor e o ganho que deve ser configurado (alto ou baixo). Esta configuração está associa a inclusão ou não da participação de um amplificador operacional no sinal que este dispositivo recebe.
receiver2high	Este método recebe o número de identificação de um receptor e atribui a ele a configuração de ganho alto.
receiver2low	Este método recebe o número de identificação de um receptor e atribui a ele a configuração de ganho baixo.
vgareceiver	Este método recebe o valor de ganho do receptor (alto ou baixo) referente a atuação ou não de um amplificador operacional, o ganho a ser configurado no VGA presente no dispositivo do receptor e a identificação do receptor que deve realizar essas modificações. Este método realiza as configurações informadas.

Fonte: Elaborada pelo autor

Tabela 12 – Relação dos métodos para usuário desenvolvidos acompanhados com as suas descrições.

Método	Descrição
high_receiver	Este método é utilizado para informar quantos receptores estão com ganho alto.
low_receiver	Este método é utilizado para informar quantos receptores estão com ganho baixo.
transmitter	Este método recebe o número de identificação do transmissor e o ganho que deve ser configurado. Este ganho pode variar de 0 até 63 dB, em degraus de 0,5 dB.
all_receiver	Este método recebe o tipo de ganho (alto ou baixo) e atribui a todos os receptores do sistema.
all_transmitter	Este método recebe o tipo de ganho (alto ou baixo) e atribui a todos os transmissores do sistema.
reset_receiver	Este método recebe o índice de identificação de um receptor. Ele retorna este receptor ao seu estado inicial de configuração.
reset_transmitter	Este método recebe o índice de identificação de um transmissor. Ele retorna este transmissor ao seu estado inicial de configuração.
reset_all_receiver	Este método retorna todos os receptores para o seu estado de configuração inicial.
reset_all_transmitter	Este método retorna todos os transmissores para o seu estado de configuração inicial.
att_transmitter	Este método recebe um valor de atenuação e retorna uma lista com os transmissores que tem este valor de atenuação.
gain_state_receivers	Este método retorna uma lista com os ganhos dos receptores.
att_state_transmitters	Este método retorna uma lista com as atenuações dos transmissores.
state_all	Este método retorna uma lista com os ganhos dos receptores e as atenuações dos transmissores.
start	Este método habilita a utilização de todos os métodos direcionados para o usuário.
disable	Este método desabilita a utilização de todos os métodos direcionados para o usuário.
reset_all	Este método retorna todos os dispositivos conectados, transmissores e receptores, para os seus estados iniciais de configuração.

Fonte: Elaborada pelo autor.

5.2 Socket de comunicação com a Raspberry

O controle do FrontEnd foi desenvolvido para acontecer por intermédio da Raspberry Pi. Desta forma, quando um usuário for utilizar o recurso desenvolvido para operar o FrontEnd, ele deve se comunicar com ela. Por este motivo, foi construída uma classe em Python 3 para gerenciar esta comunicação e um socket para realizar a comunicação entre este módulo embarcado e um terminal externo. Neste tópico será explicado sobre como foi realizado o desenvolvimento do socket e dessa classe e como eles se relacionam com controle do FrontEnd executado pela Raspberry Pi.

O desenvolvimento da comunicação foi realizado com a utilização da biblioteca "socket". Assim foi construído um canal de comunicação entre a Raspberry e o terminal ou programa que executar o *script* que cria o socket no terminal.

O *socket* desenvolvido para atuar na Raspberry é criado depois que ela iniciar a sua rotina de inicialização. Ao ser ligada, ela atribui aos transmissores o menor valor de atenuação e aos receptores o menor ganho possível, em seguida ela verifica se o *hardware* do FrontEnd continua com a mesma disposição. Ou seja, se os transmissores e os receptores estão nos mesmos slots, se tem a mesma quantidade e identificar quais foram as diferenças encontradas. Após esse processo é criado o *socket*. Esse espera a conexão dele por um outro *host*. Ao identificar a conexão ele envia para o par que se conectou a ele se houve mudanças no hardware e espera a resposta deste para saber se ele pode continuar operando. Caso venha a resposta de continuidade, a disposição atual do hardware é gravada e em seguida o programa entra em um loop em que ele espera uma solicitação do *host* que pareou com ele, chama o método main_controller da instância da classe "ControlerMotherboard" e por fim envia a resposta obtida ne execução da solicitação realizada. O método main_controller interpreta o método solicitado, chama a execução deste e retorna com a resposta de execução dele.

O host externo ao se conectar com a Rapberry, recebe uma mensagem que informa se há mudanças de hardware na Raspberry desde a última vez que ela foi utilizada. Caso a informação interpretada seja de que não houve mudanças, este socket envia a resposta "True" para que a Raspberry comece a aceitar novas solicitações. Na situação contrária, o usuário é informado que há mudanças na disposição do hardware do FrontEnd, ao qual ele deve informar se ele deseja continuar operando. Ao optar pela opção de continuar o socket entrará num loop de comunicação com a Raspberry. Neste loop é esperado que o usuário digite a sua solicitação,

esta é enviada para a Raspberry, em seguida espera-se a resposta da Raspberry, ao chegar a resposta é chamado um método que interpreta a resposta e retorna esta para o usuário final.

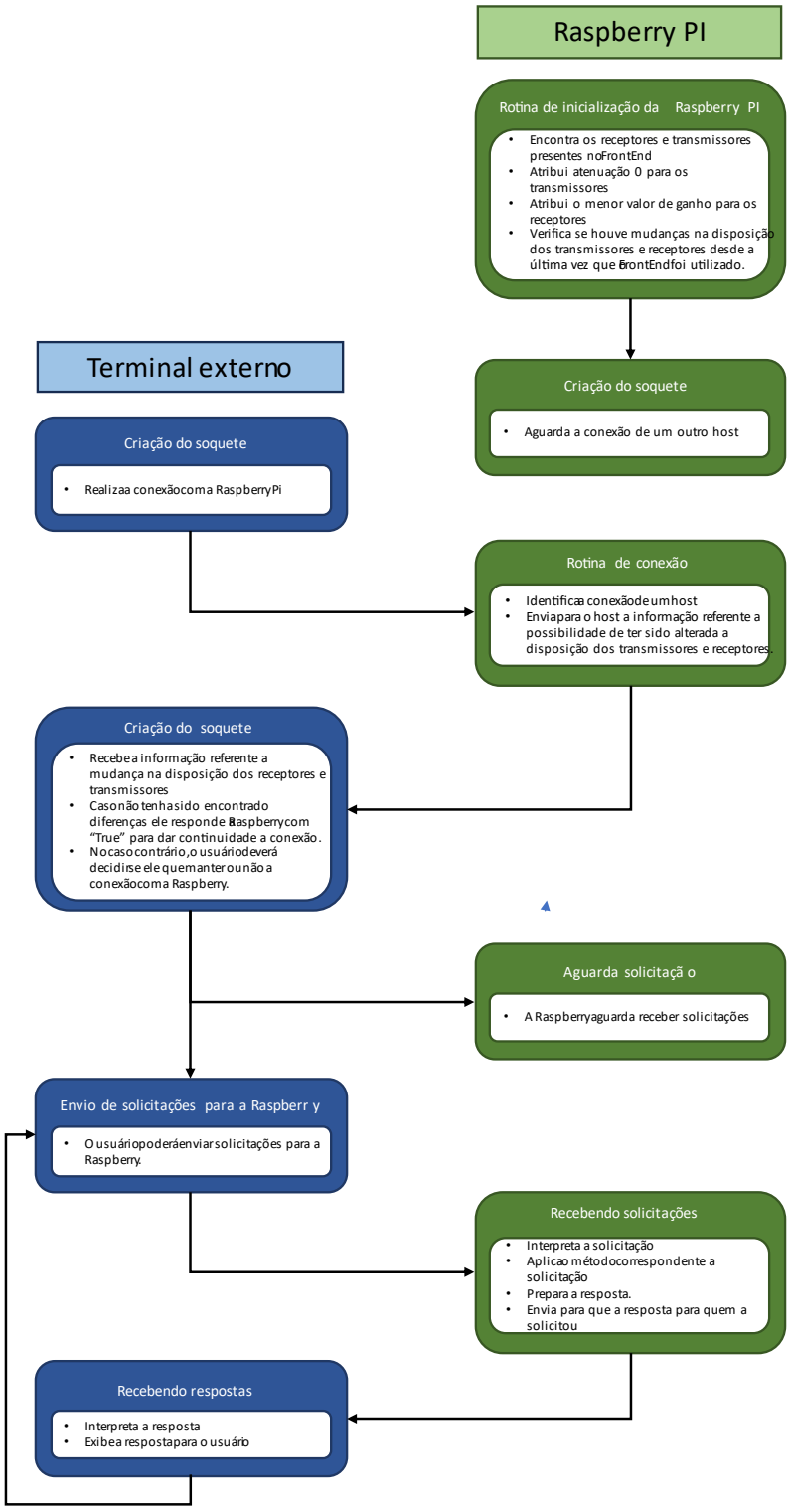

Figura 17 – Dinâmica de funcionamento da comunicação entre um terminal externo e a Raspberry Pi
Fonte: Elaborada pelo autor

O método que realiza essa interpretação é um da classe TalkToRaspberry, que foi desenvolvida para esta aplicação. Os principais métodos desenvolvidos para essa aplicação foram:

- interpret_method;
- method_selected;
- main_talk_rasp;

O método interpret_method recebe a string que foi enviada para Raspberry Pi e retorna qual é o nome do método referente a resposta de solicitação e os parâmetros necessários para que o método seja executado pela Raspberry. Enquanto que, method_selected recebe o nome do método solicitado, os parâmetros dele e a resposta recebida da Raspberry e chama o método específico para tratar e retornar o resultado obtido pelo módulo de controle do FrontEnd. Este método é formado por uma série de condicionais com os nomes dos métodos de usuários existentes para ser selecionados.

As mensagens enviadas para a Raspberry são montadas de forma que o nome do método solicitado seja escrito primeiro, seguido pelos parâmetros escritos entre parênteses e separados por virgulas, conforme mostrado na Figura 17.

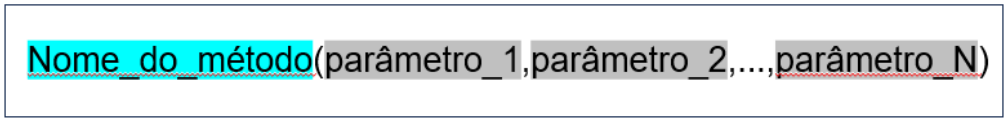

Figura 18 – Formato de string para realizar a solicitação de métodos para o módulo de controle do FrontEnd.

Todos os métodos de usuário contidos no módulo de controle do FrontEnd possuem o seu método correspondente na classe TalkToRaspberry. E as suas saídas são as mesmas dos métodos originais expressos na Tabela 11. Desta forma, quando for adicionado um método de usuário na biblioteca de controle do FrontEnd, deve ser desenvolvido o método correspondente na referida classe e tem que ser adicionado no método method_selected o condicional correspondente a este novo método criado. Desta maneira é possível realizar a comunicação entre um terminal externo e a Raspberry e interpretar o resultado obtido.

6 RESULTADOS

O desenvolvimento desse trabalho exigiu o estudo aprofundado dos projetos eletrônicos das placas correspondentes aos transmissores, receptores e da placa principal, Tx-Rx-Controller. Tal entendimento foi fundamental para que se estabelece-se a comunicação com cada dispositivo e para que eles funcionassem conforme esperado pelo projetista dessas placas.

O andamento dessa pesquisa coincidiu com o processo de aperfeiçoamento desses circuitos. Desta forma, este projeto de automação do FrontEnd foi utilizado e melhorado à medida que novas versões dessas placas fossem finalizadas e testadas. O estudo da topologia do FrontEnd e da Raspberry Pi, do protocolo de comunicação serial SPI, de sockets TCP/IP foram realizados para o desenvolvimento deste projeto, pois foi necessária a habilidade em cada um desses temas para realizar a programação de métodos para controle do FrontEnd, bem como, configurá-lo.

Foi obtido uma biblioteca com métodos para o controle do FrontEnd e uma que realiza a comunicação com a Raspberry Pi de forma que o usuário, em um terminal computador, pudesse enviar os métodos necessários para a aplicação que ele estiver realizando. A construção dos métodos operacionais exigiu o entendimento das funções do FrontEnd para o espectrômetro de ressonância magnética. Esse entendimento permitiu levantar os requisitos dos usuários para operar o FrontEnd, o que resultou nos métodos elaborados neste projeto.

Com a finalidade de demonstrar os resultados obtidos foi realizado o levantamento da curva característica de transmissores e receptores desenvolvidos pelo CIERMag. Foram utilizados 8 transmissores e 8 receptores para a demonstração dos resultados.

A montagem dos aparelhos utilizados para realizar as medidas dos transmissores pode ser vista na figura 18. A Raspberry Pi foi utilizada para programar atenuações de 0 até 60 dB. Ela foi devidamente conectada a uma placa Tx-Rx-Controller que continha 8 transmissores. As medidas foram obtidas por meio de um Network Analyzer, este fornece como saída os ganhos obtidos para uma faixa de frequência, ao qual foram anotados como resultados os valores correspondentes a frequência de 63,4 MHz. Os resultados obtidos estão expressos na Tabela 12.

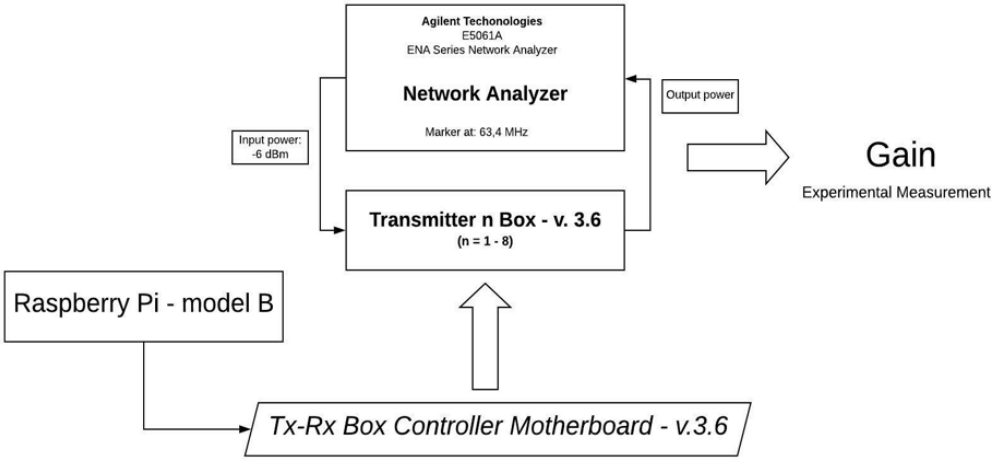

Figura 19 – Aparato experimental utilizado para realizar as medidas de ganho dos transmissores.
Fonte: Elaborada pelo autor

Tabela 13 – Respostas de ganho dos transmissores, em dB, para cada valor de atenuação programada.

Atenuação programada	**Resultado esperado [dB]**	**Tx 1 [dB]**	**Tx 2 [dB]**	**Tx 3 [dB]**	**Tx 4 [dB]**	**Tx 5 [dB]**	**Tx 6 [dB]**	**Tx 7 [dB]**	**Tx 8 [dB]**
0	6,05	5,84	5,73	5,88	5,98	5,73	5,60	5,71	5,47
6	0,05	-0,25	-0,43	-0,22	-0,08	-0,46	-0,39	-0,44	-0,55
12	-5,95	-6,21	-6,44	-6,22	-5,99	-6,48	-6,34	-6,4	-6,46
18	-11,95	-12,17	-12,44	-12,22	-11,90	-12,50	-12,27	-12,40	-12,40
24	-17,95	-18,19	-18,50	-18,27	-17,86	-18,56	-18,27	-18,45	-18,41
30	-23,95	-24,29	-24,63	-24,39	-23,95	-24,69	-24,33	-24,56	-24,48
36	-29,95	-30,68	-30,83	-30,67	-30,64	-30,99	-30,57	-30,54	-36,70
42	-35,95	-36,64	-36,83	-36,66	-36,56	-37,04	-35,56	-36,54	-36,70
48	-41,95	-42,62	-42,89	-42,69	-42,49	-43,06	-42,49	-42,58	-42,69
54	-47,95	-49,69	-48,98	-48,80	-48,57	-49,18	-48,63	-48,66	-48,68
60	-53,95	-54,68	-55,07	-54,82	-54,46	-55,22	-54,66	-54,72	-54,67

Fonte: Elaborada pelo autor.

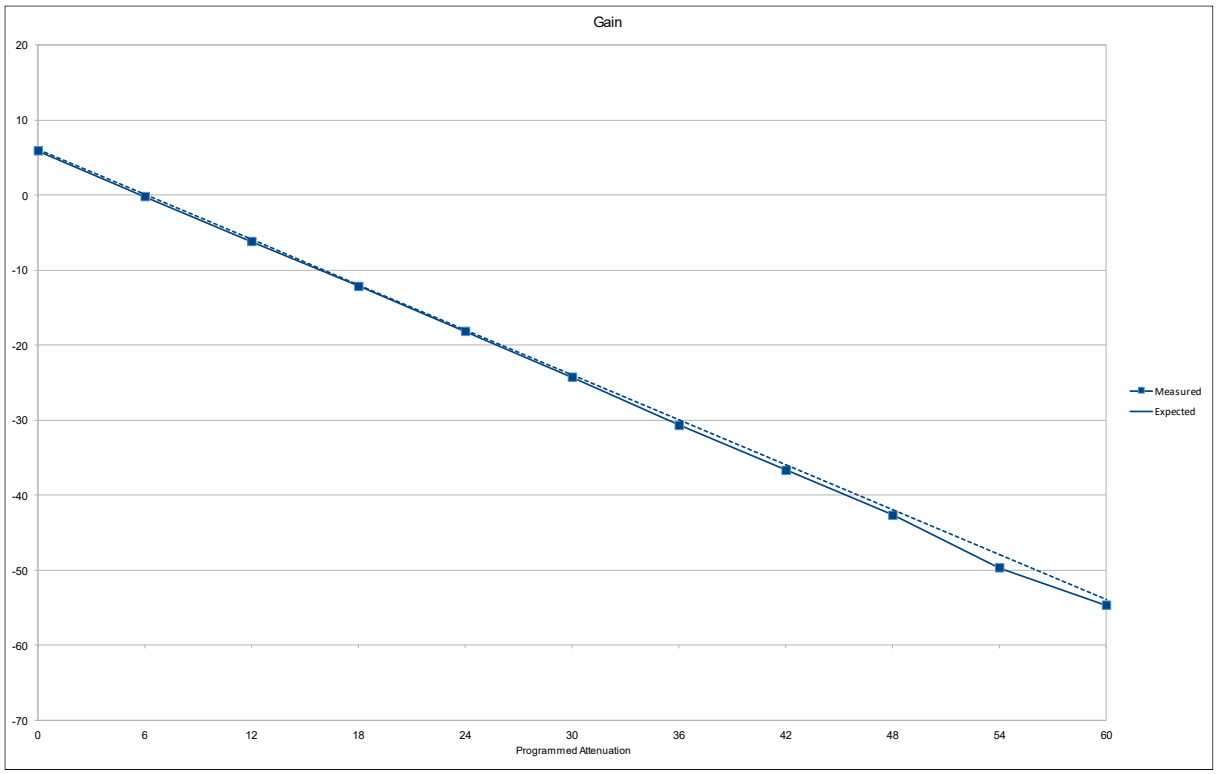

Figura 20 – Resultado obtido para o transmissor TX 1 para a medida de ganho de saída para valores de atenuações programadas nos transmissores.

Fonte: Elaborada pelo autor

A Figura 19 mostra o resultado com o transmissor nomeado como TX1. Note que o resultado medido é próximo ao que é teoricamente esperado

Os receptores contêm dois componentes que são programáveis e que afetam a saída de seu sinal. O primeiro deles seria um VGA, e o segundo seria um amplificador operacional ao qual é programado se ele vai compor o sinal de saída do receptor. A atuação de cada um desses componentes no receptor será apresentada de maneira individual. Deste modo, para realizar a caracterização dos receptores, em relação ao amplificador operacional, foi realizada a montagem dos dispositivos e equipamentos indicada na Figura 20.

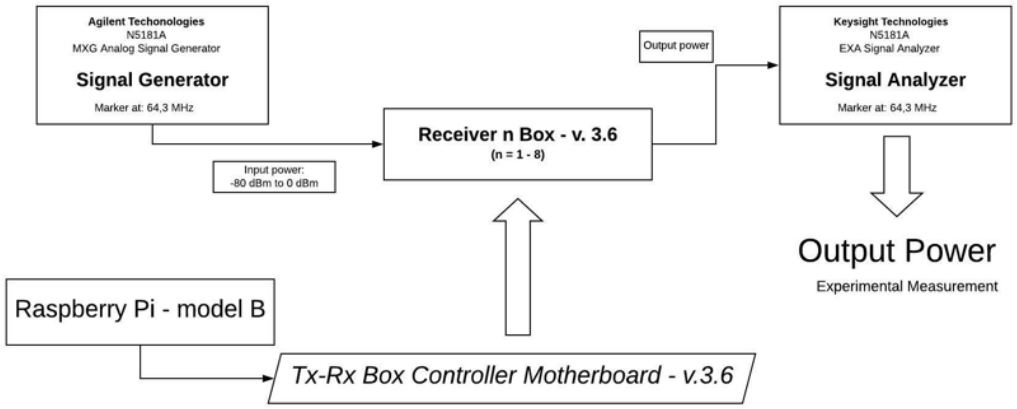

Figura 21 - Aparato experimental utilizado para realizar as medidas da linearidade de ganho de Receptores.
Fonte: Elaborada pelo autor

As caracterizações dos receptores foram realizadas com a presença e a ausência da atuação de um dos amplificadores operacionais. A Tabela 13 mostra os resultados coletados para 8 receptores com a ausência da atuação do amplificador, enquanto a Tabela 14 apresenta os relacionados a presença deste componente eletrônico.

Tabela 14 – Respostas de saída dos receptores, em dB, para cada valor de entrada, quando estes estão com o ganho baixo.

Sinal de entrada [dBm]	**Resultado esperado [dB]**	**Rx 1 [dB]**	**Rx 2 [dB]**	**Rx 3 [dB]**	**Rx 4 [dB]**	**Rx 5 [dB]**	**Rx 6 [dB]**	**Rx 7 [dB]**	**Rx 8 [dB]**
-80	-60,80	-64,00	-64,00	-63,00	-63,00	-63,00	-63,00	-63,00	-63,00
-70	-50,80	-53,90	-53,50	-54,00	-53,00	-53,00	-54,00	-53,00	-53,00
-60	-40,80	-43,90	-43,50	-44,00	-43,00	-43,00	-44,00	-43,00	-43,00
-50	-30,80	-33,90	-33,60	-34,00	-33,56	-33,88	-34,12	-33,00	-33,27
-40	-20,80	-23,95	-23,60	-24,00	-23,56	-23,86	-24,09	-23,77	-23,24
-30	-10,80	-13,96	-13,60	-14,10	-13,57	-13,87	-14,11	-13,79	-13,25
-20	-0,80	-3,98	-3,39	-4,12	-3,60	-3,89	-4,13	-3,80	-3,28
-10	9,20	6,05	6,27	5,20	5,34	5,74	5,45	5,71	6,50
-9	10,20	6,98	7,18	6,10	6,36	6,63	6,49	6,66	7,24
-8	11,20	7,88	7,92	6,95	7,20	7,38	7,22	7,32	8,04
-7	12,20	8,38	8,62	7,59	7,91	8,05	7,90	7,98	8,71
-6	13,20	8,98	9,22	8,13	8,47	8,64	8,45	8,52	9,26
-5	14,20	9,43	9,68	8,51	8,90	9,07	8,86	8,91	9,65
-4	15,20	9,74	10,00	8,76	9,18	9,36	9,12	9,17	9,90
-3	16,20	9,93	10,20	8,89	9,33	9,50	9,25	9,29	9,99
-2	17,20	10,07	10,34	8,98	9,44	9,62	9,34	9,36	10,08

Fonte: Elaborada pelo autor.

Tabela 15 – Respostas de saída dos receptores, em dB, para cada valor de entrada, quando estes estão com o ganho alto.

Sinal de entrada [dBm]	Resultado esperado [dB]	Rx 1 [dB]	Rx 2 [dB]	Rx 3 [dB]	Rx 4 [dB]	Rx 5 [dB]	Rx 6 [dB]	Rx 7 [dB]	Rx 8 [dB]
-80	-38,60	-42,00	-42,00	-42,00	-42,00	-42,00	-42,00	-42,00	-42,00
-70	-28,60	-32,00	-32,00	-33,00	-32,40	-32,00	-32,00	-32,00	-32,00
-60	-18,60	-22,00	-22,40	-23,00	-22,40	-22,70	-22,80	-22,00	-22,00
-50	-8,60	-12,70	-12,47	-13,00	-12,40	-12,70	-12,80	-12,80	-12,80
-40	1,40	-2,72	-2,51	-3,10	-2,45	-2,81	-2,85	-2,84	-2,86
-30	11,40	6,96	7,18	6,52	7,25	6,41	6,60	6,76	6,67
-29	12,40	7,60	7,65	7,37	8,10	7,15	7,51	7,59	7,56
-28	13,40	8,44	8,31	8,11	8,87	7,92	8,30	8,34	8,33
-27	14,40	9,15	8,97	8,81	9,52	8,49	9,01	9,06	9,02
-26	15,40	9,78	9,46	9,40	10,07	8,97	9,53	9,66	9,61
-25	16,40	10,24	9,82	9,86	10,49	9,27	10,11	10,12	10,07

Fonte: Elaborada pelo autor.

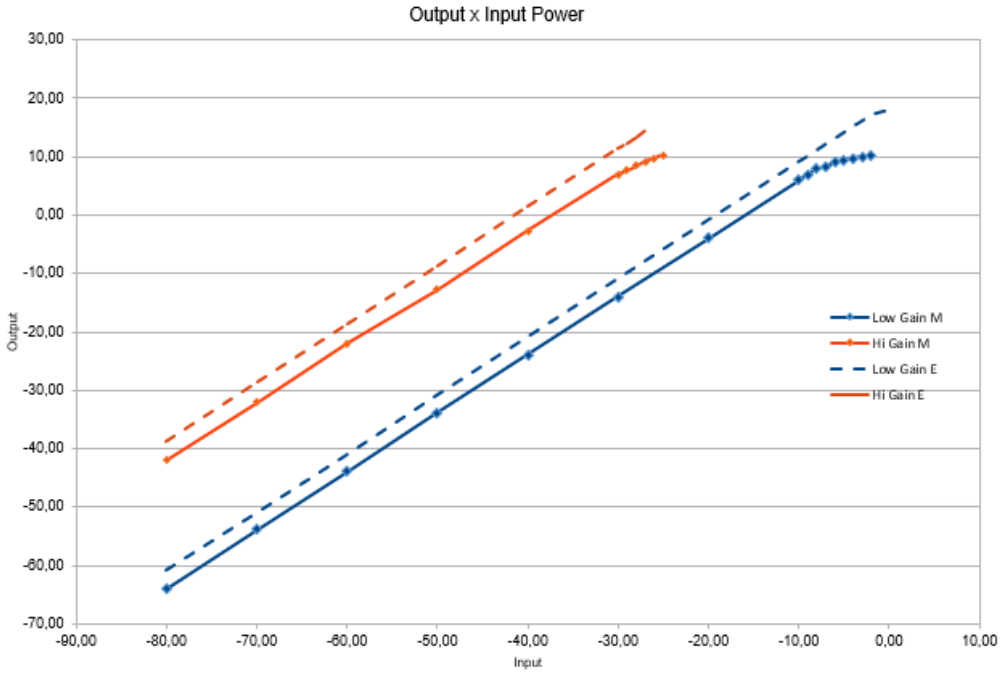

Figura 22 - Resultado obtido para o transmissor RX 1 para a medida da linearidade de ganho dele.

Fonte: Elaborada pelo autor

Para realizar a avaliação da programação do VGA foi realizada a montagem mostrada na Figura 22. Novamente foi utilizado o Network Analyzer. Para cada ganho programado foi registrado o resultado referente a frequência de 63,4 MHz.

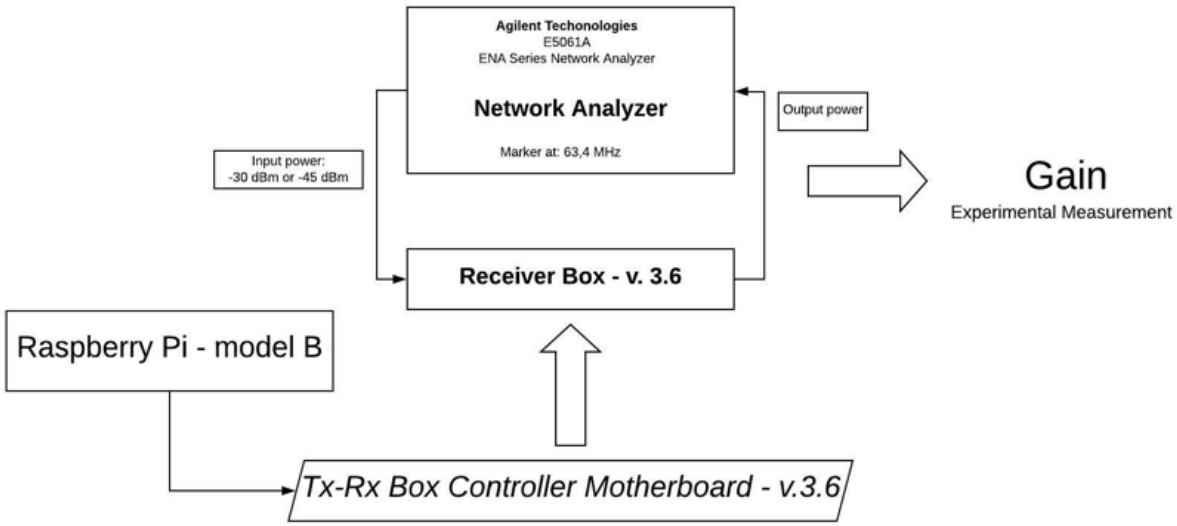

Figura 23 - Aparato experimental utilizado para realizar as medidas de ganho relacionado ao VGA dos receptores.

Fonte: Elaborada pelo autor

A Tabela 15 expõe a resposta de saída do receptor para valores selecionados programados quando um dos amplificadores operacionais não está operando.

Tabela 16 – Respostas de saída dos receptores em ganho baixo, em dB, para cada valor de ganho programado na VGA.

Ganho VGA Programável [dB]	**Rx 1 [dB]**	**Rx 2 [dB]**	**Rx 3 [dB]**	**Rx 4 [dB]**	**Rx 5 [dB]**	**Rx 6 [dB]**	**Rx 7 [dB]**	**Rx 8 [dB]**
-15	19	19	19	19	19	19	19	19
-8	26	26	26	26	26	26	26	26
-1	33	33	33	33	33	33	33	33
-6	40	40	40	40	40	40	40	40
13	47	47	47	47	47	47	47	47
20	54	54	54	54	54	54	54	54
27	61	61	61	61	61	61	61	61
34	68	68	68	68	68	68	68	68

Fonte: Elaborada pelo autor.

A tabela 16 expõe a resposta de saída do receptor para valores selecionados programados quando ambos os amplificadores operacionais estão operando.

Tabela 17 – Respostas de saída dos receptores em ganho baixo, em dB, para cada valor de ganho programado na VGA.

Ganho VGA Programável [dB]	**Rx 1 [dB]**	**Rx 2 [dB]**	**Rx 3 [dB]**	**Rx 4 [dB]**	**Rx 5 [dB]**	**Rx 6 [dB]**	**Rx 7 [dB]**	**Rx 8 [dB]**
-15	40	40	40	40	40	40	40	40
-8	47	47	47	47	47	47	47	47
-1	54	54	54	54	54	54	54	54
-6	61	61	61	61	61	61	61	61
13	68	68	68	68	68	68	68	68
20	75	75	75	75	75	75	75	75
27	82	82	82	82	82	82	82	82
34	82	82	82	82	82	82	82	82

Fonte: Elaborada pelo autor.

7 CONCLUSÃO

O desenvolvimento deste trabalho exigiu o entendimento das topologias do FrontEnd, da Raspberry Pi, assim como foi necessário se aprofundar em temas acerca de protocolos de comunicação serial e de rede. O entendimento da função do FrontEnd para o espectrômetro de ressonância magnético precisou ser buscado para que o desenvolvimento deste trabalho tenha uma aplicação válida, eficiente e que possa ser utilizada pelos operadores e desenvolvedores do software específico de controle do espectrômetro.

Ao fim deste projeto de pesquisa foi obtido uma biblioteca Python estruturada em orientação a objeto que permite o controle do FrontEnd por meio de uma Raspberry Pi, e uma biblioteca auxiliar que visa a comunicação com a Raspberry Pi via um terminal computador externo, para que seja possível enviar os métodos específicos de controle do FrontEnd, bem como para solicitar informações dele.

Dos resultados levantados para a caracterização de transmissores e receptores, é notório que o software foi capaz de traçar os seus perfis de comportamento, uma vez que os valores experimentais foram próximos dos valores teóricos previstos e que os degraus de atenuação na entrada dos dispositivos foram respeitados na saída. Vale ressaltar que os receptores obtiveram esse comportamento com sinais de entrada que variaram de -80 até -30 dBm, quando estavam sobre influência de seus dois amplificadores operacionais, e de -80 até -10 dBm, quando apenas um deles atuava no sinal de saída dos receptores. Para valores maiores que esses, em cada uma das disposições dos amplificadores, o degrau foi se reduzindo. Este fato é esperado quando se trata da caracterização de receptores. As medidas foram realizas até ser percebido o ponto de compressão de 1 dB, que é quando a diferença entre o resultado esperado para saída e o obtido experimentalmente chega a 1 dB. Este ponto caracteriza o início da distorção, que é quando a onda amplificada já não corresponde ao sinal de entrada. É valido ressaltar, também, que o software foi hábil ao lidar com a programação do VGA, visto que os resultados medidos foram lineares e respeitaram os degraus de diferença dos sinais de entrada.

Como perspectiva futura para o projeto, o potencial de controle de periféricos lentos pode ser explorado para outros tipos de periféricos como sensores e atuadores. Por estar inserido no contexto de um espectrômetro de ressonância magnética, um sensor de temperatura para avaliar a amostra em questão ou o controle de uma cama de um paciente que irá fazer uma ressonância magnética utilizando o espectrômetro do CIERMag são itens que necessitam de um controle e tais funcionalidades podem ser adicionadas na biblioteca construída e a Raspberry

pode atuar como interface de hardware. Portanto, pode se concluir que existem perspectivas para o uso e para adequação de novas funcionalidades neste projeto de pesquisa.

REFERÊNCIAS

1 COLNAGO, L. A.; ANDRADE, F. D. RMN no domínio do tempo: fundamentos e aplicações offline e online. *In*: RESENDE, R. R. (org.). **Biotecnologia aplicada à agro &indústria**. São Paulo: Blücher, 2017. v. 4, p. 439-470.

2 FERRARINI, M. C.; HAGE, N. S.; IWASAKI, M. Imagem por ressonância magnética: princípios básicos. **Ciência Rural**, v.39, n. 4, p.1287-1295, 2009.

3 MAZZOLA, A. A. Ressonância magnética: princípios de formação da imagem e aplicações em imagem funcional. **Revista Brasileira de Física Médica**, v. 3, n. 1, p. 117-129, 2009.

4 SILVA, D. M. D. D. **Desenvolvimento de console multiplataforma para aquisição, organização e visualização de dados do espectrômetro digital de RM do CIERMag:** ToRM console. 2014. Dissertação (Mestrado em Ciências) Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2014.

5 SOUZA, P. V. B. D. **Desenvolvimento de um subsistema non-real-time para o gerenciamento de dispositivos periféricos e desenvolvimento de interfaces gráficas**. 2016. Dissertação (Mestrado em Ciências) – Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2016.

6 CAUCHICK-MIGUEL, P. A. **Metodologia científica para engenharia**. Rio de Janeiro: Elsevier Brasil, 2019.

7 HART, C. **Doing a literature review**: releasing the research imagination. New York: Sage Publications, 2018.

8 TANENBAUM, A. S. **Redes de computadores**. Rio de Janeiro: Ed. Campus, 2003.

9 HUNT, C. **TCP/IP network administration**. 3rd ed. Sebastopol: O'Reilly Media, 2002.

10 LEENS, F. An introduction to I2C and SPI protocols. **IEEE Instrumentation & Measurement Magazine**, v. 12, n. 1, p. 8-13, Feb. 2009. DOI: 10.1109/MIM.2009.4762946.

11 MOTOROLA. Freescale Semiconductor. **GUIDE SPI Block V04. 01**. Disponível em: https://www.nxp.com/files-static/microcontrollers/doc/ref_manual/S12SPIV4.pdf. Acesso em: 23 jan. 2021.

12 RASPBERRY PI FOUNDATION. **About Us**. [2012?]. Disponível em: https://www.raspberrypi.org/about/. Acesso em: 21 ago. 2020.

13 ELINUX. **RPi Hardware**. [2019?]. Disponível em: https://elinux.org/RPi_Hardware. Acesso em: 21 ago. 2020.

14 RASPBERRY PI FOUNDATION. **Raspberry Pi OS**. [2012?]. Disponível em: https://www.raspberrypi.org/downloads/. Acesso em: 21 ago. 2020.

15 PYTHON SOFTWARE FOUNDATION. **The Python tutorial**. [2020?]. Disponível em: https://docs.python.org/3/tutorial/index.html Acesso em: 22 ago. 2020.

16 PYTHON SOFTWARE FOUNDATION. **RPi.GPIO 0.7.0**. [2019?]. Disponível em: https://pypi.org/project/RPi.GPIO/. Acesso em: 22 ago. 2020.

17 PYTHON SOFTWARE FOUNDATION. **Spidev 3.5**. [2020?]. Disponível em: https://pypi.org/project/spidev/#description. Acesso em: 23 ago. 2020.

18 PYTHON SOFTWARE FOUNDATION. **Json** - JSON encoder and decoder. [2020?]. Disponível em: https://docs.python.org/3/library/json.html. Acesso em: 25 ago. 2020.

19 PYTHON SOFTWARE FOUNDATION. **Socket** - low-level networking interface. [2020?]. Disponível em: https://docs.python.org/3/library/socket.html Acesso em: 25 de ago. 2020.

20 PYTHON SOFTWARE FOUNDATION. **Time** — time access and conversions. [2020?]. Disponível em: https://docs.python.org/3/library/time.html. Acesso em: 23 ago. 2020.