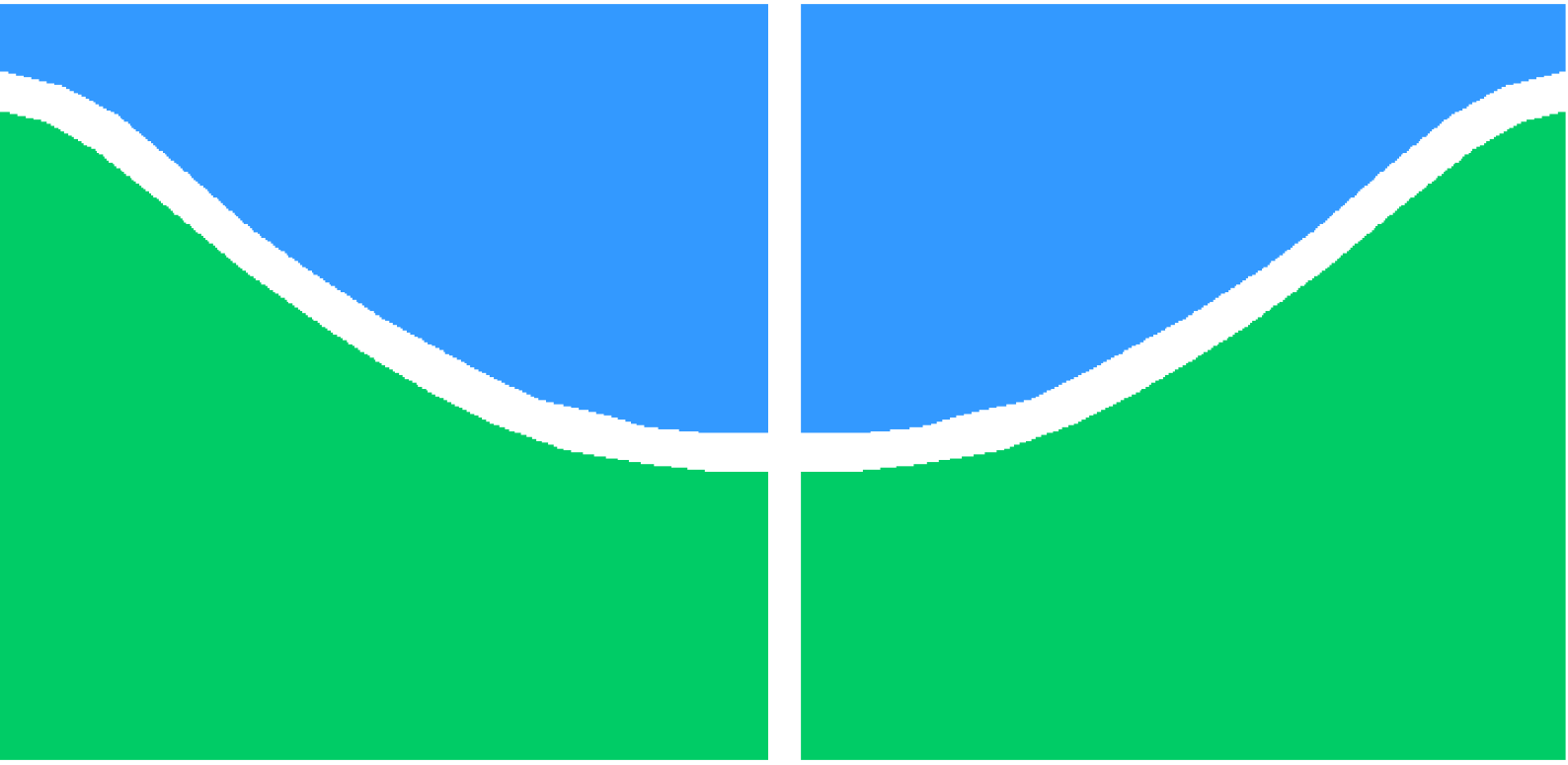

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia Eletrônica

Sistema de otimização para supressão de portadora e casamento de impedância para front-end RF de RFID

Autor: Lenin Andrade de Sousa Cerqueira

Orientador: Sébastien Rondineau

Brasília, DF

2019

Lenin Andrade de Sousa Cerqueira

Sistema de otimização para supressão de portadora e casamento de impedância para front-end RF de RFID

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Sébastien Rondineau

Brasília, DF

2019

Lenin Andrade de Sousa Cerqueira
Sistema de otimização para supressão de portadora e casamento de impedância para front-end RF de RFID/ Lenin Andrade de Sousa Cerqueira. – Brasília, DF, 2019-
76 p. : il. (algumas color.) ; 30 cm.

Orientador: Sébastien Rondineau

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. UHF RFID. 2. Supressão de portadora. I. Sébastien Rondineau. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Sistema de otimização para supressão de portadora e casamento de impedância para front-end RF de RFID

CDU 02:141:005.6

Lenin Andrade de Sousa Cerqueira

Sistema de otimização para supressão de portadora e casamento de impedância para front-end RF de RFID

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Trabalho aprovado. Brasília, DF, 3 de abril de 2019:

Sébastien Rondineau
Orientador

Dr. Wellington Avelino do Amaral
Convidado 1

Dr. Leonardo Aguayo
Convidado 2

Brasília, DF
2019

Dedico o presente trabalho de conclusão de curso à minha família por me dar todo o apoio, aos meus colegas de curso que muito contribuiram para a minha formação, bem como aos docentes da Universidade de Brasília devido à influência positiva que exerceram sobre o meu caráter e formação acadêmica.

Agradecimentos

Este trabalho deve-se, em grande parte, aos ensinamentos dados pelos professores da Universidade de Brasília. Agradeço aos meus familiares e amigos, por confiarem no meu potencial e me incentivarem a alcançar meus objetivos.

Resumo

Quando são usadas etiquetas passivas para identificação por radiofrequência, é necessário que a leitora transmita uma portadora constante durante a comunicação para garantir uma energização suficiente para essa etiqueta se manter operacional. Portanto, essa portadora de alta potência é associada a ruídos que limitam a sensibilidade e, consequentemente, o intervalo de leitura possível. Este projeto propõe uma maneira de cancelar essa portadora na leitora. Quando se trata de aplicações na faixa de frequências muito altas geralmente o receptor é isolado do transmissor fazendo uso de elementos diretivos como circuladores ou acopladores direcionais. Sabendo que as etiquetas se tornam mais sensíveis a fim de melhorar a faixa de leitura, um isolamento relativamente pobre de 20dB – 30dB limita a leitura dessa etiqueta. Uma maneira de otimizar essa separação dos canais de transmissão e recepção de um acoplador direcional é manipulando a impedância da porta não utilizada com o intuito de gerar um sinal que se cancele com o sinal da portadora. Neste projeto foi utilizado circuito de casamento de impedância fazendo uso de capacitores controlados digitalmente. A vantagem dessa solução é a possibilidade de realizar mudanças no sistema a qualquer momento com um controlador a fim de minimizar o vazamento da portadora e as perdas nos sinais de transmissão.

Palavras-chaves: UHF. RFID. Etiquetas. Acoplador direcional. Isolamento de portadora. Casamento de impedância.

Abstract

In ultra high frequency radio frequency identification systems the receiver is usually isolated from transmitter by a circulator or directional coupler. To trigger passive RFID tags, the reader need to transmit a constant carrier during communication to ensure enough power for the tag to operate normally. However, the carrier high power with is associated with noise that limits the receiver sensitivity and, hence, the possible reading range. Since tags become more sensitive in order to improve reading distance, the relatively poor isolation of 20dB – 30dB limits the tag to reader link. A way to improve a directional coupler's isolation is to mismatch the unused port to generate a carrier canceling signal. In this paper an impedance network using digitally tunable capacitors is proposed. The advantage of this solution is that changes in the system can be recalibrated at any time by a controller to minimize carrier leakage.

Key-words: Radio frequency identification. Ultra high frequency. directional coupler. leakage cancelation. isolation. carrier supression.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UHF	*Ultra High Frequency*
HF	*High Frequency*
RFID	Identificação por radiofrequência
IoT	*Internet of Things*
DTC	*Digitally Tunable Capacitor*
ADC	*Analog-to-Digital Converter*
DAC	*Digital-to-Analog Converter*
ADS	*Advanced Design System*
Rx	Receptor
Tx	Transmissor
LNA	*Low Noise Amplifier*
RF	Radiofrequência
SNR	*Signal Noise Ratio*
SPI	*Serial Peripheral Interface*
dB	decibel
RDS	Rádio Definido por Software
PCB	*Printed Circuit Board*
CI	Circuito Integrado
EPC	*Electronic Product Code*
ISO	*International Standard Organization*
GRC	*GNU Radio Companion*
AM	*Amplitude Modulation*
FM	*Frequency Modulation*

QFDM	*Orthogonal Frequency Division Multiplexing*
QAM	*Quadrature Amplitude Modulation*
DPSK	*Differential Phase Shift Keying*
FIR	*Finite Impulse Response*
IIR	*Infinite Impulse Response*
FFT	*Fast Fourier Transform*
SSB	*single-sideband modulation*
DDC	*Digital Down Converter*
DUC	*Digital Up Converter*
DSP	*Digital Signal Processor*
USRP	*Universal Software Radio Peripheral*
IP	*Internet Protocol*
SSH	*Secure Shell*
API	*Application Program Interface*

Lista de símbolos

Γ	Coeficiente de Reflexão na fonte
Γ_{Ant}	Coeficiente de Reflexão na antena
P	Potência de transmissão

Sumário

1 Introdução

Devido às suas vantagens únicas, RFID pode substituir gradualmente o sistema de código de barras, podendo ser usado em vários ambientes, expandindo a partir da vigilância eletrônica até armazenagem, logística e cargas. Juntamente com a rápida ascensão dos programas de IoT (Internet of Things), os sistemas de RFID começam a acelerar e penetrar em todos os aspectos da vida humana (CURTY et al., 2007).

Esta tecnologia recorre à ondas de rádio para se comunicar com etiquetas de RFID, que podem variar bastante o seu custo de acordo com a aplicabilidade e as características desejadas. Contudo, até as etiquetas mais baratas garantem um meio de identificação único e seguro, desta forma, permitindo a implementação de sistemas de comunicação (FINKENZELLER, 2003).

1.1 Sistema de identificação por radiofrequência (RFID)

Com o progresso da tecnologia, o potencial da identificação por radiofrequência aumentou, ao mesmo tempo que os custos de implementação foram decrescendo, cativando, assim, cada vez mais áreas de aplicabilidade. Basicamente, em um sistema de RFID são necessários quatro componentes fundamentais para que este desempenhe a sua tarefa: a leitora, a etiqueta, antenas (do leitor e outra da etiqueta) e o software de gestão.

1.1.1 Etiquetas para RFID

As etiquetas utilizadas em RFID podem ser divididas em duas categorias: ativas e passivas, e o que difere essas duas categorias é basicamente a sua fonte de energia. As etiquetas ativas possuem uma própria fonte de energia, esse fator garante um sinal mais potente e consequentemente um maior raio de alcance. Contudo, o fato de possuírem uma bateria faz com que as suas dimensões sejam maiores e o preço mais elevado. Por possuírem uma dimensão maior em relação às etiquetas passivas, elas têm uma capacidade maior de memória, o que lhes permite conter mais informações relevantes sobre o produto.

Já as etiquetas passivas não possuem nenhuma fonte de energia própria, elas são ativadas obtendo essa energia dos campos eletromagnéticos produzidos pela leitora quando penetram o seu raio de alcance. Para isso, todas as etiquetas passivas possuem um *front end* de RF, que é composto pela antena e pelo circuito de adaptação de carga, e um circuito analógico. Algumas etiquetas mais complexas podem apresentar ainda um circuito digital que pode ser um CI ou um bloco de memória. Por não possuírem uma fonte própria de energia, essas etiquetas passivas são relativamente menores e, em consequência disso, são

mais econômicas e podem ser mais facilmente integradas à várias aplicações. E devido a esse baixo custo, a etiqueta passiva tem um potencial maior de estar presente na maior parte de aplicações de identificação por radiofrequência (RFID).

Existe, além dessas duas mais comuns, as etiquetas semi-ativas. Estas possuem uma fonte de energia própria para alimentar a eletrônica da etiqueta, porém, para se comunicarem recorrem à energia envida pelo transmissor.

As etiquetas de RFID podem ser de várias formas, tamanhos e capacidades. Quando uma solução RFID é projetada, o projetista deve levar em consideração os requisitos tecnológicos antes de escolher antes de escolher qual tipo de para usar [6]. Todas as etiquetas RFID possuem os seguintes componentes essenciais em comum: antena, circuito integrado, PCB (ou substrato)

A responsabilidade principal da antena da etiqueta é de transmitir e receber ondas de rádio com a finalidade de comunicação, é também conhecida como um mecanismo de acoplamento, que pode transformar a energia na forma de radiação eletromagnética. Em um ambiente adequado e com uma proximidade com a leitora, a antena pode coletar energia suficiente para alimentar a etiqueta e os outros componentes sem a necessidade de uma bateria.

As etiquetas são criadas para obedecer a uma categorização chamadas de EPC (*Electronic Product Code*). Para funcionamento em conjunto com a tecnologia RFID, o EPC enviou seus protocolos e técnicas pra a aprovação junto a organização ISO (*International Standard Organization*), criando todo um conjunto de normas para estes sistemas. Com isto o EPC provê especificações técnicas e um número único que identifica cada objeto. O EPC tem definido seis classificações para etiquetas RFID (0 a 5). Além de conter o EPC, cada etiqueta é fabricada de acordo com uma classe específica. Cada classe, dentro do protocolo EPC, atribui uma característica específica que deve ser aplicada àquela etiqueta, como é mostrado a seguir:

- **Classe 0/ Classe 1**: Essas classes fornecem a básica capacidade passiva de radiofrequência. A classe 0 já vem programada de fábrica e além da classe 0, incluindo a classe 1, as etiquetas são programáveis pelo usuário.

- **Classe 2**: Na classe 2, uma nova funcionalidade é adicionada, o que inclui criptografia e leitura e gravação de memória RF.

- **Classe 3**: São utilizadas baterias que alimentam o circuito de lógica. Esta classe oferece maior alance e comunicações de banda larga.

- **Classe 4**: Etiquetas ativas fazem parte da definição da classe 4. comunicação textitpeer-to-peer em banda larga com outras etiquetas ou leitores

- **Classe 5**: As tags de classe 5 contêm energia suficiente para ativar outras etiquetas e podem ser efetivamente classificadas como uma leitora.

Etiquetas passivas, que não possuem fonte de energia embutida e a energia é fornecida pela onda de frequência de rádio criada pela leitora, são geralmente classificadas na faixa de classes de 0 a 3. A classe 4 descreve as etiquetas ativas, que possuem uma fonte de energia interna (uma bateria e que fornece a energia necessária para a operação da etiqueta durante um período de tempo. A classe 5 é reservada para leitoras e etiquetas ativas que podem ler dados de outras etiquetas.

1.1.2 Leitora RFID

A leitora é responsável por organizar a comunicação com qualquer etiqueta em seu alcance de leitura e, em seguida, apresenta os dados das etiquetas a uma aplicação que pode fazer o uso dessa informação.

As leitoras em todos os sistemas podem ser resumidas a dois blocos funcionais: o sistema de controle o bloco de alta frequência (HF), consistindo de um transmissor e receptor, como mostra a Fig.1 (1.1.2). Estes sistemas são manipulados por meio de comandos de controle.

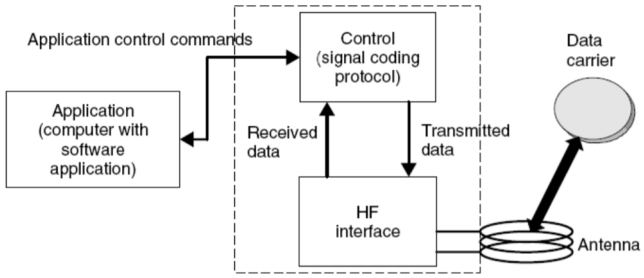

Figura 1 – Leitora RFID consistindo em um bloco de controle e interface de alta frequência

Todo o sistema é controlado por um aplicativo externo por meio de comandos de controle.

A interface de alta frequência tem como objetivo gerar a potência necessária para ativar a etiqueta, realizar a modulação do sinal de transmissão para enviar os dados para a etiqueta e realizar a recepção e demodulação dos sinais de alta frequência transmitida pelas etiquetas.

A unidade de controle da leitora executa as seguintes funções: comunicação com o software e execução dos comandos recebidos pelo aplicativo externo; controle da comunicação com a etiqueta (*master-slave*), como mostra a Fig.2 (1.1.2); codificação e decodificação

do sinal. Em alguns sistemas mais aprimorados, também são executados algoritmos anti-colisão e criptografia dos dados a serem transferidos. Esta unidade é geralmente baseada em um microprocessador para executar essas funções complexas.

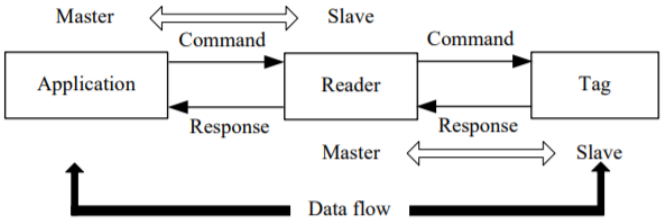

Figura 2 – Sistema de comunicação RFID

1.1.3 Comunicação entre leitora e etiqueta

Essencialmente existem dois métodos para a comunicação entre esses elementos, sendo por meio de acoplamento indutivo ou pelo método de retro-espelhamento.

Com o acoplamento indutivo, a leitora faz uso de uma bobina para gerar um campo eletromagnético variável no tempo que vai induzir uma corrente alternada na etiqueta, essa energia fornecida energiza o chip, o qual pode, então, se comunicar com a leitora pela modulação do sinal. Para esse tipo de comunicação é necessário que as antenas sejam em forma de um solenoide. A quantidade de energia transferida do transmissor para a etiqueta é proporcional ao tamanho das antenas de transmissão e recepção. Com isso, os sistemas de RFID que fazem uso dessa configuração operam na faixa de baixa e média frequência.

Para comunicações em altas frequências se faz o uso da técnica de retro-espelhamente (*Backscatter*), onde a etiqueta absorve a energia a partir de uma onda senoidal contínua irradiada pela leitora e também modula e reflete uma fração desse sinal de volta para o transmissor. Especificamente, a reflexão da onda é devido a uma incompatibilidade intencional entre a impedância e a carga. Variando a impedância da carga faz com que o coeficiente de reflexão varie de acordo com uma sequência aleatória que modula a onda refletida com os bits de informação da etiqueta.

Um sistema passivo de RFID consiste em uma leitora e uma etiqueta passiva. O princípio de funcionamento é basicamente a leitora ativar a etiqueta e, logo após, continuar a enviar o sinal de portadora para fornecer energia. Essa etiqueta envia de volta um sinal modulado após ser ativada. A leitora tem que, então, separar o fraco sinal de resposta do forte sinal da portadora. Essa técnica utilizada é conhecida como retro-espelhamento.

Essa separação dos sinais pode ser realizada com o uso de elementos como o circulador ou acoplador direcional. No entanto, esses elementos não conseguem isolar perfei-

tamente o sinal de transmissão do sinal de recepção. Existem três maneiras possíveis do sinal da portadora vazar para o receptor da leitora, como mostra a Fig.3 (1.1.3).

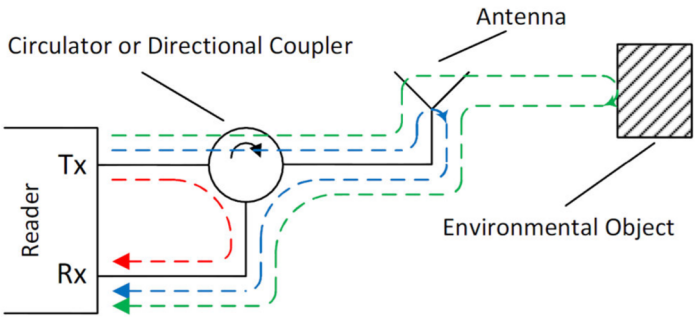

Figura 3 – Caminhos de possíveis vazamentos: entre Tx e Rx; sinal refletido da antena; sinal refletido de objetos no ambiente.

Enquanto o isolamento no circulador ou acoplador direcional e a reflexão na antena são parâmetros fixos para uma configuração específica, as reflexões vindas de objetos do ambiente são variantes no tempo.

1.2 Definição do problema

Um dos maiores desafios em sistemas de RFID é a relação entre a portadora e o sinal refletido pela etiqueta, sendo o sinal refletido o que contém a informação real. O nível da portadora é geralmente mais alto que o nível do sinal refletido, fazendo com que seja necessário um ADC com uma alta resolução de bits para que a digitalização seja completa, a fim de manter o passo de quantização pequeno o suficiente para evitar que o sinal de informação se perca no processo de quantização.

Comparando com um sistema de comunicação móvel geral, uma diferença significativa é que os sistemas de RFID são relativamente simples, as etiquetas apenas realizam uma modulação simples com o sinal de portadora. Não há nenhuma nova frequência de portadora gerada em todo o processo. Isso significa que a frequência do sinal de portadora e a frequência do sinal transmitido são idênticas. Portanto, isso leva a uma grave interferência que limita o desempenho do sistema de RFID, o vazamento de portadora.

As etiquetas passivas de RFID UHF tornam-se mais sensíveis a fim de aumentar o alcance de leitura. Enquanto a potência da portadora de transmissão é limitada, reduzir o consumo de energia das etiquetas é o principal potencial para melhorar a comunicação. Por outro lado, isso aumenta as exigências para essa comunicação com a leitora.

Devido à limitação de isolamento ou das reflexões, uma quantidade significativa de portadora pode vazar para o receptor, o que causa várias dificuldades práticas. O receptor deve ser projetado para uma alta faixa dinâmica, resultando em uma maior área

e consumo de energia. Além disso, a onda contínua de portadora exibirá componentes de ruído de amplitude e fase, o que reduzirá a sensibilidade do receptor. Para melhorar a sensibilidade, o sinal de portadora de onda contínua gerada deve ser de baixo ruído, evitando qualquer modulação não intencional.

Existem dois problemas causados pelo vazamento de portadora. Primeiro, um sinal de alta potência pode saturar o mixer na recepção e limitar o seu desempenho. Outra questão é o ruído de fase ou amplitude da portadora que limita o SNR (*Signal Noise Ratio*) no receptor. Além disso, outros problemas podem piorar a comunicação, como por exemplo uma impedância das antenas ou das linhas de transmissão descasada. Isso pode ocorrer com alguma danificação ou até mesmo um erro de montagem, e todos esses aspectos interferem negativamente na comunicação.

1.3 Objetivos Gerais

Neste projeto é apresentado um método relativamente simples para minimizar todas essas interferências que prejudicam a comunicação entre leitora e as etiquetas, fazendo apenas uma combinação de um acoplador direcional com um circuito de otimização que fará o papel de melhorar o isolamento entre Tx e Rx. Aém disso, utilizando uma configuração semelhante do circuito de otimização, realizar um processo de casamento de impedância nas antenas.

Por meio do software ADS é possível projetar e simular circuitos equivalentes à solução proposta, obtendo informações e comparando com os valores teóricos para, posteriormente, validar o modelo de otimização e casamento de impedância proposto.

1.4 Metodologia

Esse trabalho é estruturado seguindo a lógica de apresentar primeiramente os conceitos básicos referentes às tecnologias de RFID, indicando quais suas vantagens e em quais aspectos podem ser inseridos com maior vantagem. Em seguida, apresenta-se as técnicas de comunicação usadas para as diferentes configurações de antenas e etiquetas, determinando quais as vantagens e desvantagens de cada uma e, também, qual é mais indicada para certo tipo de aplicação. Sequencialmente, mostrando conceitualmente as respectivas limitações para tais aplicações e apontando como pode ser otimizado essas comunicações por RF.

Em seguida, será realizado toda a parte referente ao projeto e implementação do sistema de otimização para supressão de portador e casamento de impedância para aplicações de RFID, o qual primeiramente será projetado e simulado no software ADS, onde,

através da ferramenta de otimização do software, será encontrado os valores desejados para que a otimização seja alcançada e, por fim, validar a solução.

Após as simulações, realizar a montagem do setup para as medições necessárias para a validação. Para realizar os testes e medições necessárias para validar o sistema de otimização, foi montado um esquema utilizando uma leitora de sinal de RF (onde contém o bloco com os DTCs), um rádio definido por software (RDS), uma raspberry pi 3 model B e um computador com sistema Linux para administrar o software *GNU Radio* responsável por controlar o RDS.

O Anexo I apresenta um organograma do projeto, indicando todos os passos realizados para a conclusão do projeto.

2 Sistema de Otimização

Para solucionar os problemas citados, um bloco de supressão de portadora deve ser colocado antes da digitalização ocorrer, ou seja, antes do sinal chegar à leitora. O fator de isolamento do circuito de desacoplamento da portadora determina a sensibilidade, que, neste caso, é diretamente proporcional ao isolamento da portadora.

Para chegar em um bom isolamento entre Tx e Rx, é introduzido um sistema com um acoplador direcional e, como complemento, um fixo descasamento na sua porta livre, entretanto, essa solução é compatível com uma configuração única de antena. Para solucionar esses problemas considerando diversos tipos de antenas, com diferentes valores de impedância, é utilizado um sistema de impedância ajustável com os DTCs.

2.1 *Digitally Tunable Capacitor*

O DTC utilizado foi o PE64904 e o modelo de circuito equivalente utilizado para as simulações pode ser observado na Fig. 4(2.1). Esse componente muito versátil pode ser utilizado em duas configurações: série ou "*shunt*", para suportar uma ampla variedade de topologias. Esse dispositivo é controlado via interface SPI.

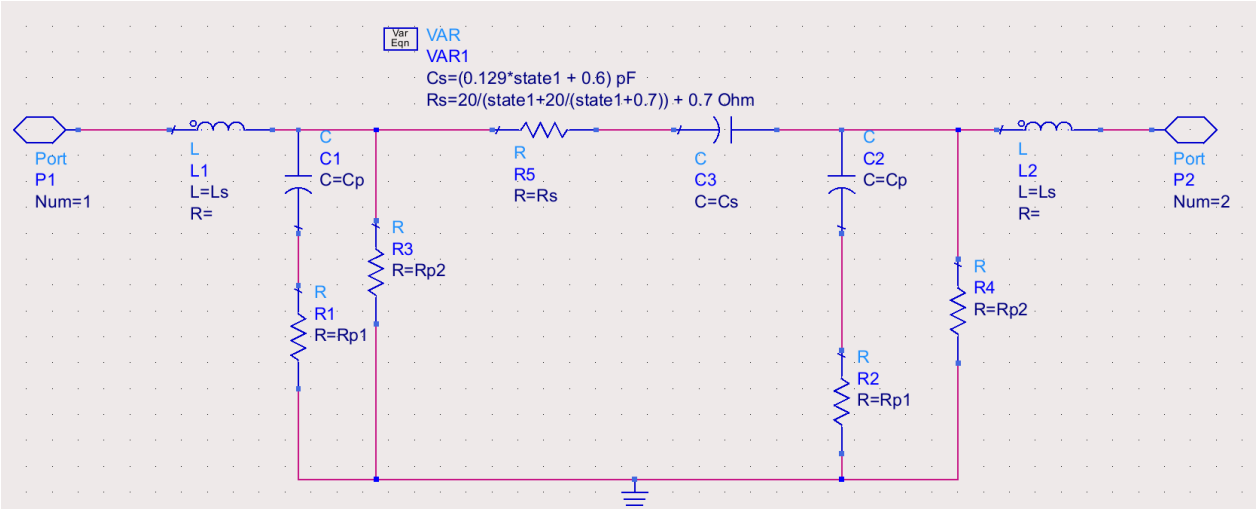

Figura 4 – Esquemático do modelo equivalente do DTC

Esse modelo de circuito equivalente inclui todos os elementos parasitas referente ao circuito físico e fornece resultados precisos com os dados medidos.

Na Tabela 1 a seguir encontra-se os valores referentes aos elementos presentes no modelo equivalente. A variável de estado do DTC (*state*) varia entre 0 e 31, o que

Variável	Equação	Unidade
Cs	$0.129 \cdot state + 0.6$	pF
Rs	$\frac{20}{\frac{(state+20)}{state+0.7}} + 0.7$	Ω
Rp1	7	Ω
Rp2	10	kΩ
Cp	0.5	pF
Ls	0.27	nH

Tabela 1 – Parâmetros do circuito equivalente DTC

possibilita uma variação da capacitância entre 0.6pF e 4.6pF na configuração em série e 1.14pF e 5.1pF na configuração "*shunt*"com uma resolução de aproximadamente 129fF.

2.2 Arquitetura

Como mostrado na Figura 5 (2.2), um acoplador direcional é escolhido como o dispositivo que irá isolar os sinais de transmissão Tx dos sinais de recepção Rx. Em aplicações comuns, a porta 4 do acoplador direcional é conectada a uma carga de 50 Ω, fazendo com que não haja reflexões recorrentes dessa porta. Para reduzir a potência da portadora na porta 3, porta de recepção, pode-se aproveitar de uma reflexão vinda da porta 4 de forma que a energia refletida na antena e a energia vazada da porta 1 para a porta 3 sejam canceladas.

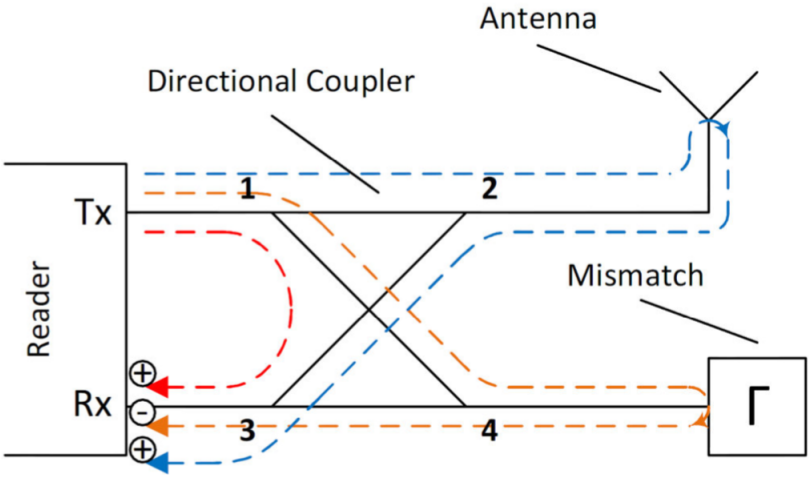

Figura 5 – Princípio de descasamento no acoplador direcional: sinal refletido pela porta 4 cancelando as outras reflexões.

As soluções de cancelamento com impedâncias estáticas (elementos concentrados ou linhas de transmissão) não levam em consideração as reflexões vindas de objetos no ambiente. Outra desvantagem é que a impedância deve ser calibrada para cada impedância de antena e comprimento de cabo.

O método de supressão de portadora utilizado neste projeto mostra seu benefício total porque anula não só a potência da portadora, mas também o ruído intrínseco de amplitude e fase. Como o ruído de fase está correlacionado ao ruído oscilador local dos receptores, o cancelamento será quase perfeito. Mesmo o ruído de fase não correlacionado, gerado pelo amplificador de potência, pode ser cancelado com o método proposto.

Esta é uma grande vantagem comparada com todos os outros métodos, que usam modulador vetorial ou o modulador I/Q, mostrado em (MAYORDOMO; BERNHARD, 2011) e (PURSULA; KIVIRANTA; SEPPä, 2009), para cancelar a portadora com meios ativos. Esses moduladores contribuem com o seu próprio ruído não correlacionado para o receptor, o que não pode ser cancelado posteriormente. Similares, os métodos que fazem uso dos diodos PIN, citados em (BRAUNER; ZHAO, 2009), contribuem com o ruído não correlacionado causado pela corrente de polarização. Em vez disso, apenas elementos como resistores, capacitores e indutores são usados. Os DTCs são dispositivos sintonizáveis e apresentam uma alta linearidade.

Portanto, o circuito de impedância deve incluir, pelo menos, dois elementos sintonizáveis (DTC) a fim de ajustar a parte real e imaginária da impedância. A Fig. 6(2.2) mostra um exemplo de configuração possível usando dois capacitores sintonizáveis PE64904. As duas faixas de capacitância são modificadas com o capacitor C_1 e indutor L_2, respectivamente.

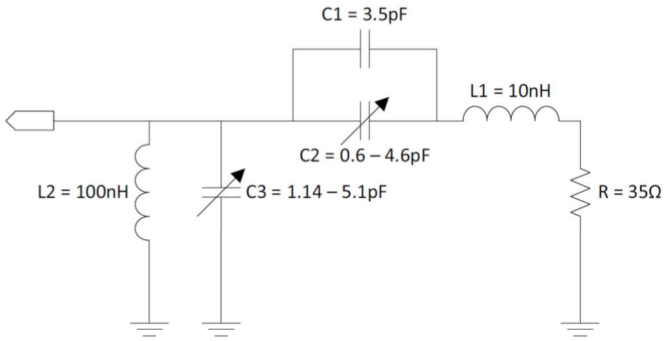

Figura 6 – Exemplo do circuito de impedância ajustável.

2.3 Vantagens de um alto isolamento de portadora

O isolamento de portadora é de importância crítica nos sistemas de RFID. Aumentando esse fator, obtém-se muitas vantagens para o sistema:

- Com um nível de portadora mais baixo, o sinal pode ser amplificando sem saturação, portanto, uma maior sensibilidade é alcançada;

- O SNR do sinal é significativamente melhorado. Tal melhoria é alcançada pelo fato de que a supressão de portadora reduz o nível de potência e, com ele, o ruído de fase da portadora. O sinal de informação não é afetado pela supressão.
- A complexidade do processamento do sinal na leitora é diminuída.

2.4 Teoria do descasamento no acoplador direcional

Em um sistema onde o coeficiente de reflexão da antena Γ_{Ant} e os parâmetros S do acoplador direcional são conhecidos, o coeficiente de reflexão Γ_4 ótimo na porta 4 pode ser obtido. A potência P_3 vazada para Rx é calculada na "Eq. (2.1)" onde P_1 é a potência de transmissão da leitora.

$$P_3 = P_1(S_{31} + S_{21}\Gamma_{Ant}S_{32} + S_{41}\Gamma_4 S_{34}) \tag{2.1}$$

Para obter um valor desejado de Γ_4, a potência P_3 é configurada para zero e, assim, isola Γ_4, como é mostrado na "Eq. (2.2)"

$$\Gamma_4 = -\frac{S_{31} + S_{21}\Gamma_{Ant}S_{32}}{S_{41}S_{34}} \tag{2.2}$$

A partir disso, dois casos especiais podem ser mencionados. Supondo um casamento perfeito com a antena, $\Gamma_{Ant} = 0$, obtem-se:

$$\Gamma_4 = -\frac{S_{31}}{S_{41}S_{34}} \tag{2.3}$$

Neste caso, apenas o acoplador direcional é aprimorado. E supondo um acoplador direcional perfeito, onde não ocorre nenhuma atenuação e nenhuma reflexão ($S_{31} = 0$, $S_{21} = S_{34} = 1$, $S_{41} = S_{32}$) a "Eq. (2.2)" resulta em apenas:

$$\Gamma_4 = -\Gamma_{Ant} \tag{2.4}$$

Vale ressaltar que um valor adequado de Γ_4 também pode ser encontrado no caso não ideal de um imperfeito acoplador, citado em (KIM et al., 2006).

3 Montagem e configurações para medições

Para a validação do sistema de otimização, foi montado um esquema utilizando um RDS que foi responsável apenas por gerar o sinal de radiofrequencia para a placa de circuito impresso de RF, onde se encontra o bloco de otimização. Para o controle dos DTCs utilizados na supressão de portadora, foi utilizado um microcontrolador Raspberry pi 3 Model B. A Fig. 7(3) mostra como esses blocos são conectados entre si.

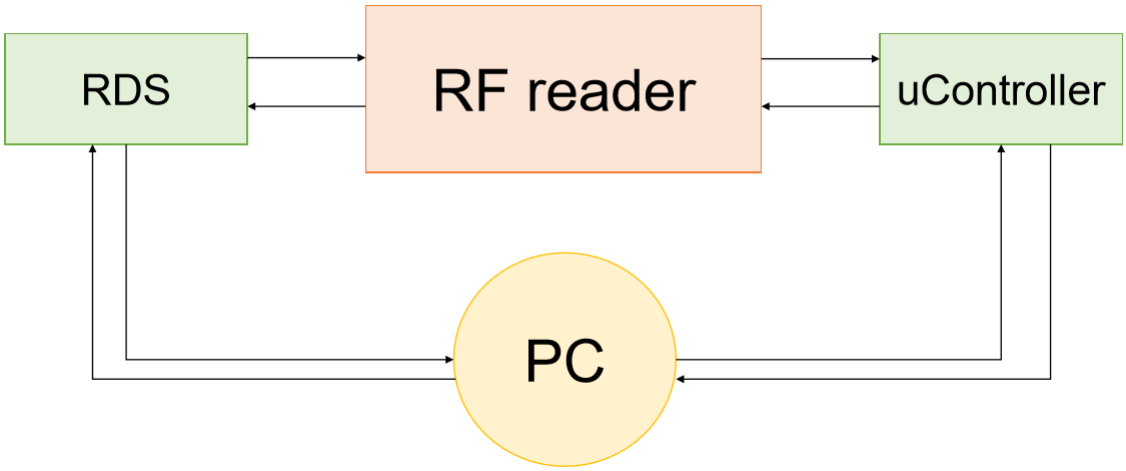

Figura 7 – Diagrama de blocos da montagem para testes.

3.1 RF Reader

A placa de circuito impresso de RF utilizada foi previamente fabricada pela empresa Solentech, em que realizei o período de estágio supervisionado. Nesta placa de RF se encontra o sistema de otimização para supressão de portadora e é mostrada na Fig. 8(3.1).

A comunicação para alimentação e controle dos módulos internos da placa RF é feita a partir do conector ZIF de 22 posições, e por não ser um conector compatível com os conectores presentes na raspberry foram necessárias algumas modificações para que se tornasse possível uma ligação com o microprocessador.

Já a comunicação com o RDS, por se tratar de sinais de alta frequência, são necessários conectores dedicados que são ligados diretamente nos componentes para teste presentes na PCB. Os conectores para transmissão e recepção encontrados no RDS são do tipo SMA e os conectores para teste na placa RF utilizados para as medições são do tipo *subminiature coaxial switch*(fig. 9) e U.FL series (fig. 10)

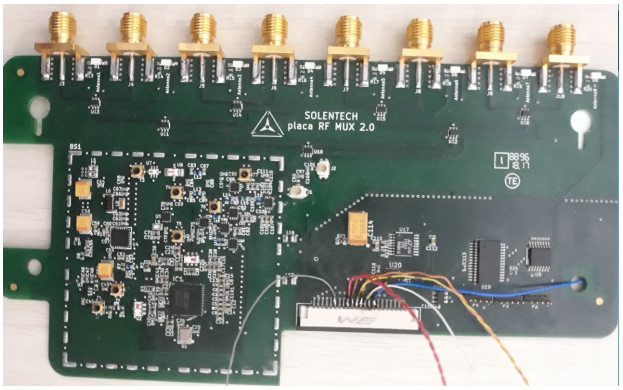

Figura 8 – Placa leitora RF.

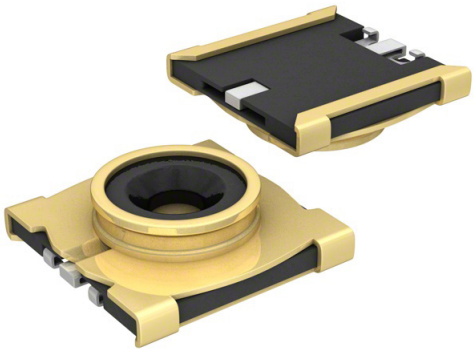

Figura 9 – Conector *subminiature coaxial switch*.

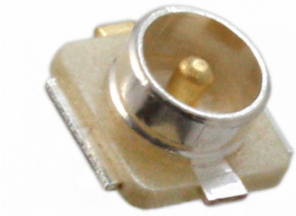

Figura 10 – Conector U.FL Series.

Assim, os cabos para a recepção e transmissão do sinal RF devem conter os seus terminais compatíveis com entradas SMA e com os conectores de teste, um exemplo é mostrado na Fig. 11(3.1)

Figura 11 – Cabo RF com transição SMA female - U.FL.

3.2 Rádio Definido por Software

Os Rádios Definidos por Software (*SoftwareDefined Radios – SDR*) são um conjunto de tecnologias de hardware e software, onde algumas ou todas as funções operacionais do rádio são implementadas através de software ou firmware modificável, que é executado em tecnologias de processamento programáveis (por exemplo, um FPGA, um CPU genérico).

Figura 12 – Rádio definido por software (RDS).

O conceito de radio definido por software (RDS) não deve ser confundido com rádios baseados ou controlados por software, pois hoje, praticamente todos os rádios se utilizam de software em sua concepção. Estes rádios baseados ou controlados por software necessitam de ajustes no hardware para qualquer mudança em interfaces baseadas em software. Ou seja, com o software podem ser controlados parâmetros do rádio como, frequência de operação, modo de operação (AM, FM, SSB), controle de ganho, etc. O software é só uma interface para ajustes no próprio hardware. Já um RDS ideal deve ter todo o processamento de sinal feito através de software, exceto a digitalização do sinal que deve ser feita logo após a captação feita pela antena.

Os rádios definidos por software (RDS) são rádios onde o processamento do sinal é feito através de um software sendo executado em um processador, ou seja, sem a necessidade de hardware para que o sinal possa ser interpretado e seja entregue a sua aplicação. É uma tecnologia com um alto grau de flexibilidade, pois problemas que em rádios con-

vencionais são tratados através de mudanças de hardware, passam a serem tratados em software

3.2.1 Arquitetura de um RDS

A arquitetura de um rádio convencional ou de um RDS é facilmente identificada e divida em 2 componentes majoritários

- *Front End de rádio*: Responsável por cuidar do recebimento e transmissão das frequências de rádio;
- *Back End de rádio*: Responsável pelo processamento do sinal de rádio.

O front end RF tem como objetivo preparar o sinal para que o conversor A/D possa digitalizar o sinal recebido pela antena, e voltar o sinal para frequência original de transmissão antes de ser transmitido pela antena. Esta preparação é feita através de uma amplificação do sinal, um controle de ganho, deslocamento para uma frequência intermediária (no caso de recepção) ou deslocamento para frequência original do sinal (no caso de transmissão), uma filtragem *anti-aliasing*.

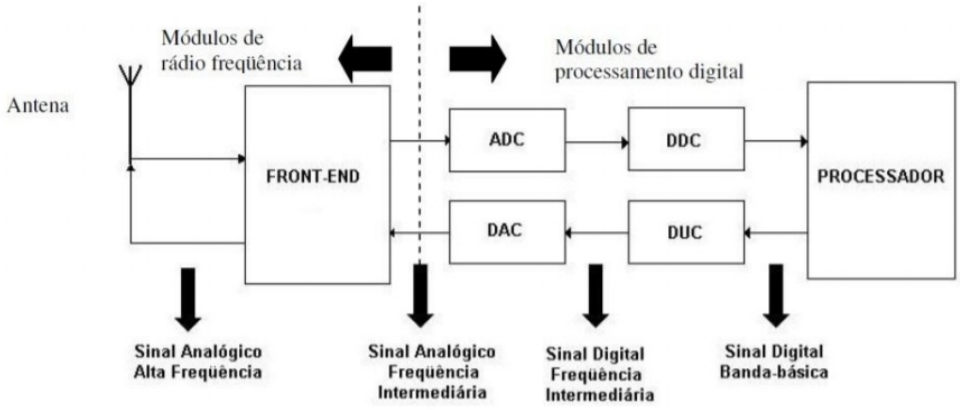

Figura 13 – Estrutura básica de um RDS.

função do DDC (*Digital Down Converter*) é fazer a re-amostragem do sinal digital de frequência intermediária para um sinal digital de banda base para que o processamento não seja efetuado em velocidades muito altas. A função do DUC (*Digital Up Converter*) é efetuar o inverso do DDC, deixando o sinal novamente na frequência digital intermediaria, pronto para o conversor DA.

A frequência intermediária do sinal, após o front end RF, não deve ser uma frequência alta, atendendo a frequência de operação dos DAC e ADC, que por limitações de tecnologia e custo são de baixa frequência em projetos de RDS. Esta frequência é conhecida como frequência intermediária, pois logo após a digitalização do sinal sofre mais

um abaixamento de frequência tornando sinal digitalizado em banda base, ou seja, é uma frequência intermediária em relação ao sinal de entrada ou saída e o sinal processador pelo software.

Um RDS dispõe de vários tipos de processadores para execução de seu software, como os DSPs (processadores digitais de sinais), os FPGAs (conjunto de portas lógicas programáveis) e os ASICs (circuitos integrados de aplicações específicas). Entretanto, um rádio definido por software pode ser executado em um microcomputador qualquer, através de plataformas livres e front-ends específicos desenvolvidos para pesquisa e desenvolvimento nesta área. A GNU Radio e a USRP são um exemplo de plataformas de pesquisa muito difundidos.

3.2.2 Plataformas de implementação de RDS

Existem muitas plataformas de implementação de rádios definidos por software, a GNU Radio é uma delas. É uma ferramenta *open source*, que pode ser implementada em linguagem de programação ou em interface gráfica.

GNU Radio é um *framework de software* livre para o desenvolvimento de rádios definidos por software. Cada rádio implementado no GNU Radio é composto de um conjunto de blocos de processamento de sinais independentes e interligados, que podem ser obtidos da biblioteca ou desenvolvidos pelo usuário.

Todo radio implementado no GNU Radio deve ser visto como um diagrama de fluxo de sinais. Assim os blocos devem estar conectados numa sequência que parte de um ou mais blocos fonte de sinais, passando por blocos intermediários, e finalizando em um ou mais blocos "sink". Entretanto, sempre deve haver ao menos uma fonte e um "sink".

Cada bloco no GNU Radio tem entradas e saídas associadas que podem ser ligadas a vários tipos de fluxos de dados. A ligação de diferentes blocos cria um fluxo que implementa os passos de processamento de um dado sistema de comunicações. Além das entradas e saídas, cada bloco tem um conjunto específico de parâmetros que definem o seu comportamento. O GNU Radio fornece centenas de blocos para os usos mais comuns, tais como operações matemáticas, conversão de tipo de dados, modulação e demodulação, entre outros. Além disso, novos blocos podem ser criados com base nas necessidades do usuário.

Dentre os blocos que compõe está plataforma de software livre, pode-se mencionar:

- Operações matemáticas, como soma, multiplicação, subtração, logaritmo, entre outras;
- Portas lógicas;

- Moduladores, como OFDM, QAM, DPSK, entre outros;
- Filtros FIR, IIR, passa-banda, passa-baixa, etc;
- Interpolação e decimação;
- Blocos de ligação com a USPR e USPR2;
- Controles de ganho;
- *Scramblers* (embaralhadores de sinal);
- A FFT (Transformada Rápida de Fourier);
- Flow Graphs;
- Sources;
- Sinks.

A implementação de um RDS em GNU Radio é feita através de duas linguagens de programação. A linguagem C++ é destinada para o processamento dos sinais, por ser uma linguagem de baixo nível e um alto desempenho de processamento. Já a interligação dos blocos é feita através da linguagem Python, uma linguagem de programação em alto nível que dá uma maior facilidade e agilidade para fazer as ligações.

3.2.2.1 GNU Radio Companion

GNU Radio Companion (GRC) é uma ferramenta gráfica livre para criar protótipos de sistemas de comunicação rápidos e funcionais a partir de diagramas de fluxos de sinais elétricos, permitindo gerar o respectivo código fonte desses fluxos. Programas no GNU Radio não precisam necessariamente serem feitos no GRC, entretanto ele permite uma prototipagem rápida e facilita o entendimento dos programas sendo desenvolvidos. Ele possui facilidades como sistemas para o desenvolvimento rápido de interfaces gráficas e análise estática do diagrama de módulos para encontrar erros simples de configuração. Nesta subseção vamos começar a explorar o GRC.

3.3 Computador com GRC

Quando se desenvolve um rádio definido por software é natural ter como objetivo transmitir ou receber sinais reais. Para tanto, basta incluir um bloco fonte ou "sink" que acesse algum front-end de rádio frequência.

Existem diversos front-end disponíveis no mercado e alguns deles com blocos já disponíveis no GNU Radio. Um exemplo desses blocos permite a conexão com a *Universal*

Software Radio Peripheral (USRP), que como o nome sugere, é uma placa integrada que possibilita a implementação (via comando de software) dos elementos utilizados nos transmissores e receptores. Essa placa incorpora basicamente conversores AD/DA, uma FPGA que é responsável pelo pré-processamento dos sinais de entrada e *slots* para inclusão de front-end de acordo com faixa de frequência de interesse. No caso de um sistema receptor, basicamente o sinal recebido em uma frequência especificada é convertido para banda básica, digitalizado e enviado ao computador via cabo USB 2.0 ou Ethernet.

O RDS implementado com o GNC é basicamente para fornecer o sinal de RF para a leitora e pode ser observado a distribuição dos blocos na Fig. 14(3.3)

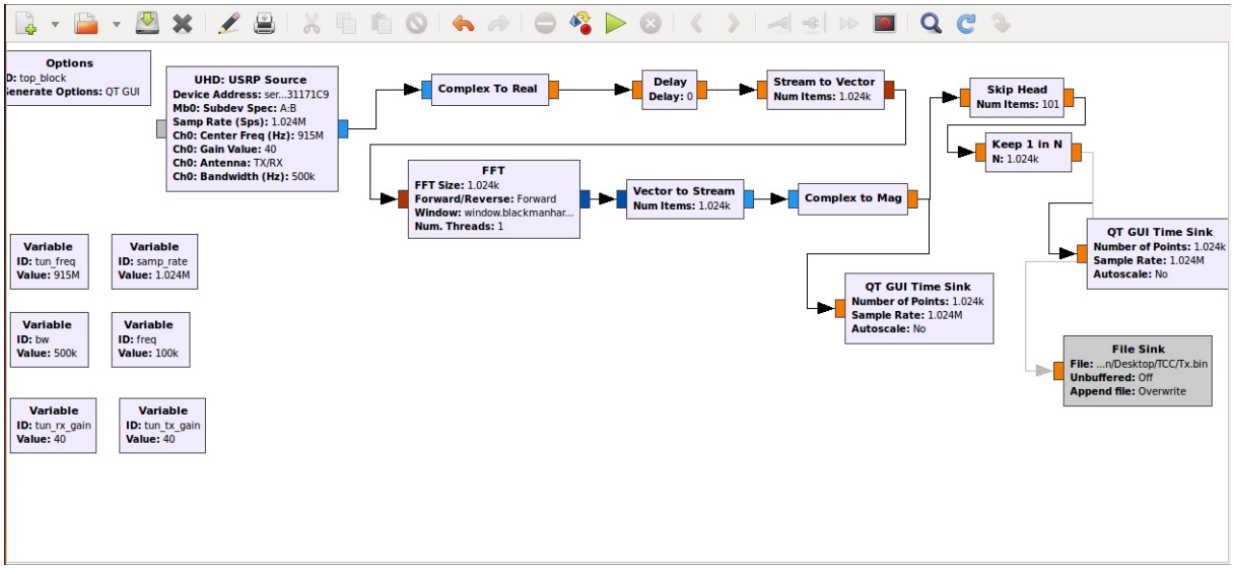

Figura 14 – Diagrama desenvolvido no GRC para gerar sinal RF.

O sinal gerado, digitalizado, é enviado para o computador via cabo USB com o nome de “Tx.bin”, e este arquivo contendo a informação do sinal de transmissão será utilizado para a verificação da otimização durante a execução do algoritmo no microcontrolador.

3.4 Microcontrolador

A raspberry é responsável pela comunicação direta com a placa RF via SPI para controlar os DTCs e executar o algoritmo da otimização de acordo com o arquivo recebido gerado pelo RDS.

A conexão entre a raspberry e o computador foi feita via cabo Ethernet por acesso remoto SSH. O SSH (Secure Shell) também conhecido como Secure Socket Shell é um protocolo que permite, de forma segura, acessar remotamente através de um PC ou outro dispositivo através do IP da rede em que estão conectados. Para facilitar o acesso, foi configurado um IP estático na raspberry.

Com o SSH já ativo e configurado adequadamente, foi implementado, juntamente com o algoritmo de otimização, um socket de internet para realizar a transferência de arquivos entre o PC e a raspberry, sendo o PC atuando como o servidor e transferindo o arquivo “TX.bin”, e a raspberry sendo o cliente e recebendo o arquivo.

Os códigos utilizados na raspberry podem ser visualizados nos anexos B e C ao final do documento.

O código disposto no anexo C descreve o programa que realiza a função de servidor na cominucação "servidor/cliente"via socket TCP/IP. Para essa arquitetura é necessário implementar um programa cliente e um programa servidor onde ambos fazem uso da mesma API de sockets.

Existem varias famílias de endereço e cada uma corresponde a um protocolo em particular. As famílias mais usadas são :

- **AF_UNIX**: Protocolo interno do UNIX
- **AF_INET**: Protocolo Internet
- **AF_NS**: Protocolo de Xerox NS

O protocolo internet foi o utilizado no projeto.

Pelo fato do servidor e cliente usarem a mesma API, o servidor inicialmente cria um socket. No entanto, diferentemente do cliente, o servidor precisa fazer um bind(), que associa o socket a uma porta do sistema operacional, e depois utilizar a função listen() para escutar novas conexões de clientes nessa porta.

Quando um novo cliente faz uma nova conexão, a chamada accept() é utilizada para começar a se comunicar e o servidor fica em um loop recebendo e enviando mensagens através do par de funções send() e recv(). Quando a comunicação com o cliente termina, o servidor volta a aguardar novas conexões de clientes.

O cliente criado para a comunicação com o servidor está presente no código disposto no anexo B, com a função create_socket().

O programa cliente primeiro cria um socket através da função socket(). Em seguida ele se conecta ao servidor através da função connect() e inicia um loop (laço) que fica fazendo send() (envio) e recv() (recebimento) com as mensagens específicas da aplicação. É no par send, recv que temos a comunicação lógica. Quando alguma mensagem da aplicação diz que é o momento de terminar a conexão, o programa chama a função close() para finalizar o socket.

A função create_socket() esta sendo chamada sempre dentro da função tunerGetReflected() com o intuito de sempre realizar a conexão com o servidor e realizar a transferência do arquivo necessário para a verificação do sinal.

Como pode-se observar, a função tunerGetReflected() está sendo chamada dentro todas as funções de otimização com o algoritmo Hill Climb e suas variações. Com isso, garante que o arquivo que contém a informação do sinal de transmissão está sempre sendo verificado durante todas as etapas do algoritmo, assim, garantindo que o sinal será otimizado adaptavivamente, fazendo a verificação sempre com o sinal já atualizado a cada etapa de otimização.

A função tunerSetCap() é responsável pela modificação do valor dos DTCs. As configurações da comunicação SPI estão sendo acionadas e inicializadas para a ligação com os varicaps. Como a raspberry utilizada so tem disponível apenas 2 portas de seleção para a comunicação SPI (SPI0 CS0/CS1) e são necessárias 3 portas de seleção, sendo uma para cada DTC utilizado, então os pinos de GPIO 27, 22 e 23, que estão definidas como SenTune1, SenTune2 e SenTune3, foram selecionados para realizar a função de seleção dos varicaps.

Deste modo, sempre que os pinos de seleção forem acionados, eles estão permitindo que o valor do varicap selecionado seja alterado, e logo após o termino dessa ação eles são desativados novamente, para evitar que ocorra alguma modificação indesejada. Esta ação é realizada pela função bcm2835_spi_transfer().

Os valores que são passados como parâmetro na função bcm2835_spi_transfer() são de acordo com a verificação realizada nas funções de otimização.

3.4.1 Configuração SPI

Para a configuração da interface SPI (*Serial Peripheral Interface*) na raspberry, foi utilizada a biblioteca bcm2835. Ela fornece acesso ao GPIO e outras funções de I/O no chip *Broadcom BCM2835*, usado na raspberry PI.

Os comandos de inicialização e configuração utilizados foram:

- bcm2835_init(): inicializa a biblioteca abrindo "/dev/mem"(se inicializado como root) ou "/dev/gpiomem"(se não inicializado como root) e obtendo ponteiros para a memória interna para os registradores do dispositivo BCM 2835.

- bcm2835_spi_begin(): inicia as operações SPI.

- bcm2835_spi_setDataMode(): ativa os dados no SPI e define a polarização e fase do sinal de *clock*.

- bcm2835_spi_setClockDivider(): especifica o divisor usado para definir a frequência de *clock* desejada.
- bcm2835_gpio_fsel(): define a função de selecionar o registro para o pino fornecido, podendo atuar como Input ou Output.

3.4.2 Algoritmo *Hill Climb*

O algoritmo utilizado foi baseado no algoritmo *Hill Climb* para localizar os valores ótimos dos DTCs e atingir a maior supressão da portadora. O *Hill Climb* é uma técnica de busca local que inicia com uma solução aleatória do problema e segue iterativamente tentando encontrar um máximo ou mínimo (ótimo local) de uma função objetivo.

A técnica *Hill Climb* é usada principalmente para resolver problemas computacionalmente difíceis. Ele observa somente o estado atual e o estado futuro imediato. Portanto, essa técnica é eficiente na memória, pois não mantém uma árvore de pesquisa.

Essa técnica tem seu desempenho máximo para problemas convexos, onde o mínimo ou máximo local é também o mínimo ou máximo absoluto. Para outros problemas, esse algoritmo pode não ser o mais eficiente, onde encontrará máximos locais (soluções que não podem ser melhoradas por quaisquer configurações vizinhas). A Fig. 15(3.4.2) a seguir ilustra melhor como esse problema pode ocorrer.

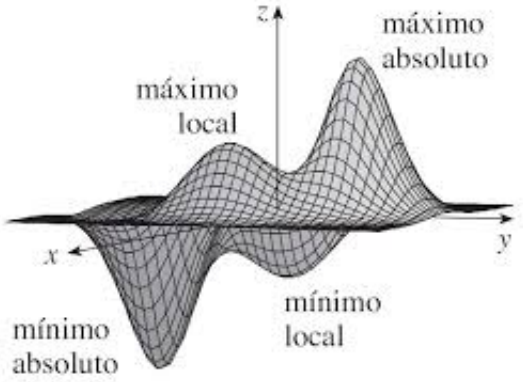

Figura 15 – Ilustração mínimo e máximo absoluto e local.

Com o intuito de minimizar as chances de o algoritmo ficar preso em um mínimo ou máximo local, foi implementado uma adaptação aumentando o raio de busca entre os estados. O algoritmo utilizado foi ajustado para realizar a verificação não só nos primeiros estados vizinhos, mas também em segundos e terceiros estados após e anteriores ao estado inicial, deste modo, o algoritmo leva um tempo maior para a verificação, mas a probabilidade de alcançar o valor ótimo é maior.

4 Resultados e Discussão

4.1 Simulações

4.1.1 Supressão de Portadora

Para a simulação do sistema de otimização foi projetado no software ADS o seguinte esquemático mostrado na Fig. 16(4.1.1).

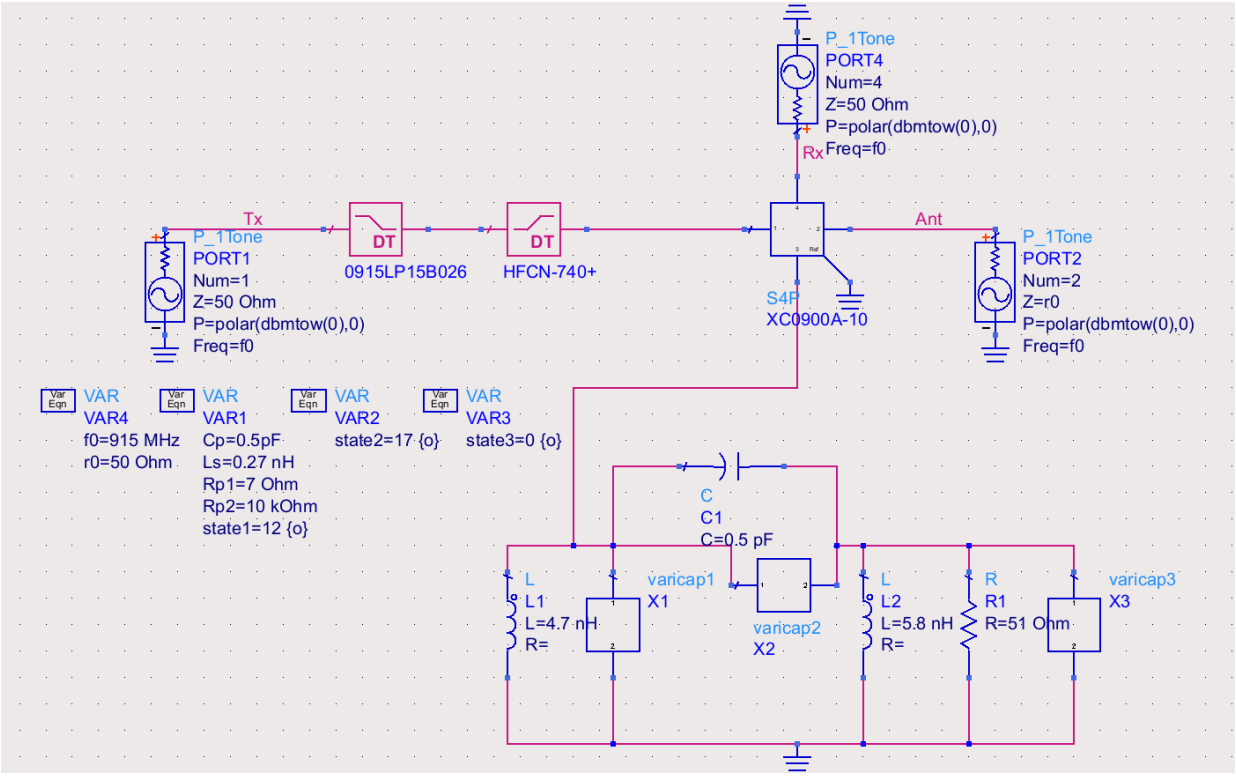

Figura 16 – Esquema elétrico para supressão de portadora.

No circuito projetado foi utilizado o acoplador direcional XC0900P-10S, com um fator de acoplamento de 10dB. Para esse caso, foi utilizado o arquivo *touchstone* S4P no software ADS referente a esse componente para criar um bloco que tenha uma resposta equivalente ao acoplador direcional. Os componentes X1, X2 e X3 representam os DTCs mostrado previamente na Fig. 2 (2.1).

Os filtros utilizados são necessários para atenuar os harmônicos de frequências não desejadas que são geradas previamente pelo estágio de amplificação, permitindo apenas frequências na faixa de UHF para aplicações em RFID.

Na porta 3 do acoplador direcional é conectado o circuito de impedância variável, o que permite melhorar adaptativamente a diretividade virtual do dispositivo de aco-

plamento. O circuito consiste em 3 capacitores ajustáveis digitalmente (DTC) e alguns componentes discretos.

Realizando a simulação desse sistema para uma frequência central f0 de 915MHz e deixando inicialmente as variáveis de estado (*state1, state2, state3*) que regulam os DTCs no valor inicial, pode-se observar no gráfico mostrado na Fig. 17(4.1.1) uma atenuação do sinal da porta 2, referente a antena, para a porta 4, caminho para Rx na leitora, de apenas 11.1 dB.

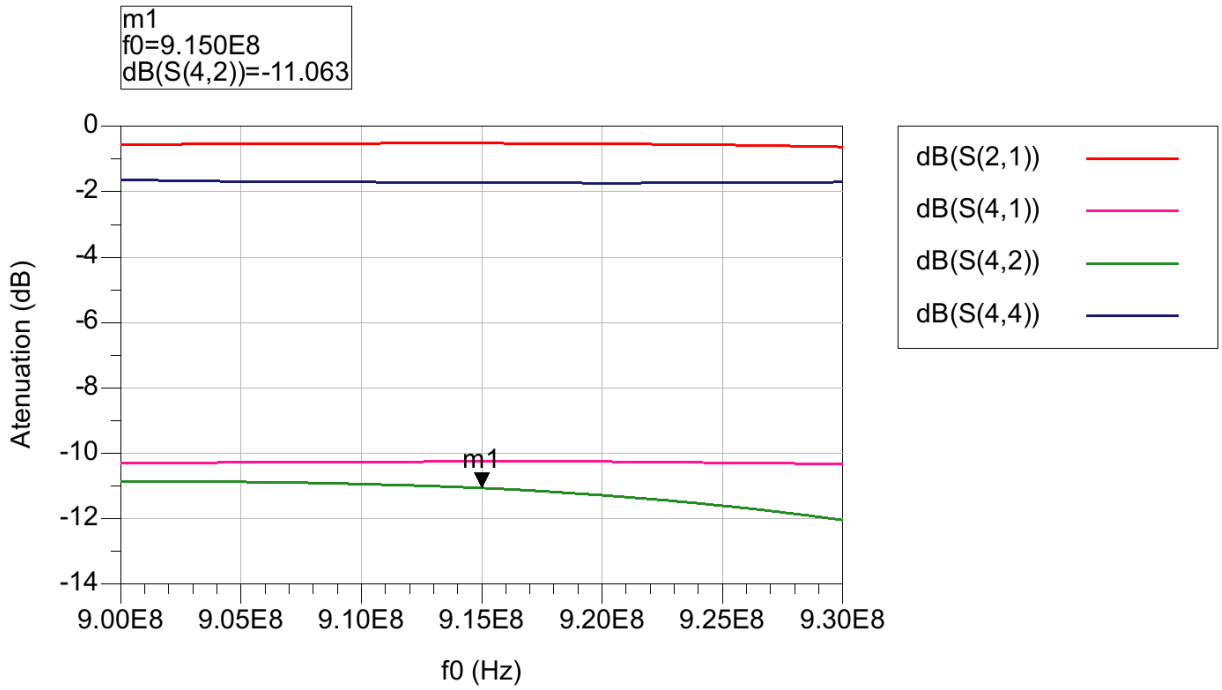

Figura 17 – Atenuação no sinal de portadora sem otimização.

Logo após foi realizado a simulação de otimização do sistema, onde o software varre todas as possibilidades de combinação das variáveis de estado que regulam os DTCs. Como cada um possui individualmente 32 valores diferentes de capacitância e resistência, então foram varridas, no total, 32768 (32x32x32) combinações diferentes a fim de encontrar uma combinação onde a supressão da portadora seja a maior possível.

Na figura 18 (4.1.1) é mostrado o resultado da simulação após essa otimização, indicando uma atenuação de 59dB no sinal de portadora.

A atenuação mais alta pode ser atingida com os seguintes valores de variável de estado: *state1* = 12, *state2* = 17 e *state3* = 0. Sabendo que os DTCs X1 e X3 estão na configuração "*shunt*"e o X2 na configuração série, corresponde a, respectivamente: Cs1 = 2.688pF, Cs2 = 2.793pF e Cs3 = 1.14pF.

Vale ressaltar que esses valores de capacitância encontrados para otimizar a supressão da portadora podem ser modificados para se adaptarem a outros sistemas com outras

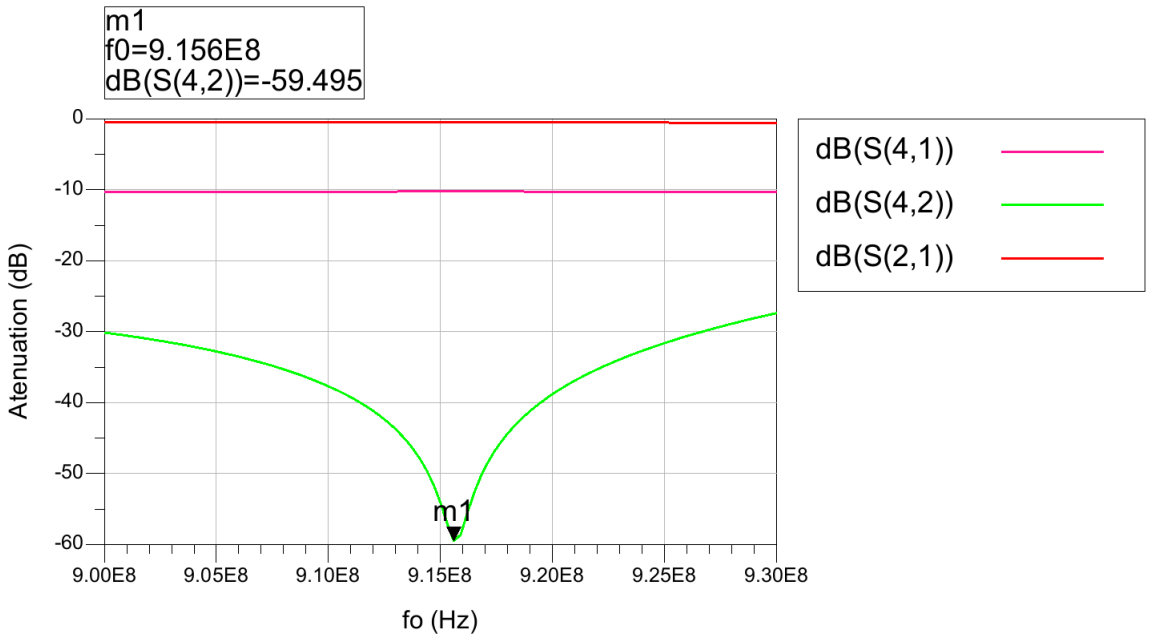

Figura 18 – Atenuação no sinal de portadora após otimização.

características, como por exemplo a antena ou as linhas de transmissão com impedâncias diferentes.

Como a figura 18 (4.1.1) mostra, existe sempre um valor mínimo de portadora para ser atingido com esse sistema. Entretanto, é relativamente fácil encontrar esse valor mínimo seguindo um algoritmo com os valores de capacitância inicialmente aleatórios. Quando o mínimo for encontrado, o algoritmo reajustará a impedância automaticamente se o vazamento da portadora for alterado.

4.1.2 Casamento de impedância para *front-end* RF de RFID

Para a simulação do circuito de casamento de impedância, foi montado o seguinte circuito:

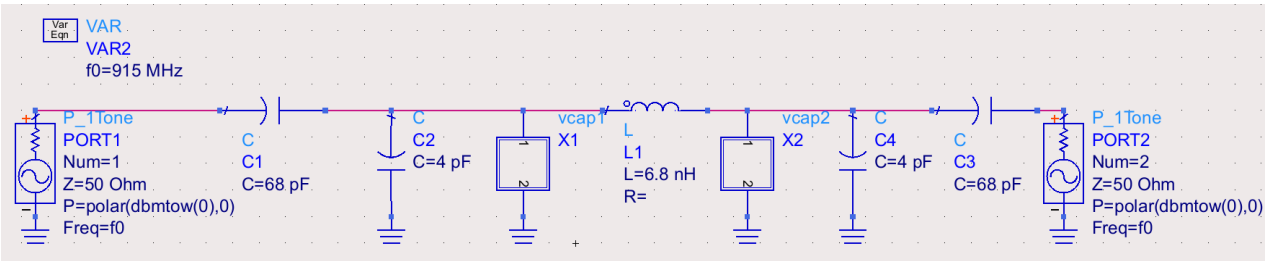

Figura 19 – Esquemático do circuito de casamento de impedância.

Nota-se que a configuração dos DTCs estão na configuração "*shunt*"e os valores de impedância nos terminais usados estão perfeitamente casados, ambos com 50Ω. Simulando um sistema com as antenas casadas.

Para demonstrar o funcionamento do sistema de casamento de impedância foi modificado os valores das impedâncias dos terminais que correspondem às antenas com o intuito de forçar um descasamento. A simulação desse sistema descasado pode ser vista na Fig. 20(4.1.2).

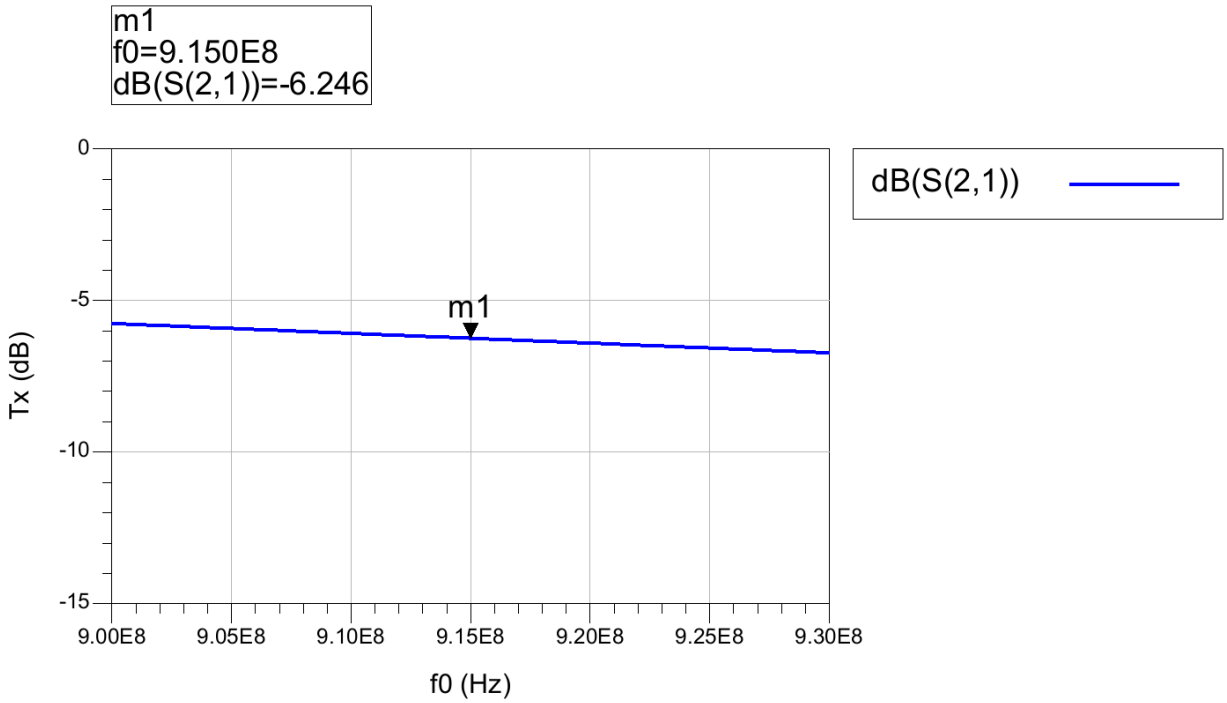

Figura 20 – Simulação circuito descasado com 10Ω e 100Ω nos terminais de impedância.

Os valores de impedância utilizado nos terminais foram, respectivamente, 10Ω e 100Ω.

Com o sistema descasado e as variáveis de estado dos DTCs com valores aleatórios, pode-se observar uma perda considerável na transmissão S_{21} de 6.2dB.

A partir disso, foi realizado a simulação de otimização do software ADS para encontrar valores de capacitância que representem uma impedância capaz de reduzir ao máximo essa perda no sinal de transmissão devido ao descasamento das antenas.

Pode ser observado na figura 21 (4.1.2) que a perda no sinal de transmissão foi reduzido consideravelmente de 6.2dB para apenas 0.4dB. Os valores de variável de estado dos DTCs encontrados para otimizar o sistema foram $state1 = state2 = 0$, que correspondem a capacitância de 1.14pF, já que estão na configuração "*shunt*".

Outra configuração foi realizada para simular não só o descasamento nos terminais de impedância, mas também foram adicionadas linhas de transmissão com fatores de impedâncias diferentes, o que causa um descasamento no sistema e prejudica a transmissão. O esquemático pode ser observado na Fig. 22(4.1.2).

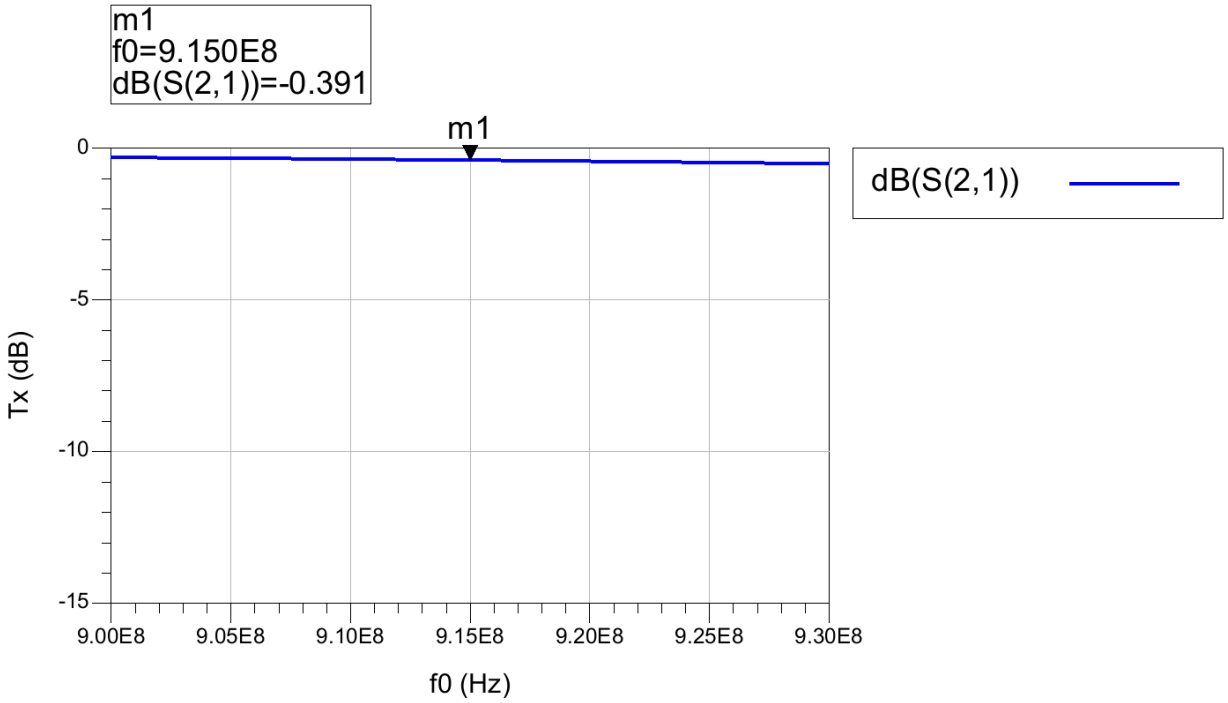

Figura 21 – Simulação do sistema de casamento entre 10Ω e 100Ω otimizado.

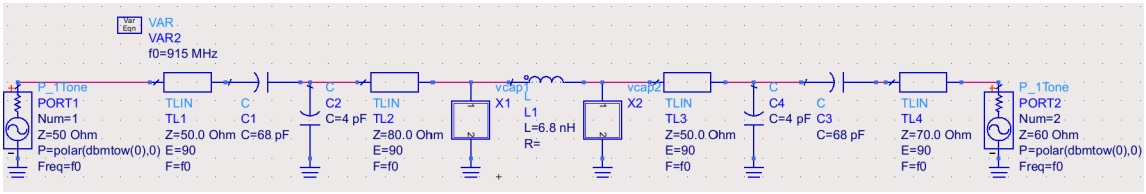

Figura 22 – Esquemático com linhas de transmissão e terminais descasados.

A simulação desse esquemático mostrado na Fig. 23(4.1.2) tem uma perda no sinal de transmissão S_{21} de 7.6dB.

Para encontrar os valores ótimos de capacitância para minimizar a perda na transmissão devido aos valores de impedância distintos dos elementos do circuito, foi realizado novamente a otimização do software ADS para varrer todas as possibilidades de combinação. O resultado obtido dessa otimização para esse circuito pode ser observado na Fig. 24(4.2).

O sinal de transmissão S_{21} foi otimizado com os seguintes valores das variáveis de estado: $state1 = 9$ e $state2 = 19$, que corresponde a capacitâncias de 2.301pF e 3.591pF respectivamente. E a perda ocorrida na transmissão que era de exatamente 7.6dB caiu para 0.6dB.

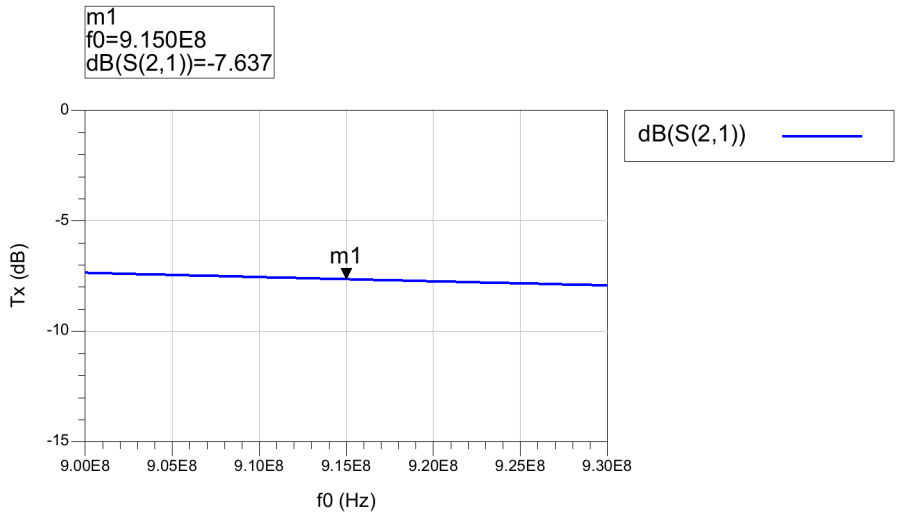

Figura 23 – Simulação do esquemático com linhas de transmissão e terminais descasados.

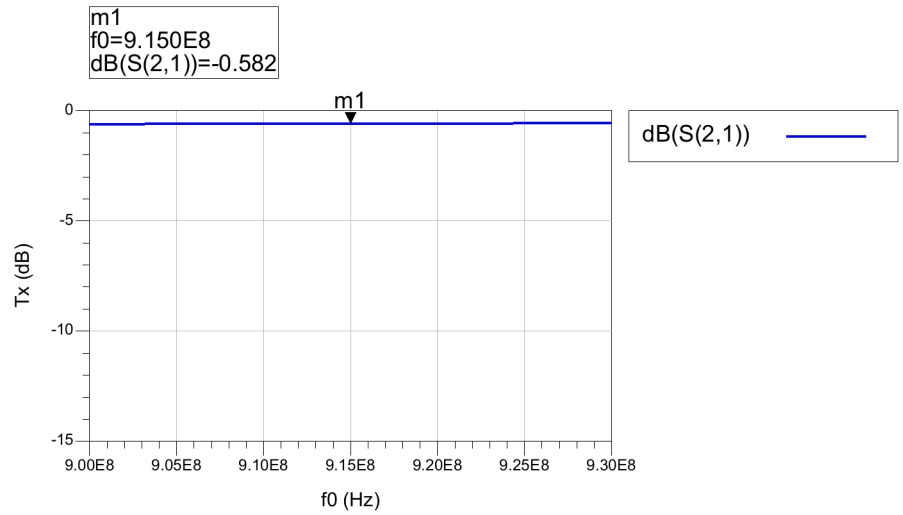

Figura 24 – Otimização do esquemático com linhas de transmissão e terminais descasados.

4.2 Medições

Após a finalização da montagem do sistema, foram realizadas as medições para a validação do projeto. A configuração da montagem pode ser observada na figura 25.

Figura 25 – Montagem do projeto.

Pode-se observar as ligações realizadas entre as placas RF, raspberryPi e o RDS. O cabo ethernet saindo da raspberry e o cabo USB saindo do RDS estao ligados diretamente ao computador onde está rodando o programa Gnuradio Companion e o código do servidor na comunicação TCP/IP.

Com todas as conexões realizadas, o sinal de entrada inicial obtido é demonstrado na figura 26 a seguir.

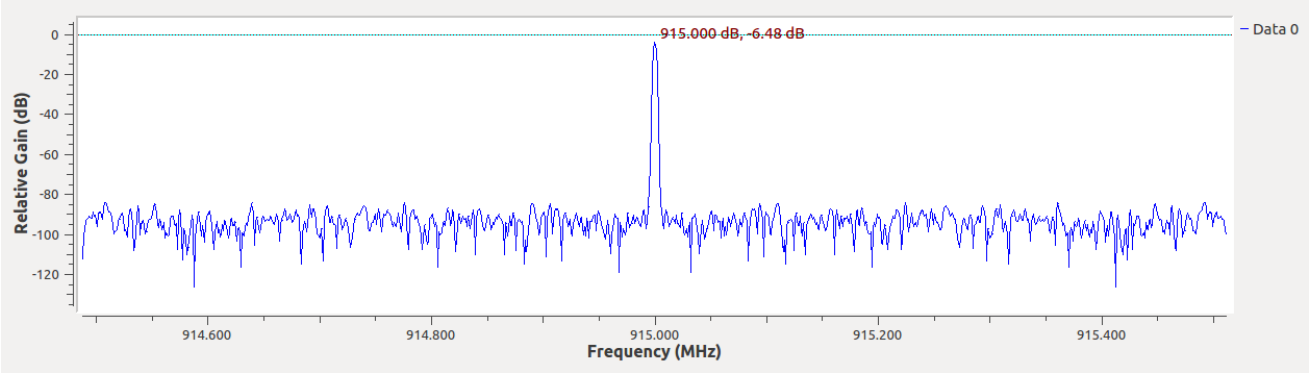

Figura 26 – Sinal de entrada sem otimização.

O sinal de entrada obtido pelo RDS, na frequência central de 915MHz, está com uma potência de -6dB. E foi dado, então, inicio ao código de otimização do sistema. Após o término da execução do programa, obteve-se o sinal de transmissão otimizado, que é mostrado na figura 27.

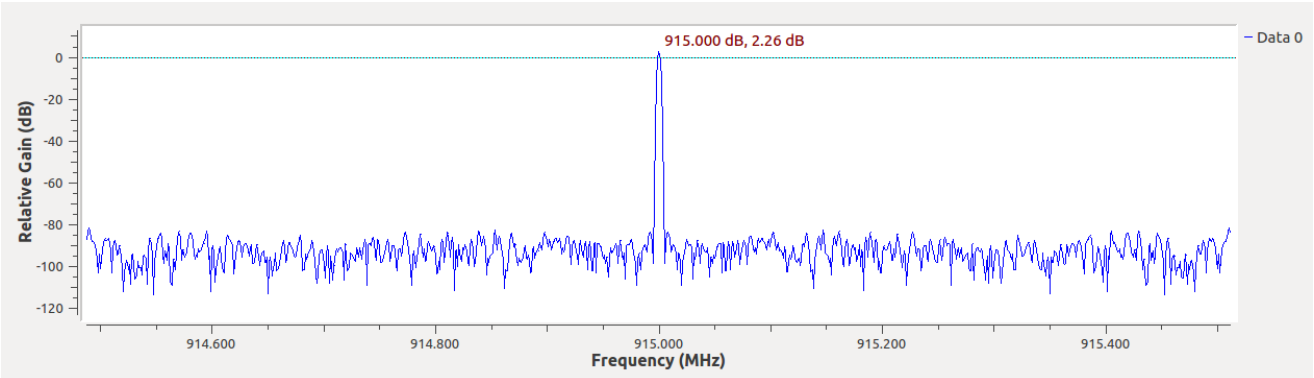

Figura 27 – Sinal de entrada com otimização.

Com esse resultado, pode-se observar que ocorreu uma melhoria de aprximadamente 8dB no sinal de transmissão.

5 Considerações Finais

Neste projeto, um método para suprimir o auto-bloqueio em leitoras de RFID de alta potência é apresentado. O método descrito usa o acoplador direcional como um dispositivo diretivo para separar os sinais de entrada dos sinais de saída. Adicionando um sistema de impedância ajustável à porta livre é possível obter um alto isolamento entre o receptor e o transmissor. Propõe-se um circuito de impedância ajustável capaz de cobrir o plano complexo completo dos coeficientes de reflexão da antena e dos sinais de vazamento devido à imperfeição do acoplador com o uso apenas de capacitâncias variáveis.

Essa estrutura do circuito de supressão de portadora é simples e de baixo custo. A análise teórica mostra que este esquema pode aumentar altamente o efeito de isolamento, e as simulações realizadas comprovam isso.

Este método também cancela o ruído intrínseco da portadora sem adicionar ruído não correlacionado. A solução barata e fácil de implementar isola o sinal de portadora em no mínimo 47dB. Isso é 10 dB a mais do que o necessário para os chips de leitura atuais e ainda dá espaço para uma melhoria no alcance de leitura.

Além de proporcionar todas essas vantagens na supressão de portadora nos sistemas de RFID, esse esquema também é capaz de realizar ajustes nas características internas do sistema com o intuito de realizar casamento de impedância entre as comunicações, deste modo, reduzindo a perda não desejada nos sinais de transmissão. E vale ressaltar que a pequena quantidade de componentes usados e um consumo muito baixo de área tornam esse sistema um forte candidato para usar em soluções altamente integradas.

Por fim, vale ressaltar que o resultado obtido nas medições finais com o sistema completamente integrado não foi o resultado ideal para o projeto, devido aos problemas encontrados no processo de modificação da placa RF, como a ligação realizada para as trilhas RF passarem pelo sistema de otimização feito apenas com um pedaço de solda, sendo que o ideal seria o uso de um capacitor de 100nF. Isto ocorreu devido à falta de equipamentos necessários para soldar componentes smd 0402, que são extremamente pequenos.

Contudo, mesmo com as ligações realizadas apenas com uma solda, o que gera uma perda na transmissão do sinal devido ao descasamento, o sistema se provou funcional e otimizou o sinal de transmissão que anteriormente era de aproximadamente -6dB para um sinal de aproximadamente 2dB, com isso, validando o projeto realizado.

Referências

BAI, P.; YIN, Y.; YANK, X. A novel rx-tx front-ends for passive rfid reader with high isolation. p. 332–335, 2007. Nenhuma citação no texto.

BRAUNER, T.; ZHAO, X. A novel carrier suppression method for rfid. v. 19, p. 128–130, 2009. Citado na página 33.

CURTY, J.-P. et al. Design and optimization of passive uhf rfid systems. Springer Science Business Media, 2007. Citado na página 23.

FINKENZELLER, K. Rfid handbook: Fundamentals and applications in contactless smart cards and identification. 2003. Citado na página 23.

KIM, W.-K. et al. A passive circulator for rfid application with high isolation using a directional coupler. p. 196, 2006. Citado na página 34.

MAYORDOMO, I.; BERNHARD, J. Implementation of an adaptive leakage cancellation control for passive uhf rfid readers. p. 121 – 127, 2011. Citado na página 33.

PEREGRINE SEMICONDUCTOR. *Digitally Tunable Capacitor (DTC) 100-3000MHz*. [S.l.]. Nenhuma citação no texto.

PURSULA, P.; KIVIRANTA, M.; SEPPä, H. Uhf rfid reader with reflected power canceller. v. 19, p. 48–50, 2009. Citado na página 33.

VILLAME, D. P.; MARCIANO, J. S. Carrier suppression locked loop mechanism for uhf rfid readers. p. 141 – 145. Nenhuma citação no texto.

Anexos

ANEXO A – Primeiro Anexo

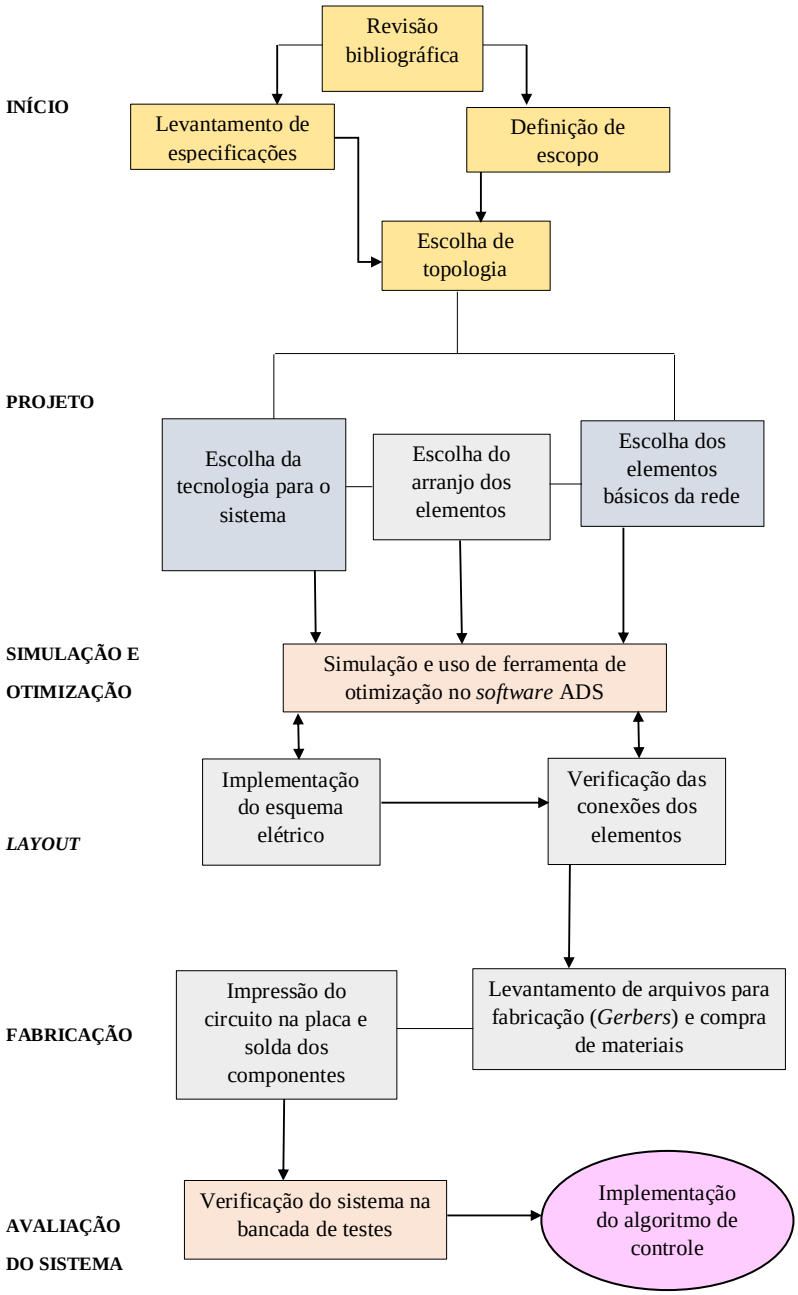

Figura 28 – Organograma para o projeto.

ANEXO B – Segundo Anexo

```
/*
                              .--- L ---.
                              |         |
  in ----.----.----- L --.--C_len--.--.-----.-----.----- out
         |    |                      |     |     |
       C_in   L                    C_out   L     R
         |    |                      |     |     |
        ___  ___                    ___   ___   ___
         _    _                      _     _     _
         ,    ,                      ,     ,     ,
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <unistd.h>
#include <bcm2835.h>
#include <math.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
#include <arpa/inet.h>

/** define to identify Cin cap of tuner. */
#define TUNER_CIN       0x01
/** define to identify Clen cap of tuner. */
#define TUNER_CLEN      0x02
/** define to identify Cout cap of tuner. */
#define TUNER_COUT      0x04

#define senTune1 RPI_GPIO_P1_13 //GPIO 27
```

```
#define senTune2 RPI_GPIO_P1_15 //GPIO 22
#define senTune3 RPI_GPIO_P1_16 //GPIO 23

float GetReflectedPower(FILE *f) {
  float v;
  fread((void*)(&v), sizeof(v), 1, f);
  return v;
}

typedef struct {
    uint8_t cin;
    uint8_t clen;
    uint8_t cout;
    float reflectedPower;
} TunerParameters;

typedef struct {
    int senTune1;
    int senTune2;
    int senTune3;
} TunerConfiguration;

void create_socket(){
    int sockfd = 0;
    struct sockaddr_in serv_addr;
    int bytesReceived = 0;
    char recvBuff[1024];
    memset(recvBuff, '0', sizeof(recvBuff));

    /* Create a socket first */
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0))< 0)
    {
        printf("\n Error : Could not create socket \n");
        return;
    }
    /* Initialize sockaddr_in data structure */
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000); // port
    char ip[12] = "192.168.0.2";
```

```
//strcpy(ip,"192.168.0.2");
serv_addr.sin_addr.s_addr = inet_addr(ip);

/* Attempt a connection */
if(connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))<
{
    printf("\n Error : Connect Failed \n");
    return;
}

printf("Connected to ip: %s : %d\n",inet_ntoa(serv_addr.sin_addr),ntc

/* Create file where data will be stored */
FILE *fp;
    char fname[7] = "Tx.bin";
    read(sockfd, fname, 256);

    printf("File Name: %s\n",fname);
    printf("Receiving file...");
     fp = fopen(fname, "wb");
    if(NULL == fp)
    {
     printf("Error opening file");
     return;
    }

/* Receive data in chunks of 256 bytes */
while((bytesReceived = read(sockfd, recvBuff, 256)) > 0)
{
    fflush(stdout);
    fwrite(recvBuff,1,bytesReceived,fp);
}

if(bytesReceived < 0)
{
    printf("\n Read Error \n");
}
printf("\nFile OK....Completed\n");
```

```
    fclose(fp);
}

float tunerGetReflected(void)
{
    FILE *fs;
    float r = 0;

    create_socket();

    fs = fopen("Tx.bin", "rb"); // read binary mode
    if (fs == NULL)
    {
       perror("Error while opening the file.\n");
       exit(EXIT_FAILURE);
    }

    //sleep(1);

    r = GetReflectedPower(fs);

    fclose(fs);
    return r;
}

void tunerSetCap(TunerConfiguration *config,  uint8_t component, uint8_t val
{
    /* set the SPI */

        bcm2835_spi_begin();
        bcm2835_spi_setBitOrder(BCM2835_SPI_BIT_ORDER_MSBFIRST);
// The default
        bcm2835_spi_setDataMode(BCM2835_SPI_MODE0);
// The default
        bcm2835_spi_setClockDivider(BCM2835_SPI_CLOCK_DIVIDER_65536); // The
    /*Using 3 GPIO as CS, Rpi has only 2 CS pins*/
        bcm2835_gpio_fsel(senTune1,BCM2835_GPIO_FSEL_OUTP);
        bcm2835_gpio_fsel(senTune2,BCM2835_GPIO_FSEL_OUTP);
        bcm2835_gpio_fsel(senTune3,BCM2835_GPIO_FSEL_OUTP);
```

```
    switch (component)
    {
        case TUNER_CIN:
            bcm2835_gpio_write(senTune1,HIGH);
            break;
        case TUNER_CLEN:
            bcm2835_gpio_write(senTune2,HIGH);
            break;
        case TUNER_COUT:
            bcm2835_gpio_write(senTune3,HIGH);
            break;
    }

    bcm2835_spi_transfer(val);

    switch (component)
    {
        case TUNER_CIN:
            bcm2835_gpio_write(senTune1,LOW);
            break;
        case TUNER_CLEN:
            bcm2835_gpio_write(senTune2,LOW);
            break;
        case TUNER_COUT:
            bcm2835_gpio_write(senTune2,LOW);
            break;
    }
}

void tunerSetTuning(TunerConfiguration *config, uint8_t cin_t, uint8_t cl
{
    if (config->senTune1
        tunerSetCap(config, TUNER_CIN, cin_t);
    }
    if (config->senTune2){
        tunerSetCap(config, TUNER_CLEN, clen_t);
    }
    if (config->senTune3){
```

```
        tunerSetCap(config, TUNER_COUT, cout_t);
    }
}

void tunerInit(TunerConfiguration *config)
{
    tunerSetTuning(config, 15, 15, 15);

    return;
}

uint8_t tunerClimbOneParam(TunerConfiguration *config,  uint8_t el, uint8_t *
{
    float refl,  start = *reflectedPower;
    int8_t dir = 0;
    int8_t add = 0;
    uint8_t improvement = 3;
    uint8_t bestelval;

    if (*maxSteps == 0)
    {
        return 0;
    }

    if ( *elval != 0 )
    {
        tunerSetCap(config, el, *elval - 1 );
        refl = tunerGetReflected();
        if ( refl <= *reflectedPower )
        {
            *reflectedPower = refl;
            dir = -1;
        }
    }

    if ( *elval < 31 )
    {
        tunerSetCap(config, el, *elval + 1 );
        refl = tunerGetReflected();
```

```
    if ( refl <= *reflectedPower )
    {
        *reflectedPower = refl;
        dir = 1;
    }
}

/* if it's possible we try to change the value by 2 and check what di
 * This improves tuning in the case of a local minima. */
if ( *elval > 1 )
{
    tunerSetCap(config, el, *elval - 2 );
    refl = tunerGetReflected();

    if ( refl <= *reflectedPower )
    {
        *reflectedPower = refl;
        dir = -1;
        add = -1;
    }
}

if ( *elval < 30 )
{
    tunerSetCap(config, el, *elval + 2 );
    refl = tunerGetReflected();

    if ( refl <= *reflectedPower )
    {
        *reflectedPower = refl;
        dir = 1;
        add = 1;
    }
}

*elval += add;
*elval += dir;
bestelval = *elval;
```

```
    if (dir!=0)
    {
        (*maxSteps)--;
        while ( improvement && *maxSteps )
        {
            if ( *elval == 0 ) break;
            if ( *elval == 31 ) break;
            tunerSetCap(config, el, *elval + dir );
            refl = tunerGetReflected();
            if ( refl <= *reflectedPower )
            {
                (*maxSteps)--;
                *reflectedPower = refl;
                *elval += dir;
                bestelval = *elval;
                improvement = 3; /* we don't want to stop when a local minima
                continue tuning for 3 more steps, even if they are worse than
                If it does not improve in this 3 steps, we will go back to th
            }else{
                improvement--;
            }
        }
    }
    *elval = bestelval;
    tunerSetCap(config, el, *elval);

    return start > (*reflectedPower);
}

void tunerOneHillClimb(TunerConfiguration *config,  TunerParameters *p, uint1
{
    uint8_t improvement = 1;

    tunerSetTuning(config, p->cin, p->clen, p->cout);

    p->reflectedPower = tunerGetReflected();

    while (maxSteps && improvement)
    {
```

```
        improvement = 0;
        if (config->senTune1){
            improvement |= tunerClimbOneParam(config, TUNER_CIN, &p->cin,
        }
        if (config->senTune2){
            improvement |= tunerClimbOneParam(config, TUNER_CLEN, &p->cle
        }
        if (config->senTune3){
            printf("senTune3 climbOneParam\n");
            improvement |= tunerClimbOneParam(config, TUNER_COUT, &p->cou
        }
    }

    tunerSetTuning(config, p->cin, p->clen, p->cout);
}

static const uint8_t tunePoints[3] = {5,16,26};

void tunerMultiHillClimb(TunerConfiguration *config,  TunerParameters *re
{
    TunerParameters p;
    uint8_t i,j;

    tunerSetTuning(config, res->cin, res->clen, res->cout);

    res->reflectedPower = tunerGetReflected();

    for( i = 0; i < 27; i++ )
    {
        j=i;
        if (config->senTune1)
            p.cin  = tunePoints[j%3]; j/=3;
        if (config->senTune2)
            p.clen = tunePoints[j%3]; j/=3;
        if (config->senTune3)
            p.cout = tunePoints[j%3]; j/=3;
        tunerOneHillClimb(config, &p, 30);

        if ( p.reflectedPower < res->reflectedPower )
```

```
        {
            *res = p;
        }
    }

    tunerSetTuning(config, res->cin, res->clen, res->cout);
}

int main()
{
    TunerConfiguration *config;
    typedef TunerParameters *pointer;
    pointer r;

    r = (pointer)malloc(sizeof(TunerParameters));

    if(!bcm2835_init())
        printf("bcm2835_init failed..\n");

    system("clear");

    int sockfd = 0;
    struct sockaddr_in serv_addr;
    int bytesReceived = 0;
    char recvBuff[1024];
    memset(recvBuff, '0', sizeof(recvBuff));

    /* Create a socket first */
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0))< 0)
    {
        printf("\n Error : Could not create socket \n");
        return 1;
    }
    /* Initialize sockaddr_in data structure */
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000); // port
    char ip[12] = "192.168.0.2";

    serv_addr.sin_addr.s_addr = inet_addr(ip);
```

```
/* Attempt a connection */
if(connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))<
{
    printf("\n Error : Connect Failed \n");
    return 1;
}

printf("Connected to ip: %s : %d\n",inet_ntoa(serv_addr.sin_addr),ntc

/* Create file where data will be stored */
FILE *fp;
    char fname[7] = "Tx.bin";
    read(sockfd, fname, 256);

    printf("File Name: %s\n",fname);
    printf("Receiving file...");
     fp = fopen(fname, "wb");
    if(NULL == fp)
    {
     printf("Error opening file");
     return 1;
    }

/* Receive data in chunks of 256 bytes */
while((bytesReceived = read(sockfd, recvBuff, 256)) > 0)
{
    fflush(stdout);
    fwrite(recvBuff,1,bytesReceived,fp);
}

if(bytesReceived < 0)
{
    printf("\n Read Error \n");
}
printf("\nFile OK....Completed\n");
fclose(fp);

tunerSetTuning(config, 15, 15, 15);
```

```
    tunerMultiHillClimb(config, r);

    bcm2835_spi_end();
    return 0;
}
```

ANEXO C – Terceiro Anexo

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>


struct sockaddr_in c_addr;
char fname[100];

void* SendFileToClient(int *arg)
{
      int connfd=(int)*arg;
      //printf("Connection accepted and id: %d\n",connfd);
      //printf("Connected to Clent: %s:%d\n",inet_ntoa(c_addr.sin_addr),n
       write(connfd, fname,256);

        FILE *fp = fopen(fname,"ab+");
        if(fp==NULL)
        {
            printf("File opern error");
            return 1;
        }

        /* Read data from file and send it */
        while(1)
        {
            /* First read file in chunks of 256 bytes */
            unsigned char buff[1024]={0};
            int nread = fread(buff,1,1024,fp);
```

```
            //printf("Bytes read %d \n", nread);

            /* If read was success, send data. */
            if(nread > 0)
            {
                //printf("Sending \n");
                write(connfd, buff, nread);
            }
            if (nread < 1024)
            {
                if (feof(fp))
                {
                    printf("End of file\n");
                    printf("File transfer completed for id: %d\n",connfd);
                }
                if (ferror(fp))
                    printf("Error reading\n");
                break;
            }
        }
printf("Closing Connection for id: %d\n",connfd);
close(connfd);
//shutdown(connfd,SHUT_WR);
sleep(2);
}

int main(int argc, char *argv[])
{
    int connfd = 0,err;
    pthread_t tid;
    struct sockaddr_in serv_addr;
    int listenfd = 0,ret;
    char sendBuff[1025];
    int numrv;
    size_t clen=0;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    if(listenfd<0)
        {
```

```
            printf("Error in socket creation\n");
            exit(2);
          }

    //printf("Socket retrieve success\n");

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(5000);

    ret=bind(listenfd, (struct sockaddr*)&serv_addr,sizeof(serv_addr));
    if(ret<0)
    {
      printf("Error in bind\n");
      exit(2);
    }

    if(listen(listenfd, 10) == -1)
    {
        printf("Failed to listen\n");
        return -1;
    }

if (argc < 2)
{
        //printf("Enter file name to send: ");
        strcpy(fname,"Tx.bin");
        printf("Enviando arquivo %s \n", fname);
        //printf("IP address is: %s\n", inet_ntoa(serv_addr.sin_addr));
        //printf("port is: %d\n", (int) ntohs(serv_addr.sin_port));
}
else
   strcpy(fname,argv[1]);

    while(1)
    {
        clen=sizeof(c_addr);
        //printf("Waiting...\n");
        //printf("IP address is: %s\n", inet_ntoa(serv_addr.sin_addr));
```

```
        connfd = accept(listenfd, (struct sockaddr*)&c_addr,&clen);
        if(connfd<0)
        {
          printf("Error in accept\n");
          continue;
        }
        err = pthread_create(&tid, NULL, &SendFileToClient, &connfd);
        if (err != 0)
            printf("\ncan't create thread :[%s]", strerror(err));
    }
     close(connfd);
     return 0;
}
```