INSTITUTO FEDERAL DE SANTA CATARINA

MARCOS VINICIOS PINHO

Analisador de Controle Remoto Utilizando RTL-SDR

São José - SC

dezembro/2018

ANALISADOR DE CONTROLE REMOTO UTILIZANDO RTL-SDR

Trabalho de conclusão de curso apresentado à Coordenadoria do Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Roberto Wanderley da Nóbrega

Coorientador: Ramon Mayor Martins

São José - SC

dezembro/2018

Marcos Vinicios Pinho
Analisador de Controle Remoto Utilizando RTL-SDR/ Marcos Vinicios Pinho. – São José - SC, dezembro/2018-
63 p. : il. (algumas color.) ; 30 cm.

Orientador: Roberto Wanderley da Nóbrega

Monografia (Graduação) – Instituto Federal de Santa Catarina – IFSC
Campus São José
Engenharia de Telecomunicações, dezembro/2018.

1. GNU Radio. 2. Rádio definido por *software*. 3. Controle remoto. I. Roberto Wanderley da Nóbrega. II. Instituto Federal de Santa Catarina. III. Campus São José. IV. Analisador de controle remoto utilizando RTL-SDR

MARCOS VINICIOS PINHO

ANALISADOR DE CONTROLE REMOTO UTILIZANDO RTL-SDR

Este trabalho foi julgado adequado para obtenção do título de Engenheiro de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 18 de dezembro de 2018:

Roberto Nóbrega

Prof. Roberto Wanderley da Nóbrega, Dr.
Orientador
Instituto Federal de Santa Catarina

Ramon Mayor Martins

Prof. Ramon Mayor Martins, Me.
Coorientador
Instituto Federal de Santa Catarina

Prof. Marcos Moecke, Dr.
Instituto Federal de Santa Catarina

Diego da Silva de Medeiros

Prof. Diego da Silva Medeiros, Me.
Instituto Federal de Santa Catarina

AGRADECIMENTOS

Agradeço a minha família, em especial a minha mãe, pelo amor, carinho e apoio incondicional.

Agradeço ao meu orientador Roberto Wanderley da Nóbrega e ao meu coorientador Ramon Mayor Martins, por todos os incentivos e ensinamentos.

Agradeço aos amigos que encontrei ao longo desse curso que de alguma forma contribuíram na minha jornada.

“Quem perde seus bens perde muito;

quem perde um amigo perde mais;

mas quem perde a coragem perde tudo.”

(Miguel de Cervantes)

RESUMO

Os controles remotos baseados em radiofrequência são amplamente utilizados no dia a dia. Porém com o constante uso desses aparelhos surge a necessidade de realizar a manutenção dos mesmos. Em vista disso, foi desenvolvido um analisador de controle remoto utilizando um dispositivo RTL-SDR (*Realtek software-defined radio*) na plataforma GNU Radio. O analisador tem como objetivo fornecer automaticamente informações sobre frequência de operação, frequência de *clock*, sequência de dados transmitida e o codificador utilizado para controles remotos que transmitem um sinal na modulação *on-off keying* (OOK). O sistema desenvolvido faz uso de blocos de processamentos de sinais disponíveis pelo GNU Radio e de blocos específicos programados em Python, com o objetivo de implementar técnicas e módulos como: detector de energia para o sensoriamento espectral, *zero crossing* para a determinação da frequência de *clock*, demodulador OOK para receber os sinais transmitidos, e decodificadores para os circuitos integrados mais utilizados no mercado. O trabalho apresenta também testes de validação e avaliação do desempenho do sistema, sendo apresentados os resultados e suas respectivas análises.

Palavras-chave: GNU Radio. Rádio definido por *software*. Controle remoto.

ABSTRACT

Radiofrequency-based remote controls are widely utilized nowadays. However with the constant use of these devices comes up the need of realize their maintenance. Therefore, a remote control analyzer using a RTL-SDR (Realtek software-define radio) receiver on the GNU Radio platform was developed. The analyzer aims to automatically provide information about operating frequency, clock frequency, transmitted data sequence and the utilized encoder of the remote controls that transmits an on-off-keying (OOK) signal. The developed system uses signals processing blocks available on GNU Radio and specific blocks implemented in Python, aiming to use techniques like: energy detector to spectral sensing, zero crossing to determine the clock frequency, OOK demulator to receive transmitted signals and decoders to the integrated circuits most utilized in the market. The work also presents validation tests and evaluation of the system performance, being presented the results and them respective analyzes.

Keywords: GNU Radio. Software-defined radio. Remote control.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GRC *GNU Radio Companion* 13

AWGN *additive white Gaussian noise* 32

FFT *fast Fourier transform* 32

PMT *polymorphic types* 40

NRZ *non-return-to-zero* 39

USB *Universal Serial Bus* 30

IQ *in-phase and quadrature* 30

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de controle remoto baseados em radiofrequência ou *remote keyless entry* (RKE), foram introduzidos na década de 80 (LAKE, 2001). Após isso, tornaram-se muito populares pela comodidade e segurança que esse sistema traz. Desse modo, foram empregados nas mais variadas funções, como por exemplo habilitar/desabilitar sistemas, travar/destravar portas de veículos, etc. (SUDA; LEHMER, 2004).

Com o uso constante desses dispositivos surge a necessidade de realizar a manutenção, podendo ser desde a calibração de componentes até a cópia do código para um novo controle. No entanto, para realizar essas manutenções de forma correta, é necessário o conhecimento de algumas características do sistema, como por exemplo a frequência de operação, o circuito integrado (codificador) utilizado, a sequência de informação, a frequência do *clock* do controle, entre outras.

Essas características são importantes para a compatibilidade do transmissor com o receptor, são elas que permitem o correto funcionamento do sistema. Segundo a VTE Tecnologia Eletrônica[1], a maioria dos receptores aceitam até 30% de variação na frequência de *clock* e toleram diferenças de até 2 MHz na frequência de operação do controle remoto. Em vista disso, é necessário o uso de algum equipamento que auxilie e consiga detectar essas informações para facilitar a manutenção.

Atualmente existem no mercado aparelhos que realizam as medições necessárias para a aplicação dessas manutenções, como por exemplo o Analisador Digital de Controle Remoto da empresa VTE Tecnologia Eletrônica. No entanto, este aparelho possui a impossibilidade de atualização do sistema adquirido, sendo compatível apenas com controles remotos que operam na faixa de frequência de 90 MHz a 655 MHz, usam modulação *on-off keying* (OOK) e utilizam um dos oito codificadores compatíveis com o aparelho.

Em vista disso, é proposto o desenvolvimento de um analisador de controle remoto baseado nos conceitos de *software-defined radio* (SDR), visando apenas os controles remotos que empregam um transmissor de radiofrequência e transmitem um sinal contendo um código de acesso, em uma certa frequência, utilizando a forma de modulação OOK.

Com a abordagem de utilizar um SDR, o projeto do analisador de controle remoto será desenvolvido majoritariamente em *software*, substituindo a tradicional implementação em *hardware* por uma mais flexível, deixando a implementação em *hardware* apenas para o receptor SDR. Utilizando o conceito de SDR é possível implementar modificações no sistema sem a necessidade de alterar nenhum componente físico, o que facilita a reconfiguração ou inserção de funcionalidades no sistema (SELVA et al., 2012).

Objetivando o baixo custo, será empregado o *dongle Realtek software-defined radio*

[1] Disponível em: <http://www.vtetecnologia.com.br/pages/produtos.html>.

(RTL-SDR), baseado no *chip* RTL2832U da Realtek[2]. A implementação do projeto será realizada utilizando uma plataforma de desenvolvimento gratuita e de código aberto, o GNU Radio, que permite projetar, simular e implantar sistemas SDR, e que fornece diversos blocos de processamento prontos (GNU RADIO, 2017b).

1.1 Objetivos

O projeto tem como objetivo principal o desenvolvimento de um analisador de controle remoto utilizando um dispositivo RTL-SDR. O projeto inclui os seguintes objetivos específicos a serem realizados:

- Implementar o módulo receptor.
- Encontrar a frequência do sinal transmitido.
- Identificar o codificador utilizado.
- Detectar a frequência do *clock* do transmissor.
- Detectar a sequência de bits transmitida (informação).
- Desenvolver uma interface gráfica para o usuário.

1.2 Organização do texto

O texto está organizado da seguinte forma: no Capítulo 2 é apresentado a fundamentação teórica, mostrando as principais técnicas e tecnologias utilizadas no trabalho. No Capítulo 3 está apresentado o desenvolvimento do projeto, descrevendo os passos necessários para a implementação do analisador de controles remotos. O Capítulo 4 apresenta os os testes aplicados no sistema e os resultados encontrados. O Capítulo 5 apresenta as conclusões finais e os trabalhos futuros sugeridos.

[2] Disponível em: <http://www.realtek.com.tw>.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentadas as principais tecnologias e conceitos aplicados no desenvolvimento deste trabalho. O texto está dividido em quatro grandes seções, a primeira seção apresenta informações relacionadas aos controles remotos, a segunda seção apresenta o modelo receptor, a terceira seção apresenta o conceitos e tecnologias sobre rádios definido por *software* e a quarta seção apresenta técnicas de sensoriamento espectral.

2.1 Controle remoto

Os controles remotos de interesse para esse trabalho são aqueles que utilizam um pequeno rádio transmissor, e enviam um código em uma certa frequência, sendo esse código gerado por um codificador de circuito integrado (MARNEWECK, 1996, p. 1). Esses dispositivos geralmente são empregados em variados sistemas de segurança, com a finalidade de travar/destravar portas, habilitar/desabilitar sistemas, etc. Para isso, um controle deve enviar um sinal contendo a informação de autenticação, e a unidade de controle do receptor deve processar o sinal, determinando se o comando foi autorizado ou não (SUDA; LEHMER, 2004).

2.1.1 Codificadores

Os circuitos codificadores utilizados apresentam basicamente três categorias: os de códigos fixos, os de códigos de salto e os de *learning code*. Os de códigos fixos são aqueles que sempre enviam o mesmo sinal de autenticação ao receptor, e possuem sua codificação definida manualmente, usando uma chave DIP[1] ou soldagem para definir qual o tipo de ligação nas entradas de dados/endereços (GATES, 2013). Um codificador que utiliza códigos de salto usa um algoritmo para gerar um sinal único a cada transmissão, mudando de forma imprevisível após uma transmissão, aumentando assim a segurança do sistema contra possíveis cópias (MICROCHIP TECHNOLOGY INC, 2001). Os codificadores *learning code* são aqueles que possuem uma codificação automática, copiando os códigos de controles que utilizam codificadores de código fixo ou mesmo de outros *learning code* (GATES, 2013).

Uma das funcionalidades propostas para o analisador de controle remotos é encontrar o código transmitido. Porém, o conhecimento sobre essa informação só é interessante para transmissores que utilizam códigos fixos ou *learning code*. Em vista disso, serão identificadas apenas as sequências de controles de códigos fixos. Para a implementação dessa funcionalidade é necessário o conhecimento sobre o funcionamento do codificador, como exemplo serão apresentados o funcionamento de três codificadores o HT12E, o HT6P20B e o HT6026, todos produzidos pela Holtek Semicondutor Inc[2].

[1] Uma chave *dual in-line package* (DIP) é um pequeno interruptor eletrônico manual, que possui uma série de pequenos interruptores entrada construídos em uma só placa. Disponível em: <https://www.usinainfo.com.br/diversos/dip-switch-chave-dip-4-vias-chds004-3071.html>.

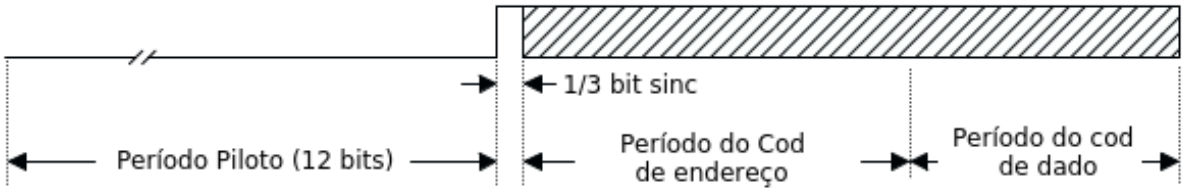
[2] Disponível em: <http://www.holtek.com>.

O codificador HT12E é capaz de codificar em dois estados diferentes 12 bits de entradas, resultando em $2^{12} = 4096$ diferentes palavras possíveis, sendo N bits para endereço e $12 - N$ bits de dados. Usualmente as entradas de endereço são setadas por uma chave DIP ou soldagem e as entradas de dados são selecionadas pelos botões do controle (HOLTEK SEMICONDUCTOR INC, 2000).

Um ciclo de transmissão utilizando o HT12E consiste no envio de quatro períodos diferentes, são transmitidos nesse ciclo os períodos de piloto, de sincronismo e os períodos de endereço/dados. A Figura 1 apresenta o formato desse ciclo, nela é possível observar que o piloto possui um período de 12 bits com a ausência da portadora na transmissão, já o sincronismo possui um período de 1/3 de bit transmitindo a portadora e por fim os períodos de endereço/dados com 12 bits (HOLTEK SEMICONDUCTOR INC, 2000).

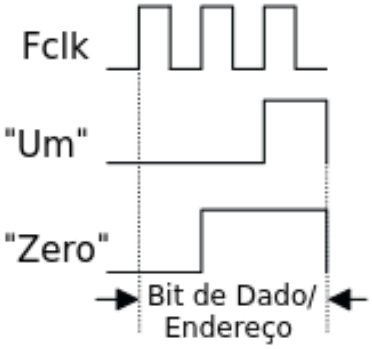
A Figura 2 apresenta os formatos de ondas possíveis após a codificação de cada bit de endereço/dado conforme seu estado lógico. O valor da frequência de *clock* é de tipicamente $f_{\text{clk}} = 3\,\text{kHz}$, o que resulta em um período de um bit de informação de 1 ms. A Figura 3 apresenta a forma de onda de uma possível sequência após a codificação (HOLTEK SEMICONDUCTOR INC, 2000).

Figura 1 – Formato de ciclo de transmissão – HT12E.



Fonte: Adaptado de (HOLTEK SEMICONDUCTOR INC, 2000).

Figura 2 – Formato de onda codificada – HT12E.

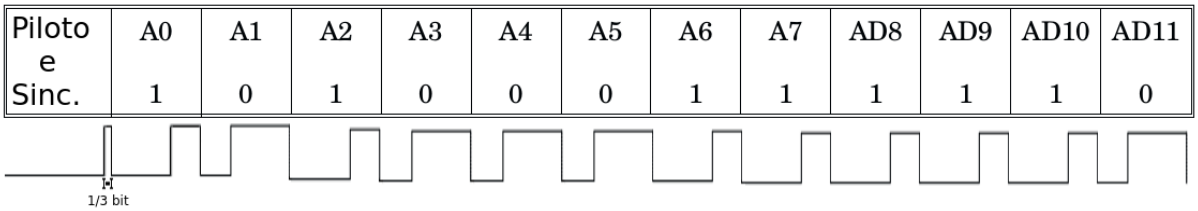


Fonte: Adaptado de (HOLTEK SEMICONDUCTOR INC, 2000).

O codificador HT6026 é capaz de codificar 9 entradas de informação em três estados lógicos diferentes, resultando em $3^9 = 19.683$ sequências diferentes. Esses 9 trits[3] consistem

[3] Um trit é a menor unidade de informação que pode ser armazenada em um sistema que faz uso da lógica ternária, ou seja, um trit faz referência a três estados lógicos possíveis. Para esse trabalho os três estados possíveis são: um, zero e aberto.

Figura 3 – Exemplo de transmissão utilizando o HT12E.

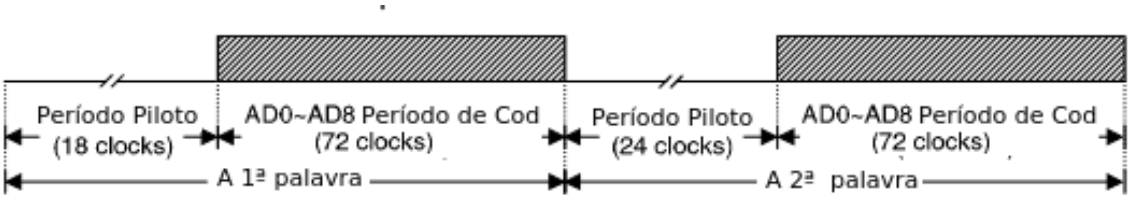


Fonte: Adaptado de (HOLTEK SEMICONDUCTOR INC, 2000).

em N entradas de endereço e $9 - N$ trits de dado. A informação transmitida consiste em duas sequências da mesma informação e dois períodos piloto, o primeiro piloto transmitido possui 18 *clocks* de duração, já o segundo possui 24 *clocks*. Esses períodos pilotos representam a ausência da portadora. A Figura 4 apresenta a duração e como estão separados esses períodos (HOLTEK SEMICONDUCTOR INC, 2009).

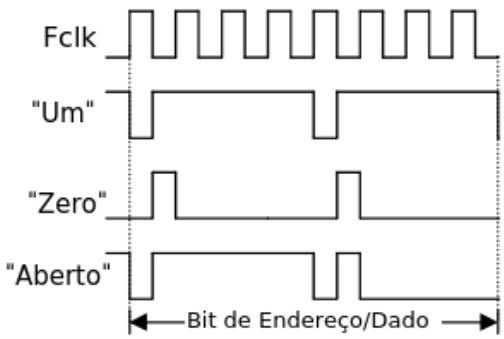
A Figura 5 apresenta os formatos de ondas possíveis após a codificação dos estados lógicos de cada trit de endereço/dado, sendo possível os valores de "um", "zero" e "aberto". Como pode ser observado nessa figura, um trit de informação tem duração de 8 *clocks*. O tempo de duração de um *clock* é 45 µs levando em consideração a frequência de oscilação típica $f_{\text{clk}} = 22\,\text{kHz}$. O valor de f_{clk} depende de uma expressão que leva em consideração dois resistores e um capacitor entre as entradas do oscilador. Uma possível transmissão realizada pelo HT6026 está apresentada na Figura 6, em que o valor Z significa o estado "aberto" (HOLTEK SEMICONDUCTOR INC, 2009).

Figura 4 – Formato de ciclo de transmissão – HT6026.



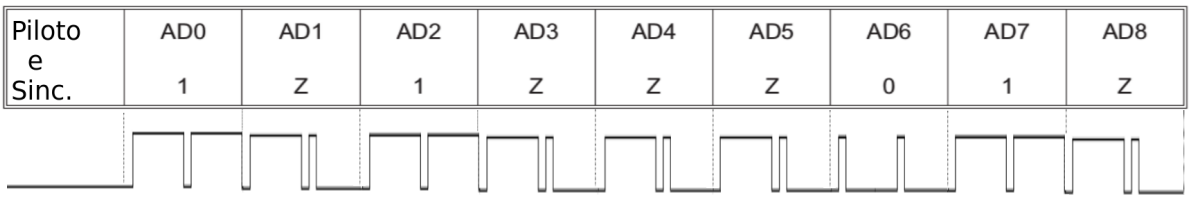
Fonte: Adaptado de (HOLTEK SEMICONDUCTOR INC, 2009).

Figura 5 – Formato de onda codificada – HT6026.



Fonte: Adaptado de (HOLTEK SEMICONDUCTOR INC, 2009).

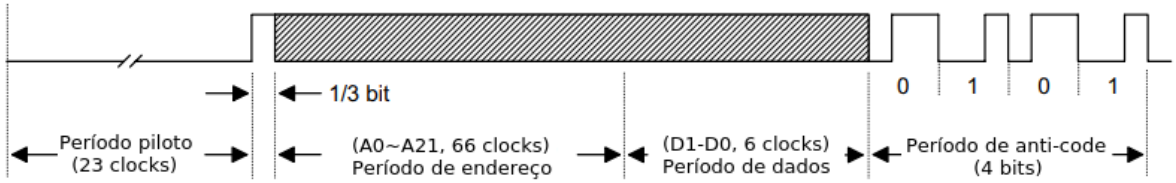
Figura 6 – Exemplo de transmissão utilizando o HT6026.



Fonte: Adaptado de (HOLTEK SEMICONDUCTOR INC, 2009).

O codificador HT6P20B é capaz de codificar 24 bits de informação, gerando $2^{24} = 16.777.216$ sequências possíveis. Esse codificador tem o código programado na fábrica, e ele consiste em 22 bits de endereço e 2 bits de dados. A informação transmitida é dividida em 5 períodos, como apresentado na Figura 7, um período piloto com duração de 23 *clocks* com ausência da portadora, um período de sincronismo com a presença da portadora com duração de 1/3 de bit, os períodos de endereço/dado com duração total de 72 *clocks* e por fim um período de *anti-code*[4] com 12 *clocks* de duração. Assim como para o codificador HT12E, um bit de informação tem duração de 3 *clocks*, isso pode ser observado na Figura 2. Levando em consideração que a frequência tipica do oscilador é $f_{\text{clk}} = 3\,\text{kHz}$, cada *clock* tem um período de 0,333 ms e o período de bit é 1 ms (HOLTEK SEMICONDUCTOR INC, 2003).

Figura 7 – Formato de um ciclo de transmissão – HT6P20B.



Fonte: Adaptado de (HOLTEK SEMICONDUCTOR INC, 2003).

2.1.2 Modulação

Antes de transmitir uma mensagem de autenticação para o receptor é necessário preparar o sinal para o envio. Para isso é utilizado uma técnica conhecida como modulação. Segundo Rappaport (2008)

> Modulação é o processo de codificar informações a partir de uma fonte de mensagem de uma maneira adequada à transmissão. Isso geralmente envolve traduzir um sinal de mensagem na banda base para um sinal da banda de passagem em frequências que são muito altas comparadas com a frequência da banda base.

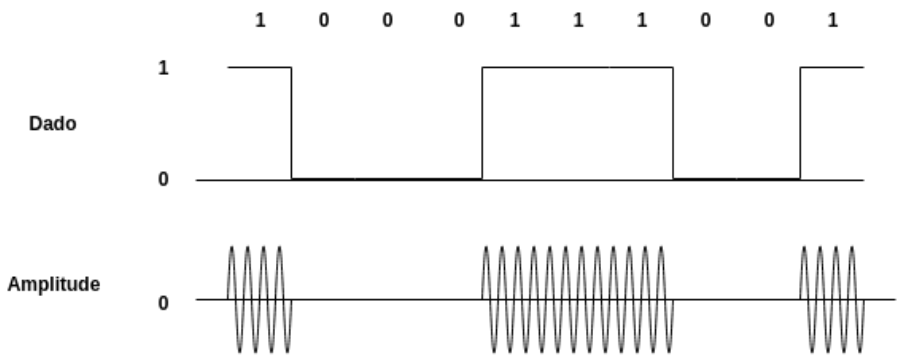
No caso dos controles remotos são utilizadas basicamente as técnicas de modulação *on-off keying* (OOK) e *frequency-shift keying* (FSK), as quais empregam demoduladores não-coerentes. O termo não-coerente refere a sistemas demoduladores que não utilizam o conhecimento sobre o

[4] O *anti-code* serve como um código de verificação. Para o caso desse codificador, sempre é enviado "0101".

valor absoluto da fase para detectar o sinal recebido, o que resulta em um sistema mais simples, porém com uma probabilidade de erro maior (SKLAR, 2001).

A modulação OOK é a mais utilizada nos controles remotos e a de maior interesse para o trabalho, ela é um caso especial da modulação *amplitude-shift keying* (ASK), na qual durante a transmissão de um zero a portadora não é transmitida. A Figura 8 apresenta a lógica da modulação OOK, que corresponde a dois tipos diferentes de formatos de ondas, um com valor de amplitude existente e outro com valor zero. Essa característica do OOK tem como vantagem conservar energia, o que é muito importante para controles remotos (ANTHES, 2007).

Figura 8 – Modulação OOK.



Fonte: Baseado em (SKLAR, 2001).

2.1.3 Frequência de operação

As faixas de frequências utilizadas pelos controles remotos seguem algumas considerações legais que variam em cada país. A Comissão Federal de Comunicações dos Estados Unidos, *Federal Communications Commission* (FCC), define a banda de 260 MHz a 470 MHz para ser utilizada em transmissões de comandos e controles de dados. Na regulamentação brasileira, definida pela Agência Nacional de Telecomunicações (Anatel), os controles remotos se encaixam no regulamento sobre equipamentos de radiocomunicação de radiação restrita[5]. Eles estão na categoria de dispositivos de operação periódica, que são sistemas que operam de forma descontínua em relação às características de duração da transmissão e dos períodos de silêncio regulares. Os limites gerais de emissão definidos pela Anatel para a faixa de 216 MHz a 960 MHz são de 200 $\mu V\,m^{-1}$ a uma distância de 3 m (ANATEL, 2017).

As duas frequências mais utilizadas no Brasil e nos Estados Unidos são as de 315 MHz e 433,92 MHz (LINX TECHNOLOGIES, 2012). Outras duas frequências que estão amplamente disponíveis no mercado brasileiro, são as frequências de 292 MHz e 299 MHz.

Existem dois modelos de controle relacionados a frequência de operação, um deles é o de frequência fixa, enquanto o outro utiliza o modelo de frequência ajustável. Exemplos de modelos de frequência fixa são aqueles que utilizam ressonadores baseados em *surface acoustic*

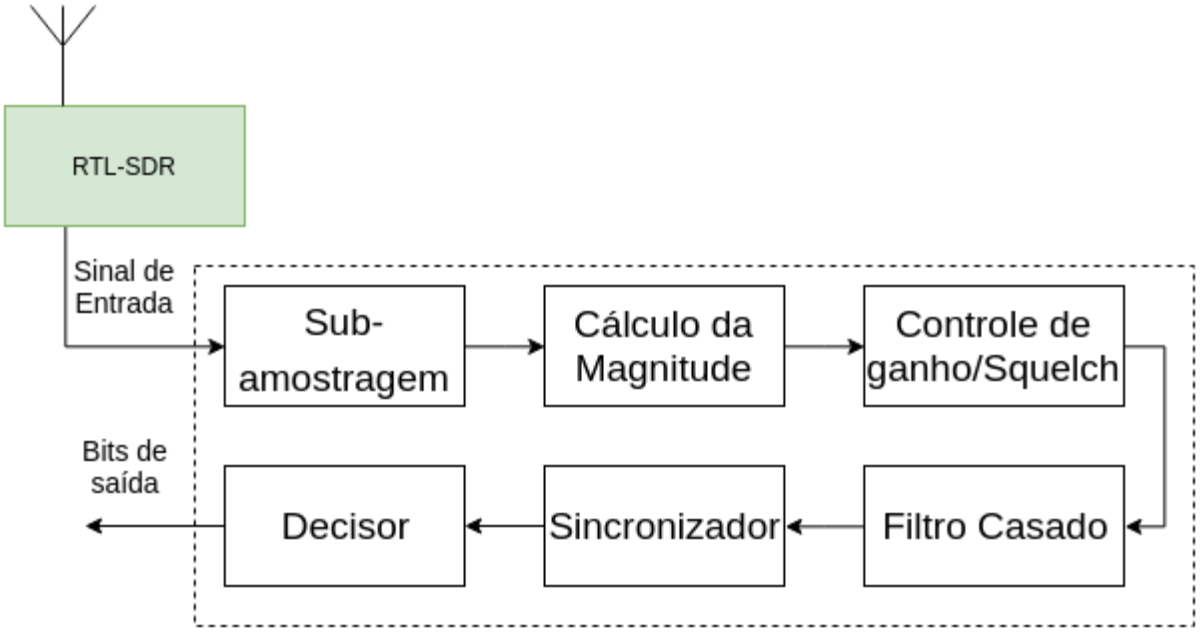
[5] Resolução nº 680, de 27 de junho de 2017. Disponível em: <http://www.anatel.gov.br/legislacao/resolucoes/2017/936-resolucao-680>.

wave (SAW)[6], esses são mais fáceis de identificar a frequência de operação e não perdem a calibração com o tempo. O modelo de frequência ajustável usa circuitos indutor–capacitor, no qual a mudança da frequência central é feita utilizando um capacitor ajustável ou um indutor ajustável, esses são modelos mais simples e sofrem desvios na frequência de operação ao longo do tempo de uso (GATES, 2013).

2.2 Receptor

Para a recepção de um sinal de controle remoto com a modulação OOK e utilizando um *dongle* RTL-SDR, o qual converte internamente o sinal para uma frequência intermediária, é descrito um modelo demodulador/detector na Figura 9. Nesse modelo são apresentadas as etapas a serem realizadas após receber o sinal. Cada uma dessas etapas é descrita a seguir.

Figura 9 – Demodulador/detector OOK utilizado neste trabalho.



Fonte: Elaborada pelo Autor.

Subamostragem. É realizada para reduzir a taxa de amostragem de saída do RTL-SDR. O termo subamostragem ou decimação é o processo de diminuir a quantidade de amostras de sinal, criando uma sequência da taxa de amostragem mais baixa, mantendo apenas uma de N amostras do sinal original. (PRANDONI; VETTERLI, 2008).

Cálculo da magnitude. Nessa etapa é executada a função `abs()`, que retorna o valor absoluto do dado de entrada, transformando uma entrada de número complexo em ponto flutuante. Essa operação é realizada pelo motivo que a informação da modulação OOK está inserida na sua amplitude, ou seja, na sua magnitude.

[6] Um ressonador SAW é um dispositivo que compreende um material piezoelétrico que permite que um sinal elétrico seja convertido em um sinal acústico, que é uma onda acústica que se move na superfície do cristal.(MUKHTAR1 et al., 2016).

Controle de ganho / *squelch*. Para manter o sinal com um nível de saída predeterminado é utilizado um controle de ganho automático em conjunto com um *squelch*. O *squelch* suprime a saída quando o sinal esta abaixo de um dado limiar. Essa característica diminui a atividade de saída de dados (MICREL INC, 2015).

Filtro casado. Segundo (SKLAR, 2001) um filtro casado é um filtro linear desenvolvido para prover a máxima relação sinal ruído para uma sinal transmitido em um canal com ruído aditivo gaussiano (AWGN). Com o objetivo de prover sinais de saída com a máxima relação sinal-ruído no período de símbolo T, é levada em consideração uma propriedade básica do filtro casado, em que a resposta ao impulso do filtro de recepção é uma versão espelhada e atrasada do pulso de formatação utilizado no transmissor (SKLAR, 2001).

Sincronizador. O sincronizador é utilizado para extrair símbolos de um sinal digital assíncrono, permitindo sincronizar o receptor com os centros de uns e zeros do sinal recebido (ŠOLC, 2015).

Decisor. Nesse bloco é realizada a conversão do sinal do tipo float para bit, esse valor poderá ser bit “0” quando o sinal foi negativo ou “1” quando o sinal for positivo.

2.3 Rádio definido por *software*

O termo *software-defined radio* (SDR) foi definido pela primeira vez por Mitola em 1991 (SILVA et al., 2017). Mas também são encontrados diversas outras definições para descrever o que é SDR. Uma definição bem abrangente é a utilizada pelo SDR Forum, que define o termo como, “um rádio em que algumas ou todas as funções da camada física são definidas por software” (SDR FORUM, 2007). Segundo Selva et al. (2012)

> O conceito de rádio definido por *software* SDR permite substituir a tradicional implementação em *hardware* dos dispositivos de comunicação por uma implementação mais flexível, que faz uso de dispositivos programáveis controlados por *software*, como por exemplo, um computador pessoal ou um processador embarcado.

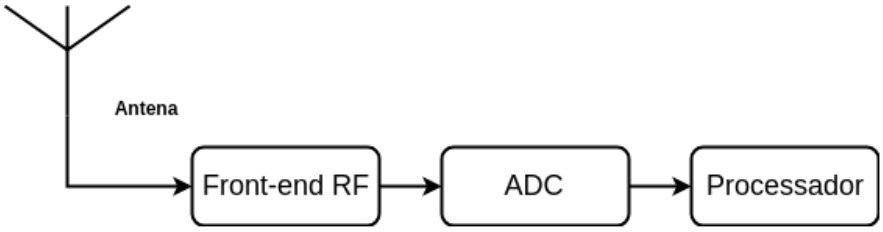
Mais especificamente substituindo a implementação de componentes que eram realizados em *hardware*, como por exemplo: filtros, amplificadores, moduladores/demoduladores e outros, por uma implementação em *software* (SILVA et al., 2017). Além disso, as principais características de operação podem ser modificadas em tempo de execução, facilitando a reconfiguração do sistema para realizar diferentes funções (SELVA et al., 2012).

2.3.1 *Hardware*

Uma arquitetura básica de um dispositivo SDR que opera como receptor é apresentado na Figura 10. Nela o *analog-to-digital converter* (ADC) e o *front-end* de radiofrequência são componentes que devem ser implementados em *hardware*, enquanto todas as outras operações do transceptor podem ser realizadas por unidades de processamento. O *front-end* de RF tem como função realizar a conversão de frequência do sinal de banda passante para a frequência

intermediária ou para banda base (BALL; NAIK; JENKINS, 2017). Segundo (NAGURNEY, 2009), o bloco processador pode utilizar uma *central processing unit* (CPU) de propósito geral, um *digital signal processor* (DSP) ou um *field-programmable gate array* (FPGA).

Figura 10 – Arquitetura básica de um receptor SDR.

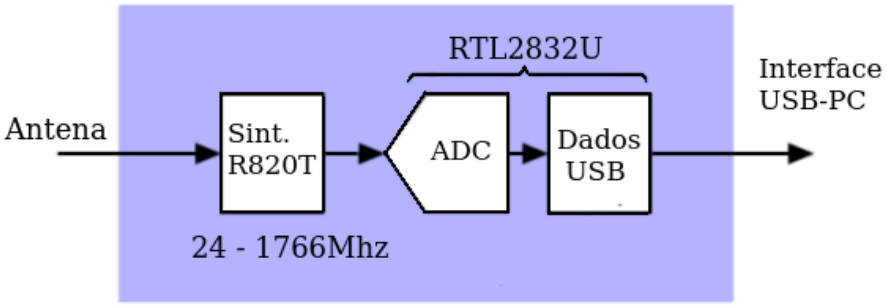


Fonte: Baseado em (SELVA et al., 2012).

O dispositivo utilizado nesse trabalho será um *dongle Realtek software-defined radio* (RTL-SDR). Esse dispositivo contém um sintonizador digital e um processador Realtek RTL2832U (FANAN et al., 2015), a Figura 11 apresenta um modelo de um RTL-SDR utilizando um sintonizador R820T (BALL; NAIK; JENKINS, 2017).

O RTL-SDR possui uma largura de banda[7] máxima de 3,2 MHz, sendo a recomendada de 2,8 MHz. Para o RTL-SDR a largura de banda equivale à taxa de amostragem, por exemplo, uma taxa de amostragem de 2 MS/s resulta em uma largura de banda de 2 MHz, isso acontece pois ele utiliza amostragem *in-phase and quadrature* (IQ)[8] com dois ADCs (LAUFER, 2015). O conversor analógico-digital utilizado possui 8 bits de de precisão, podendo quantizar o sinal em 256 diferentes escalas. O sintonizador digital é o responsável pela faixa de frequência, os dois sintonizadores mais comuns são o Rafael Micro R820T e o Elonics E4000. A faixa de frequência de ambos está apresentada na Tabela 1 (LAUFER, 2015). O RTL-SDR possui uma interface *Universal Serial Bus* (USB) utilizada para se conectar com o dispositivo onde será realizado o processamento.

Figura 11 – Modelo RTL-SDR.



Fonte: Baseado em (BALL; NAIK; JENKINS, 2017).

[7] A largura de banda é a faixa do espectro de frequência que pode ser analisado ao mesmo tempo.
[8] O termo IQ, em fase e quadratura, se refere a dois sinusoides que possuem a mesma frequência e estão defasados em 90 graus (EETECH, 2018).

Tabela 1 – Faixa de frequência dos sintonizadores.

Sintonizador	Frequência min	Frequência max
R820T	24 MHz	1766 MHz
E4000	52 MHz	2200 MHz

2.3.2 GNU Radio

O GNU Radio[9] é um conjunto de ferramentas de desenvolvimento gratuita e de código aberto, licenciado sob a GNU *General Public License* (GPL) versão 3. Essa ferramenta fornece diversos blocos de processamento de sinais para a implementação de um projeto de um SDR. Esses blocos fornecidos são tipicamente utilizados em sistemas de rádio, como por exemplo filtros, equalizadores, demoduladores, decodificadores e muitos outros (GNU RADIO, 2017b).

Por mais que existam diversos blocos prontos no GNU Radio, é possível implementar novos blocos, ou modificar as funcionalidades de um existente. Esses blocos personalizados são implementados utilizando as linguagens de programação C++ ou Python. De acordo com (SILVA et al., 2017), para as aplicações em GNU Radio é utilizado principalmente a linguagem Python, enquanto que os blocos de processamento que realizam operações criticas são desenvolvidos em C++, por questões de desempenho.

Além de permitir a criação e utilização de blocos de processamento, existe um ambiente de desenvolvimento integrado, o GRC, que permite o desenvolvimento de aplicações usando uma interface gráfica para o usuário (SILVA et al., 2017). O GRC é uma ferramenta gráfica para criar fluxogramas[10] e gerar código-fonte referente ao fluxograma implementado (GNU RADIO, 2017a). Com o GRC é possível utilizar GNU Radio sem escrever nenhuma linha de código, criando aplicativos de processamento de sinal apenas arrastando e soltando blocos (GNU RADIO, 2017b). Além disso, os novos blocos de processamento criados podem ser incluídos no GRC, o que permite utilizar o desenvolvimento rápido da interface gráfica e a análise estática do diagrama de módulos para encontrar erros de configuração (SILVA et al., 2017).

Ainda que o GNU Radio seja a ferramenta principal do trabalho, outras também serviram de auxílio no estudo e desenvolvimento do projeto. As que mais se destacam são o Gqrx[11] e o Inspectrum[12]. O Gqrx é um receptor SDR de código aberto que foi implementada utilizando o GNU Radio, ele suporta diversos dispositivos e pode operar recebendo sinais AM, FM, etc. (CSETE, 2018). O Inspectrum é uma ferramenta que permite analisar um sinal capturado fornecendo funções como plots de amplitude, frequência, etc. (WALTERS, 2018).

2.4 Sensoriamento espectral

O sensoriamento espectral é uma técnica utilizada para distinguir se um canal está ocupado ou livre. Duas técnicas de sensoriamento espectral muito utilizadas são: detecção por filtro casado e detecção de energia, elas serão apresentadas a seguir.

[9] Disponível em: <https://www.gnuradio.org>.
[10] Os blocos conectados são chamados de fluxograma.
[11] Projeto disponível em: <https://github.com/csete/gqrx>.
[12] Projeto disponível em: <https://github.com/miek/inspectrum>.

2.4.1 Detecção por filtro casado

O filtro casado é um filtro linear projetado para prover a máxima relação sinal-ruído e uma baixa probabilidade de detecção errônea (RIBAS, 2015). Entretanto, essa detecção executa operações coerentes, e requer o conhecimento de características do sinal transmitido, como por exemplo: tipo de modulação, formato de pacotes e forma de pulso, o que nem sempre é viável quando se recebe o sinal em diferentes padrões e modulações (ATAPATTU; TELLAMBURA; JIANG, 2014) (RIBAS, 2015). A detecção por filtro casado requer uma perfeita temporização e sincronismo, e sua complexidade de implementação é uma grande desvantagem (ATAPATTU; TELLAMBURA; JIANG, 2014).

2.4.2 Detecção de energia

Essa é a técnica considerada como a forma mais comum e simples de sensoriamento espectral, devido ao seu baixo custo computacional e a baixa complexidade de implementação (RIBAS, 2015). Ela é utilizada para estimar a energia associada a um sinal recebido sobre uma largura de banda específica. O valor estimado resultante é comparado com um limiar previamente determinado com o objetivo de identificar a presença ou ausência de um sinal (ATAPATTU; TELLAMBURA; JIANG, 2014).

A detecção de energia tipicamente opera sem o conhecimento prévio dos parâmetros do sinal transmitido, em vista disso, é classificada como um esquema não-coerente (RIBAS, 2015). A performance e confiabilidade dos resultados dessa técnica têm relação direta com algumas métricas que são definidas no sistema, são elas: o limiar de detecção, o número de amostras e a potência estimada do ruído (ATAPATTU; TELLAMBURA; JIANG, 2014).

De acordo com (RIBAS, 2015) os modelos convencionais consideram apenas um limiar de detecção. Para comparar com esse limiar duas hipóteses são definidas a seguir:

$$H_0 : y(t) = w(t) \tag{2.1}$$

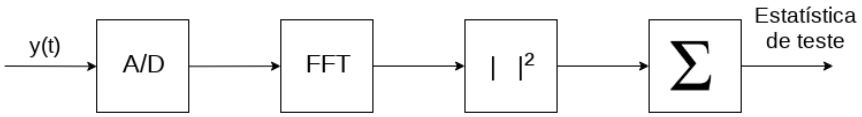
$$H_1 : y(t) = hx(t) + w(t), \tag{2.2}$$

em que H_0 é a ausência do sinal transmitido e H_1 é a presença do sinal. Como pode ser visto, $y(t)$ é o sinal recebido, $x(t)$ é o sinal original transmitido, $w(t)$ é o ruído branco gaussiano aditivo (*additive white Gaussian noise* (AWGN)), e h é o ganho do canal.

Existem duas formas principais em que pode ser desenvolvido o detector de energia, são elas: no domínio do tempo ou no domínio da frequência (RIBAS, 2015). A Figura 12 apresenta o diagrama de blocos para a implementação no domínio da frequência, que será a utilizada neste trabalho. Esse diagrama constitui de um ADC, um bloco onde é aplicada a *fast Fourier transform* (FFT), um bloco de elevação quadrática e por fim um bloco que soma as N magnitudes ao quadrado (ALVAREZ et al., 2011). A estatística do teste para esse caso pode ser representada por (ATAPATTU; TELLAMBURA; JIANG, 2014)

$$\Lambda = \sum_{n=1}^{N} |y(n)|^2. \tag{2.3}$$

Figura 12 – Detector de energia convencional no domínio da frequência.



Fonte: Baseado em (ATAPATTU; TELLAMBURA; JIANG, 2014).

Após calcular o valor da estatística de teste, é realizada a comparação com o limiar pré-determinado, para decidir a presença ou não do sinal. A seleção desse limiar é muito importante, pois ela influencia diretamente se o resultado final será confiável e eficiente. Para calcular essa eficiência são utilizadas algumas métricas, tais como: probabilidade de detecção, probabilidade de falso alarme, e probabilidade de não-detecção (RIBAS, 2015).

3 DESENVOLVIMENTO

Neste capítulo será apresentado o desenvolvimento do analisador de controle remoto utilizando um *dongle* RTL-SDR na plataforma GNU Radio. A apresentação desse capítulo foi dividida em sete partes, que são: uma introdução geral sobre o sistema e as considerações mais importantes sobre o desenvolvimento no GNU Radio; os quatro módulos implementados, que são: o módulo receptor, responsável por receber o sinal a partir da entrada USB e fazer o tratamento necessário para conversão em bits; o módulo decodificador, que identifica a sequência código transmitida e o codificador utilizado; o módulo detector de frequência, que encontra em qual frequência houve a transmissão do sinal; e o módulo detector de *clock*; a integração dos módulos do sistema; a interface gráfica do usuário implementada e por fim considerações do desenvolvimento.

Para o analisador de controle remoto foram desenvolvidos cinco novos blocos, que serão apresentados junto com os módulos em que estão inseridos. A implementação deles foi realizada utilizando a linguagem de programação Python, em conjunto com as funções do GNU Radio e da biblioteca NumPy, que fornecem diversas funções matemáticas e de processamento de sinais.

A abordagem escolhida para a criação dos blocos foi utilizar o *Embedded Python Block*, disponível no GRC, e que permite criar um novo bloco diretamente no fluxograma. Esse bloco de criação fornece uma estrutura inicial definindo as entradas e saída do bloco, porém é possível mudar completamente essas definições iniciais, como por exemplo: o tipo do bloco, os tipos e número de portas de entradas e saídas e o processamento que será realizado no bloco. Com essa abordagem o desenvolvimento fica mais simples e mais rápido, pois o bloco é inserido diretamente no fluxograma, e sempre que o código desenvolvido é modificado e salvo, o GRC atualiza automaticamente as portas e os parâmetros do bloco, tornando muito mais rápido integrar e testar as novas modificações com o resto do sistema.

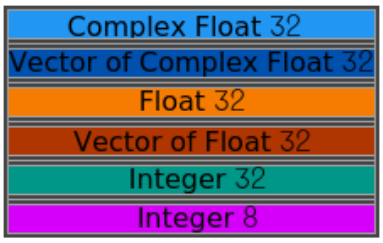
Nesse trabalho foram utilizados dois tipos de bloco disponíveis no GNU Radio, o síncrono e o básico. O tipo síncrono possui o mesmo número de amostras de entrada e de saída, ou seja, o bloco produz e consome o mesmo número de itens. O tipo básico não tem nenhuma regra em relação ao número de amostras de entrada e saída, porém todo o controle de consumo das amostras é realizado manualmente.

Ao longo do fluxograma os dados que fluem entre os blocos possuem diferentes tipos, esses tipos são apresentados com diferentes cores nas portas de entrada e saída do bloco. Uma ilustração com os tipos utilizados no sistema pode ser vista na Figura 13, nela pode ser observado cada tipo com a quantidade de bits que possui e sua respectiva cor no GRC.

3.1 Módulo receptor

O módulo receptor implementado foi baseado no modelo apresentado no capítulo anterior, ele foi desenvolvido utilizando blocos de processamento disponíveis no GNU Radio e novos blocos

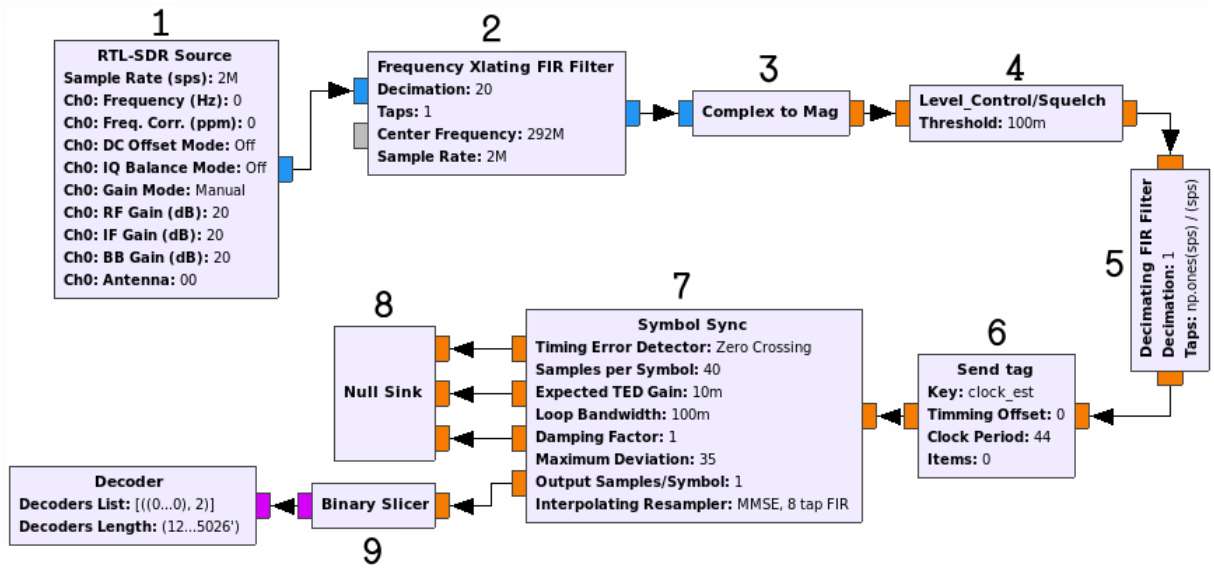
Figura 13 – Tipos de dados utilizados e suas representações por cores no GNU Radio.



Fonte: Baseada em (GNU RADIO, 2018).

implementados em Python. Esse modelo recebe e identifica sinais com modulação OOK e ao final fornece os bits de informação para o módulo decodificador. A Figura 14 apresenta o fluxograma completo do módulo receptor, cada um dos blocos que compõem o módulo serão explicados a seguir.

Figura 14 – Fluxograma do módulo receptor com os blocos que o compõem.



Fonte: Elaborada pelo autor.

3.1.1 RTL-SDR Source

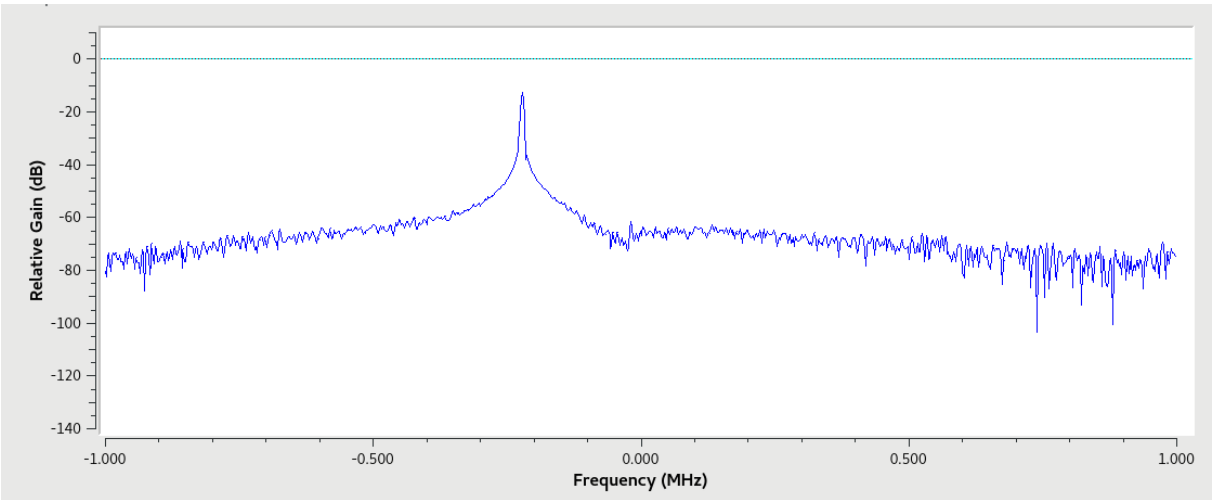
RTL-SDR Source é o primeiro bloco do fluxo. É esse bloco que recebe as amostras IQ enviadas pelo *dongle* RTL-SDR via USB. Ele é considerado um bloco fonte, pois não possui entradas, apenas uma saída com amostras do tipo *complex float*. Esse bloco é responsável por configurar características como ganho, frequência de sintonização, taxa de amostragem e outros diversos parâmetros. Os mais relevantes são apresentados a seguir:

- `Sample Rate`: Taxa de amostragem em amostras por segundo, esse parâmetro determina o número de amostras de saída do bloco e também a largura de banda O valor máximo para esse parâmetro é 3,2 MS/s, e o valor padrão utilizado no trabalho foi 2 MS/s.
- `Frequency`: É o parâmetro que é passado ao sintonizador do RTL-SDR para selecionar a frequência central desejada. A faixa sintonizada é a frequência central ± a metade

da largura de banda definida, ou seja ± 1 MHz. O valor desse parâmetro é selecionado automaticamente através do módulo detector de frequência que será apresentando na seção 3.3.

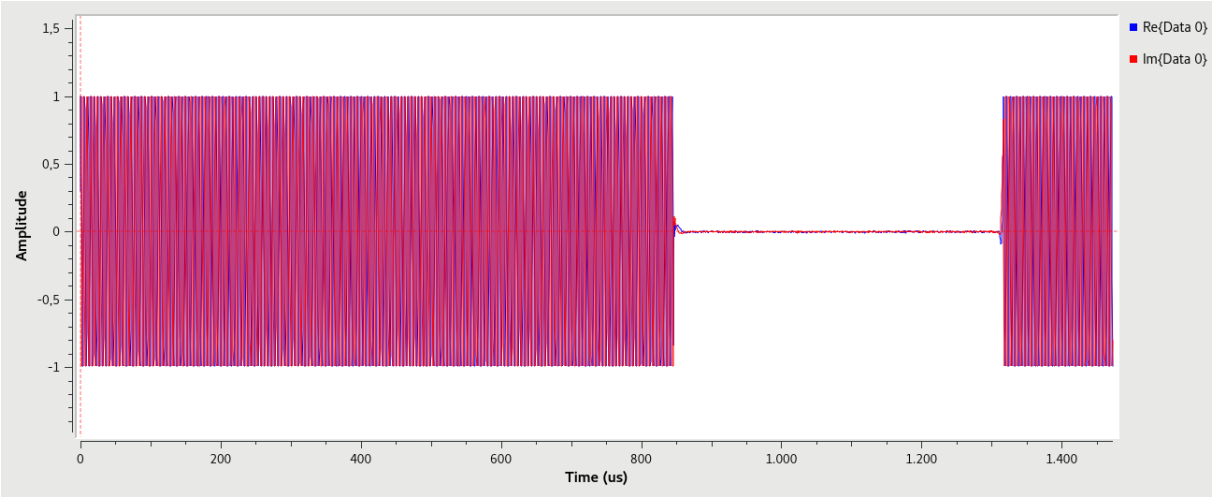
As Figuras 15 e 16 ilustram um sinal recebido pelo *RTL-SDR Source* no domínio da frequência e no domínio do tempo, respectivamente. É possível observar com a Figura 16 o sinal complexo das amostras IQ recebidas.

Figura 15 – Saída do bloco *RTL-SDR Source* no domínio da frequência.



Fonte: Elaborada pelo autor.

Figura 16 – Saída do bloco *RTL-SDR Source* no domínio do tempo.



Fonte: Elaborada pelo autor.

3.1.2 Subamostragem

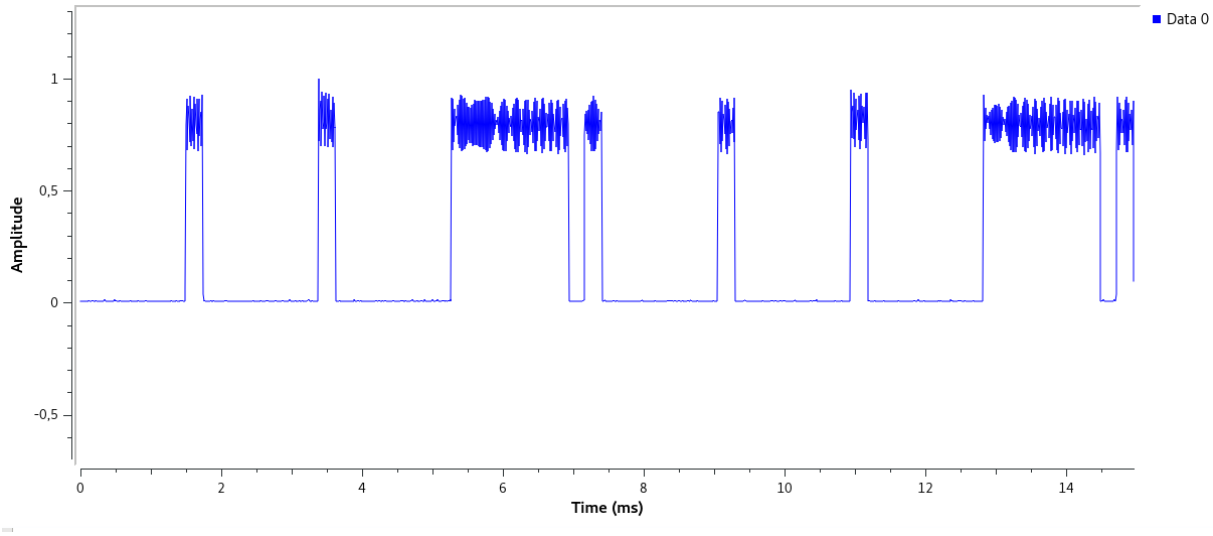
No segundo bloco da Figura 14 é realizado a subamostragem do sinal para esse fluxo. Para isso é utilizado o bloco *Frequency Xlating FIR Filter*. Esse bloco executa três passos: desloca a frequência do sinal de interesse para o centro, aplica um filtro *anti-aliasing*, e por fim realiza a subamostragem. Para isso foram utilizados alguns parâmetros do bloco, o `Decimation`, que

recebe um valor inteiro e seleciona a taxa de decimação, `Center Frequency` que recebe o valor da frequência detectada, retornada pelo módulo detector de frequências, e o `Sample Rate` que é o valor original de taxa de amostragem de 2 MS/s. O outro parâmetro configurável é o `Taps`, é nele que são inseridos os coeficientes do filtro desejado. A entrada e saída desse bloco é do tipo *complex float*. O valor de decimação utilizado para sub-amostrar o sinal foi de 20, resultando em um frequência de amostragem de 100 kS/s a partir desse bloco.

3.1.3 Cálculo da magnitude

O sinal sub-amostrado recebido é então passado pelo terceiro bloco da Figura 14, o *Complex to Mag*. É ele quem calcula a magnitude das amostras complexas de entrada, resultando em uma saída do tipo *float*. Esse processo é realizado pois toda a informação do sinal está na sua magnitude, nele é extraído o formato do sinal original a partir do sinal de onda da portadora, ou seja do sinal OOK. Esse processo pode ser observado comparando a Figura 16 com a Figura 17. A última apresenta a saída do bloco *Complex to Mag* no domínio do tempo, nela podemos observar que o sinal não está mais no formato complexo, e é possível observar a informação de interesse sendo representada pelo nível da amplitude.

Figura 17 – Sinal OOK demodulado pelo bloco *Complex to Mag*.



Fonte: Elaborada pelo autor.

3.1.4 Bloco *Level control/Squelch*

O quarto bloco do fluxo é o bloco *Level control/Squelch*, apresentado na Figura 18. Ele foi desenvolvido utilizando o *Embedded Python Block*, como mencionado anteriormente. O seu tipo foi definido como básico, ou seja, o número de amostras de entrada e saída não precisam ser os mesmos, a entrada e a saída foram definidas como tipo *float*. O bloco possui apenas um parâmetro, apresentado a seguir:

- `Threshold`: Limiar de decisão utilizado para suprimir a saída. O valor limiar do sistema foi definido após observações do sistema com ausência e presença de transmissão.

Figura 18 – Bloco *Level control/Squelch*.

Fonte: Elaborada pelo autor.

O objetivo desse bloco é fazer o controle do nível das amostras, incluindo nele uma função *noise squelch* para suprimir a saída na ausência do sinal dos controles remotos. A implementação da lógica do bloco é simples e funciona seguindo alguns passos. O primeiro passo é encontrar dentro de uma sequência variável de amostras de entrada aquela com maior amplitude. O segundo passo é verificar se a amostra de maior valor é inferior ao limiar determinado, caso seja menor a saída do bloco será suprimida e o processo é terminado, caso seja maior que o limiar, toda a sequência de amostras será dividida pelo valor da amostra máxima encontrada, resultando em amostras com uma variação de amplitude entre 0 e 1. Após todo esse processo, todas as amostras são subtraídas com um valor constante de 0,5, resultando em amostras *non-return-to-zero* (NRZ) que variam entre $-0{,}5$ e $+0{,}5$. Esse passo é necessário para o funcionamento dos blocos *Zero Crossing* e o *Symbol Sync*, que serão descritos mais adiante.

3.1.5 Filtro casado

No quinto bloco é implementado o filtro casado. Para isso é utilizado o bloco *Decimating FIR Filter*, que possui dois parâmetros, o `Decimation` que seleciona a taxa de decimação, e o `Taps` que é o único parâmetro de interesse para esse processamento, nele são inseridos os coeficientes do filtro a ser implementado, sendo eles do tipo *float*. O número de coeficientes utilizados é a quantidade de amostras por símbolo do controle analisado. Isso é necessário para que a saída do filtro tenha o seu valor de pico no período de duração de um símbolo completo T, resultando assim na entrada necessária ao bloco sincronizador. O valor da quantidade de amostras por símbolo é obtido através do módulo detector de *clock*. que será apresentado mais adiante.

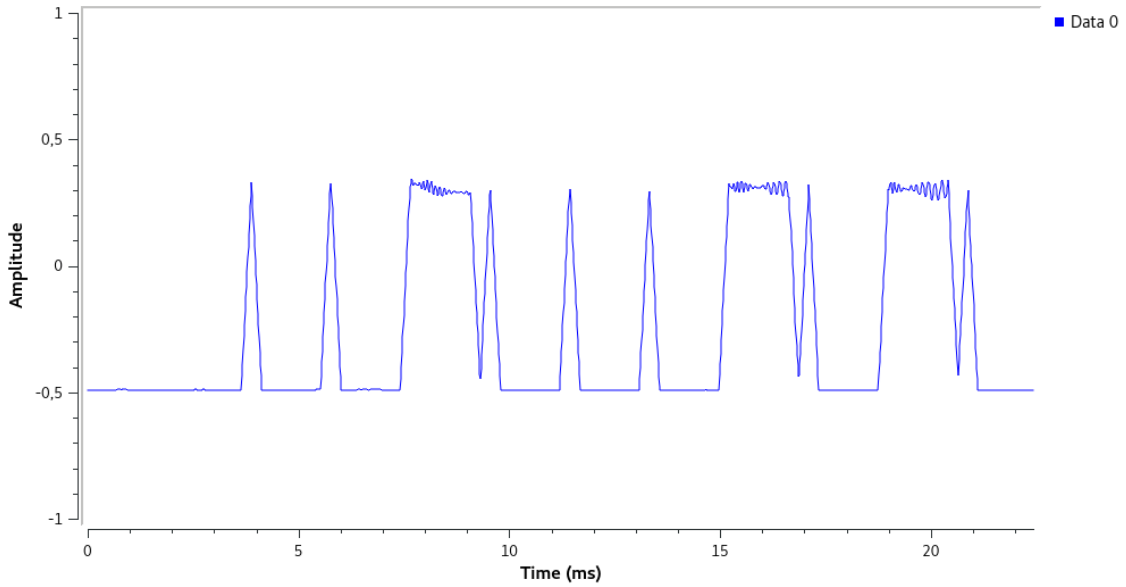
A Figura 19 apresenta o sinal após o processamento pelo filtro casado, nela é possível observar o formato NRZ, realizado pelo bloco *Level control/Squelch*. É possível comparar as Figuras 18 e 19 e observar os picos gerados pelo filtro casado para a mesma informação transmitida, sendo nesse caso para um sinal do codificador HT6026.

3.1.6 Sincronizador

O bloco *Symbol Sync*, representado pelo sétimo bloco na Figura 14, é utilizado para encontrar os centros de uns e zeros, para isso é preciso que o sinal de entrada possua picos, sendo necessário o processamento anterior realizado pelo filtro casado. Esse bloco possui quatro saídas, a única de interesse para o sistema é a `Out`, que é a saída de fluxo de amostras sincronizado com o centro de símbolo dos dados. O oitavo bloco do fluxograma é um *Null Sink* que recebe as saídas não utilizadas e as descarta. Os parâmetros de configuração mais importantes são:

- `Timing Error Detector (TED)`: É o algoritmo de TED a ser utilizado, nesse caso foi

Figura 19 – Resposta do sinal ao filtro casado no domínio do tempo.



Fonte: Elaborada pelo autor.

usado o *Zero Crossing*. O TED emite um valor de erro proporcional a diferença de tempo entre o tempo de amostragem do símbolo estimado e o tempo de amostragem do símbolo real.

- `Samples per Symbol`: Estimativa inicial do período de *clock* especificado pelo usuário em amostras por símbolo. O valor estimado utilizado no trabalho foi de 40 amostras por símbolo, esse valor foi definido de acordo com as características dos controles utilizados nos testes.

- `Maximum Deviation`: É o máximo desvio do *clock* em unidades de amostras por símbolo que o sincronizador pode assumir. É importante frisar que o intervalo de *clock* permitido pelo *Symbol Sync* é o valor nominal de amostras por símbolo mais ou menos o máximo desvio especificado. Dessa maneira, o valor do `Maximum Deviation` deve ser grande o suficiente para ser possível alterar o *clock* do bloco com o valor desejado. O valor utilizado foi de 35, permitindo sincronizar com controles que possuam sinais entre 5 a 75 amostras por símbolo.

- `Output Samples/Symbol`: Número de amostras por símbolo de saída após o sincronismo, o valor utilizado é 1 amostra por símbolo.

O bloco *Symbol Sync* permite receber fluxo de dados com a presença de *tags*, essas *tags* no fluxo de dados são uma maneira de passar dados de controle entre blocos através de *polymorphic types* (PMT)[1], além de apenas amostras ou bits. O bloco *Symbol Sync* define duas *tags*, a `clock_est` e a `time_est`, a de interesse para o trabalho é a `clock_est` que permite resetar o período de *clock* do bloco. Uma *tag* é definida como:

[1] *polymorphic types* são tipos de dados desenvolvidos como *containers* genéricos, podendo armazenar qualquer tipo, porém possuem fácil conversão para os tipos mais comuns de dados.

- `Offset`: Número associado a *tag*, sendo a posição absoluta do item no fluxo de dados.
- `Key`: Símbolo PMT que identifica uma tag.
- `Value`: Informação PMT a ser transmitida com o fluxo.

O bloco *Send Tag*, representado pelo sexto bloco no fluxograma da Figura 14, foi implementado em Python, utilizando o *Embedded Python Block* definido como tipo síncrono. O objetivo do bloco é enviar uma *tag* para o bloco *Symbol Sync* e alterar o período de *clock* do sincronizador, com base no valor estimado pelo módulo detector de *clock*, esse módulo será apresentado na seção 3.7. O bloco possui um lógica para controlar quando uma *tag* será anexa ao fluxo, sendo necessário um intervalo de dez mil amostras para anexar uma nova *tag* para resetar o *clock*. Essa operação é necessária pois cada controle remoto tem sua frequência de *clock* própria, e o sincronizador precisa de uma estimativa para o funcionamento correto. A Figura 20 ilustra o bloco *Send tag* no GRC, e os parâmetros utilizados são apresentados a seguir:

- `Key`: Chave que identifica a *tag*, nesse caso `clock_est`.
- `Timming Offset`: Desvio da amostra estimado, variando no intervalo $[-1, 1]$.
- `Clock Period`: Período do *clock* estimado pelo módulo detector de *clock*, valor em amostras por símbolo.
- `Items`: Número de amostras que determina o intervalo de inserção da *tag*.

Figura 20 – Bloco *Send Tag*.

Fonte: Elaborada pelo autor.

3.1.7 Decisor

Por fim, após o bloco de sincronismo enviar uma amostra por símbolo é realizada a decisão final pelo nono bloco e último bloco, o *Binary Slicer*. Esse bloco recebe uma entrada em *float* e produz um bit de saída, esse valor poderá ser bit "0" quando a entrada é negativa ou "1" quando positiva. Agora o sinal já está pronto para ser enviado ao módulo decodificador que será apresentando na seção 3.2.

3.2 Módulo decodificador

O módulo decodificador, apresentado na Figura 21, possui apenas um bloco de processamento, o *Decoder*. O objetivo dele é identificar a sequência código transmitida e o codificador

utilizado. Ele recebe o fluxo de dados em bits após passar por todos os blocos de recepção do sinal.

Figura 21 – Bloco *Decoder*.

Fonte: Elaborada pelo autor.

O bloco *Decoder* foi implementado com o objetivo de ser genérico, permitindo receber novos modelos de codificadores. Para isso ele recebe dois parâmetros, o *Decoders List* e o *Decoders Info*.

O parâmetro *Decoders List* é uma lista de lista de tuplas com informações dos decodificadores, ou seja uma lista aninhada que contém tuplas, em que cada elemento da lista aninhada é referente às configurações de um codificador específico. As tuplas definidas trazem informações contendo o valor binário/ternário original da informação e o código que representa essa informação. Um exemplo da informação armazenada em uma lista aninhada está apresentada na Tabela 2, no qual o valor 'P' de decodificação foi escolhido para representar a sequência piloto e o valor 'E' foi escolhido para representar uma sequência que não está presente na lista aninhada, ou seja um erro de decodificação para aquele codificador. As informações de codificação apresentadas na Tabela 2 são referentes ao codificador HT6P20B, e cada linha da tabela representa uma tupla.

Tabela 2 – Informação do codificador HT6P20B.

	Valor decodificado	Código
Zero	'0'	001
Um	'1'	011
Piloto	'P'	000000001
Erro	'E'	outros

O `Decoders Info` é uma lista de tuplas com a informação do tamanho que representa um quadro completo e o nome do codificador associado. A Tabela 3 apresenta um exemplo da informação armazenada. É importante mencionar que as duas listas são relacionadas através do índice, em que cada elemento da lista aninhada é mapeado com um elemento da lista `Decoders Info`, possuindo assim todas as informações necessárias de um codificador.

Tabela 3 – Informação do parâmetro `Decoders Info`.

Codificador	Tamanho
HT6026	9
HT12E	12
HT6P20B	28

A lógica implementada executa a decodificação da sequência de dados recebida com as diferentes listas aninhadas e suas respectivas tuplas, ou seja, realiza a operação com os diferentes decodificadores até ser identificado um quadro de transmissão completo. Para isso são realizados

dois processos. O primeiro processo é onde ocorre a efetivamente a decodificação, a sequência de bits que chega no bloco *Decoder* é varrida do início ao fim sendo comparada com os códigos definidos, gerando uma sequência decodificada contendo símbolos de "valor decodificado", como apresentado na Tabela 2.

Com a sequência decodificada pronta, é realizado o segundo processo, que consiste em verificar se um quadro completo foi recebido, a Figura 22 apresenta esse processo. Como pode ser observado na figura, um quadro será considerado completo quando for encontrado um piloto inicial, seguido de uma sequência decodificada que não possua erros, do mesmo tamanho de um quadro completo definido para o decodificador selecionado, e com um piloto ao final. Na Figura 22, a sequência decodificada é representada pela variável `seq`, `Tamanho_quad` é o tamanho para ser considerado um quadro completo, os valores para a variável `estagio` são: `estagio` = 0, estado inicial, sistema espera um quadro piloto inicial; `estagio` = 1, sistema já encontrou o piloto inicial e espera sequências códigos; e `estagio` = 2, significa que foi encontrada uma sequência código com o tamanho esperado e o sistema aguarda que o próximo símbolo seja um piloto. Quando um quadro completo for encontrado, a sequência desse quadro e o codificador serão armazenados para serem usadas pelas funções `retorna_seq_detectada` e `retorna_cod_detectado`, essas funções são utilizadas para apresentar a sequência encontrada e o codificador detectado na interface gráfica do usuário.

3.3 Módulo detector de frequência

O módulo detector de frequência é responsável por fazer o sensoriamento espectral, ele possui como objetivos determinar a presença ou ausência de um sinal em uma faixa pré-determinada utilizando a técnica de detecção de energia no domínio da frequência, e também identificar a frequência com maior pico de amplitude na faixa selecionada, encontrando assim a frequência utilizada pelo controle remoto.

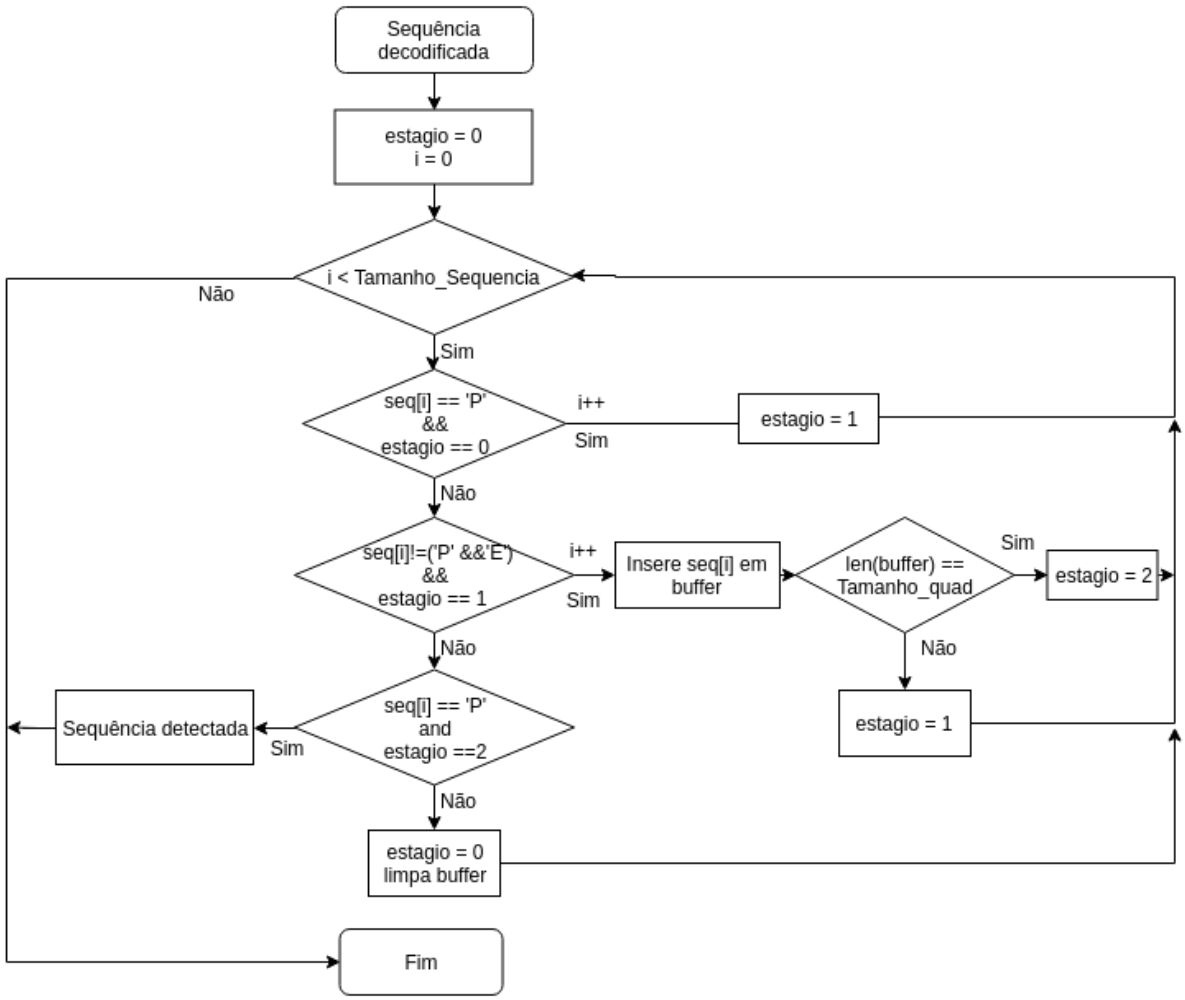
Levando em consideração que os valores de frequência mais utilizados pelos controles remotos variam entre 292 MHz e 433,92 MHz, e visando evitar o sensoriamento em faixas de frequências desnecessárias, foi determinado uma lista com as frequências mais utilizadas pelos controles remotos. A Tabela 4 apresenta a lista de frequências centrais que o sintonizador poderá receber como parâmetro, e a faixa sensoriamento referente a frequência selecionada, são nessas faixas que o detector de energia realizará o sensoriamento. Ademais, o controle sobre qual frequência será selecionada no bloco *RTL-SDR Source* é feito por esse módulo, o que permite o sistema ficar centralizado na frequência em que exista a presença de um sinal ou pular para próxima faixa caso não exista.

Tabela 4 – Lista das frequências centrais.

Frequência central	Faixa de sensoriamento	
292 MHz	291 MHz	293 MHz
299 MHz	298 MHz	300 MHz
315 MHz	314 MHz	316 MHz
434 MHz	433 MHz	435 MHz

A Figura 23 apresenta o fluxograma completo do módulo detector de frequência na plata-

Figura 22 – Fluxograma de identificação de um quadro completo.



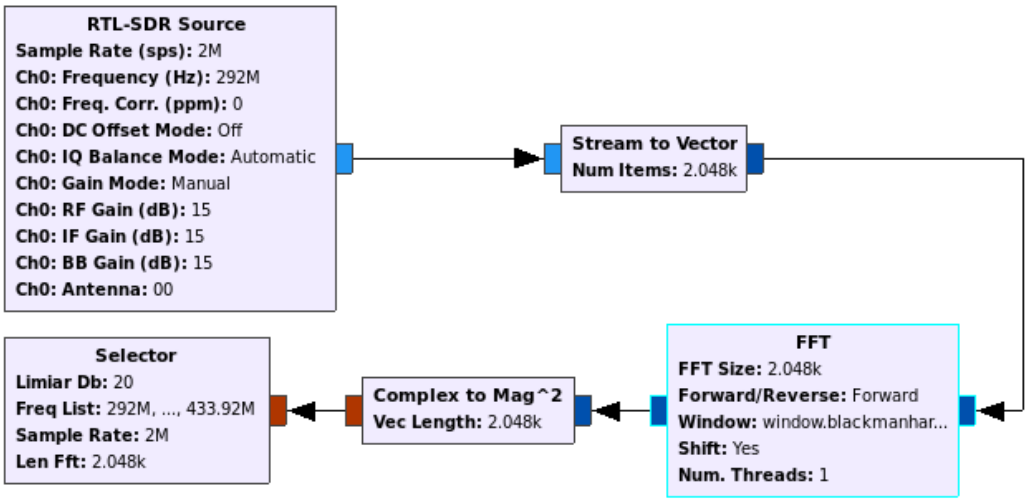
Fonte: Elaborada pelo autor.

forma GNU Radio. A sua implementação foi baseada no modelo apresentado na subseção 2.4.2, e foi realizada utilizando quatro blocos disponíveis pelo GRC e um bloco desenvolvido em Python, o *Selector*, que controla e fornece as informações referente a frequência de operação do sistema.

O primeiro bloco apresentado na Figura 23 é o *RTL-SDR Source*, pois é nele que é atualizado a frequência central a ser sintonizada, é importante ressaltar que o fluxograma desse módulo recebe as amostras com a taxa de amostragem original de 2 MS/s. O segundo bloco em sequência é o *Stream to Vector*, que transforma o fluxo em um vetor, recebendo 2048 amostras, que são 2048 itens e retornando um vetor com 2048 amostras. O bloco *FFT* computa a transformada rápida de Fourier com um tamanho de 2048. A saída da FFT complexa é convertida para sua magnitude quadrática, retornando um vetor do tipo *float*. Por fim o sinal é processado pelo bloco *Selector*.

No bloco *Selector* é recebida uma entrada de vetores com 2048 itens, com eles é calculado a somatória, gerando como resultado a estatística de teste Λ, que será comparada com um valor limiar determinado. Em conjunto com o cálculo da estatística de teste é encontrado o índice do vetor de pico recebido, e com base na sua posição é determinado o desvio da frequência em

Figura 23 – Fluxograma do módulo detector de frequência no GRC.



Fonte: Elaborada pelo autor.

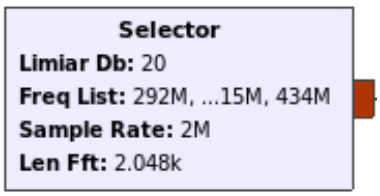
comparação com a frequência central, de acordo com

$$f_{\text{desvio}} = \frac{nF_s}{N} - \frac{F_s}{2}, \tag{3.1}$$

em que F_s é a taxa de amostragem, n é o índice no vetor e N é o tamanho da FFT. Como o tamanho da FFT é de $N = 2048$ itens e a taxa de amostragem é de $F_s = 2\,\text{MS/s}$, a precisão do desvio de frequência será de 976,56 Hz. O bloco *Selector* é apresentado na Figura 24, os seus parâmetros recebidos são apresentados a seguir:

- `Limiar Db`: Valor limite (em dB) utilizado para ser comparado com a estatística de teste Λ. O valor determinado foi de 20 dB, esse valor foi escolhido empiricamente.
- `Freq List`: Lista das frequências centrais as quais serão sensoriadas. Elas podem ser vistas na Tabela 4.
- `Sample Rate`: Taxa de amostragem, nesse caso é 2 MS/s.
- `Len Fft`: Tamanho utilizado para o cálculo da FFT. O valor utilizado foi de 2048.

Figura 24 – Bloco *Selector* no GRC.



Fonte: Elaborada pelo autor.

Os valores de Λ e f_{desvio} calculados são armazenados para serem utilizados por duas funções independentes do fluxograma definidas no bloco *Selector*, a `freq_ativa` que verifica a

presença ou ausência do sinal e a `freq_detectada` que identifica a frequência de operação. É a partir dessas funções que é realizada a integração do módulo detector de frequência com os outros módulos.

A função `freq_ativa` é utilizada para retornar a frequência que será sintonizada no bloco *RTL-SDR Source*. Para isso são utilizadas as informações da estatística de teste calculada, a lista de frequências centrais e o valor limiar, ambos fornecidos como parâmetros. Após a função ser chamada, o valor Λ atual é comparada com o limiar, caso o valor Λ seja inferior ao limiar é identificado a ausência de sinal na faixa, então a função seleciona a próxima frequência da lista e retorna essa frequência selecionada, caso seja superior ao limiar, é encontrado a presença de um sinal e a função retorna a frequência atual já selecionada, o que faz o sistema ficar sintonizado quando existe a presença do sinal.

A função `freq_detectada` é usada para apresentar a frequência identificada do controle remoto na interface do usuário e para centralizar o sinal no processo de subamostragem, como já abordado na subseção 3.1.2. Essa função utiliza as informação de f_{desvio}, estatística de teste, frequência atual selecionada e limiar. Quando essa função é chamada, primeiramente é verificado se existe a presença do sinal na faixa da mesma maneira que na função anterior, caso não exista a função retorna '0', isso significa que não foi identificado uma frequência de operação e não será apresentado um valor na interface do usuário, caso exista a presença do sinal, é calculado a frequência de operação somando a f_{desvio} calculado e a frequência atual sintonizada, após isso o valor encontrado é retornado.

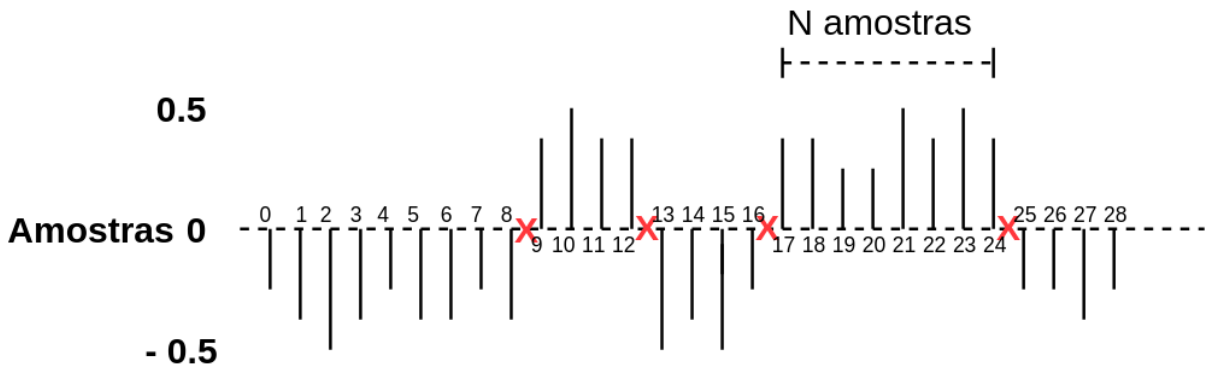
3.4 Módulo detector de *clock*

Para identificar a frequência de *clock* do controle remoto foi implementado um bloco que é baseado na técnica denominada *zero crossing*, esta técnica é utilizada para estimar o período do sinal transmitido (CHEN; CHIEN, 2015). A ideia original do método é levar em consideração que, entre dois cruzamentos por zero é possível determinar a metade do período do sinal, desta maneira, é calculado o intervalo entre esses dois pontos que cruzam o zero, e então é convertido o valor estimado para a frequência de *clock* (MAžEIKA; DRAUDVILIENė, 2010).

O bloco desenvolvido funciona seguindo alguns passos. Primeiramente é recebido uma sequência de amostras do bloco *Level control/Squelch*, no formato NRZ, com valores de amplitudes entre $[-0.5, 0.5]$, ou seja, valores que variam entre a passagem por zero. Depois disso, é determinado os índices das amostras em que ocorreram um cruzamento. Com a Figura 25 é possível observar melhor esse processo, no qual os pontos marcados com "X" são os pontos onde ocorrem uma passagem por zero. O resultado dos índices para as amostras da figura seriam $[8, 12, 16, 24]$. Com os índices determinados, é então calculada a diferença entre os índices adjacentes. Por fim é identificado o menor valor entre as diferenças, para o caso da figura o valor seria 4 amostras.

O valor encontrado no processo descrito anteriormente é o número de amostras por símbolo, levando em consideração que para a modulação OOK, entre dois cruzamentos por zero é equivalente ao período do símbolo inteiro. Esse valor encontrado é armazenado para ser utilizado por duas funções definidas nesse bloco, a `retorna_sps` e a `retorna_freq`, essas funções são

Figura 25 – Identificando índices das amostras que fazem cruzamento por zero.



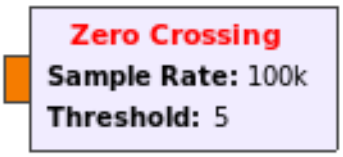
Fonte: Elaborada pelo autor.

responsáveis por fazer a integração desse módulo com os outros, elas serão descritas mais adiante. O bloco implementado é ilustrado na Figura 26 e seus parâmetros são apresentados a seguir:

- `Sample Rate`: Taxa de amostragem, sendo nesse caso 100 kHz, pois o bloco *Zero Crossing* faz parte do fluxo que recebe o sinal sub-amostrado no módulo receptor.

- `Threshold:` Limiar de menor diferença entre os índices. Caso o bloco encontre dois cruzamentos por zero com intervalo menor que cinco, o sistema ignora e mantém o valor anterior. Isso é feito para evitar detecção errônea enquanto não existir um sinal transmitido.

Figura 26 – Bloco *Zero Crossing*.



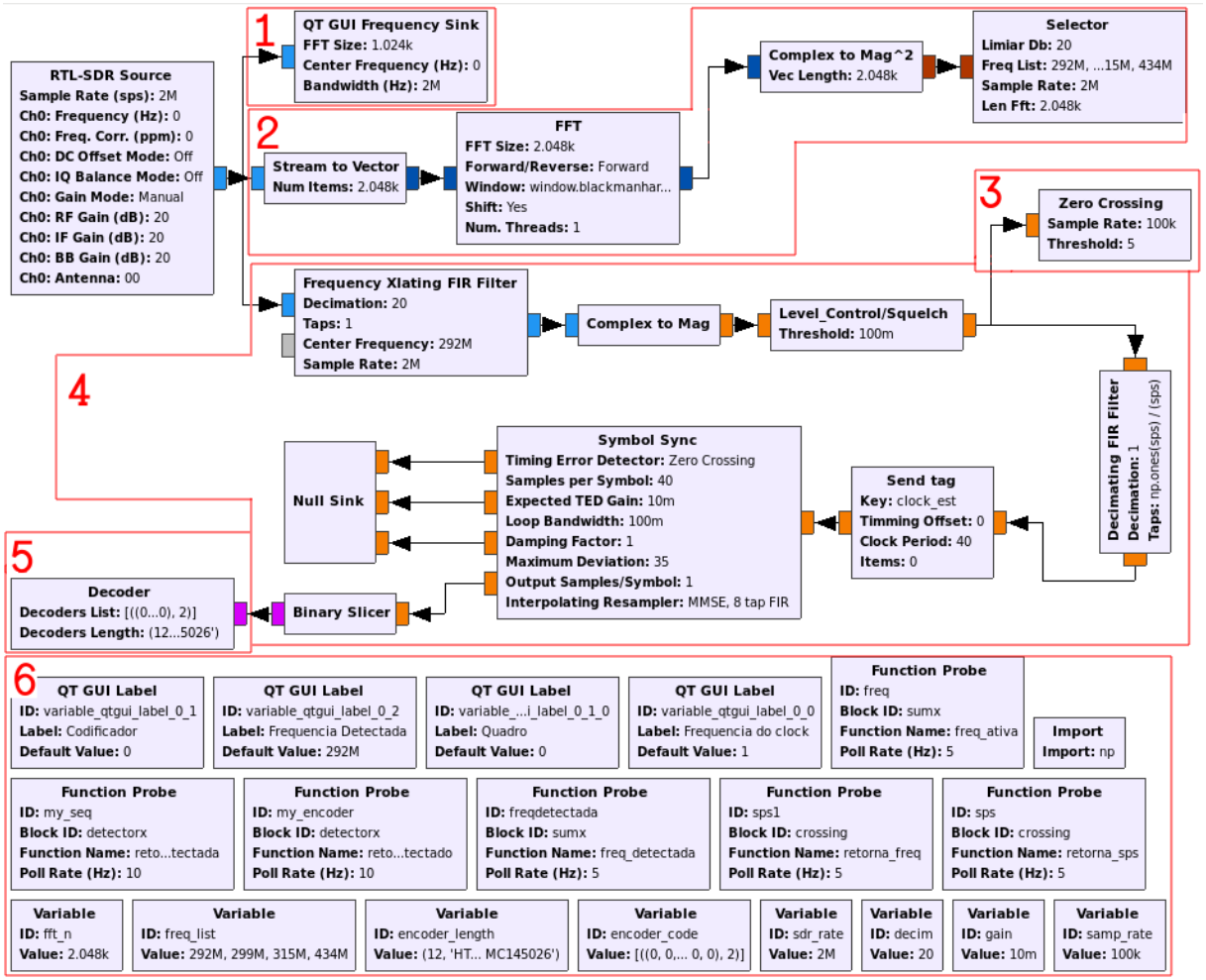
Fonte: Elaborada pelo autor.

A função `retorna_sps` é responsável por fornecer o valor do número de amostras por símbolo para o filtro casado e o bloco *send_tag*, essa função só retorna o valor calculado se ele for maior que o limiar recebido como parâmetro, caso seja menor ele retorna o valor de cinco amostras por símbolo. A função `retorna_freq` é responsável por retornar a frequência de *clock* que será apresentada na interface de usuário, ela pode ser calculada utilizando o valor da taxa de amostragem e o número de amostras por símbolos encontrado, o cálculo é realizado como: $f_{\text{clk}} = \frac{F_s}{\text{sps}}$, esse valor calculado só é retornado se o número de amostras por símbolo encontrado for maior que o limiar, caso seja menor é retornado zero. Com o valor da frequência de *clock* é possível calcular também a taxa de transmissão antes da codificação como $R_{\text{b}} = \frac{f_{\text{clk}}}{N_{\text{bits}}}$, sendo N_{bits} o número de bits por codificação.

3.5 Integração dos módulos

A Figura 27 apresenta o sistema completo no GRC com todos os módulos integrados e os blocos necessários para o funcionamento do analisador de controle remoto. A integração do sistema se da através de dois modos: um diretamente ligado pelos fluxogramas, que são os módulos já apresentados anteriormente, essa integração pode ser observada na parte superior da Figura 27, nela os quadros enumerados de 2 a 5 são respectivamente os módulos detector de frequência de operação, detector de *clock*, receptor e decodificador, lembrando que todo o processamento de sinais no GNU Radio é realizado apenas através desses fluxogramas, em que cada bloco realiza um processamento. O quadro número 1 representa o bloco que implementa a interface gráfica do usuário, ela será descrita mais adiante. O outro modo de integrar o sistema é através de troca de informações fora do fluxograma, nela o sistema utiliza blocos que não executam processamento diretamente no fluxo, são blocos de auxilio que fazem a integração entre os módulos, como por exemplo, os blocos *Function Probe* e *variable*, apresentados na parte inferior da Figura 27, contidos no quadro número 6.

Figura 27 – Fluxograma do sistema completo.



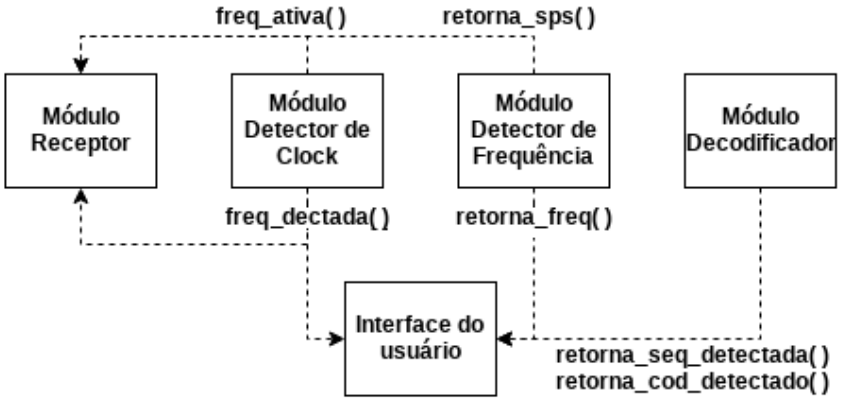
Fonte: Elaborado pelo autor.

Esses blocos de auxilio integram os módulos a partir das funções apresentadas anteriormente, para isso é utilizado o bloco *Function Probe* que periodicamente chama essas funções e

define o valor de retorno em uma variável visível para todo o sistema, permitindo a passagem de dados de controle entre os módulos, essa variável é representada pelo bloco *variable*. A Figura 28 apresenta o diagrama de integração através dessas funções, nela é possível observar as funções definidas de cada módulo e com qual módulo ela interage. O *Function Probe* é controlado através de quatro parâmetros principais, que são:

- `ID`: O Nome da variável que recebe o retorno da função.
- `Block ID`: O Nome do bloco onde a função está declarada.
- `Function Name`: O nome da função a ser chamada.
- `Poll Rate`: Parâmetro que define a frequência em que a função será chamada pelo sistema.

Figura 28 – Diagrama de integração fora do fluxograma.



Fonte: Elaborado pelo autor.

Para ser possível o correto funcionamento do sistema uma sequência de eventos entre os módulos devem ocorrer, esses eventos são transmitidos através da funções mencionadas anteriormente. Um cenário de uso é descrito a seguir, com ela é possível observar melhor esses eventos.

Com um usuário transmitindo um sinal de controle remoto, o sistema primeiramente deve encontrar em qual faixa de frequência a transmissão está ocorrendo, cada faixa é varrida por 0.2 segundos, após encontrada a faixa ativa, o sintonizador ficará fixo nela enquanto o sinal estiver sendo transmitido. Quando o sintonizador estiver na faixa correta o módulo detector de *clock* calculará a quantidade de amostras por símbolo do sinal, e então o módulo receptor será atualizado com o novo valor de amostras por símbolo a cada 0.2 segundos. Após o módulo receptor estar configurado para o controle remoto, o decodificador começa a receber as a sequência de amostras sincronizadas corretamente, e então será possível decodificar o sinal.

É importante frisar que, o processamento do módulo detector de frequência de *clock* e decodificador ocorrem em paralelo, são concorrentes, por isso é necessário o uso do controle remoto por um certo período para que todos os módulos estejam configurados para receber o sinal do controle remoto em análise. O tempo necessário para conseguir obter as informações foi avaliado e apresentado no Capítulo 4.

3.6 Interface gráfica do usuário

A Figura 29 apresenta a interface gráfica do sistema em ociosidade. Essa é a tela inicial do sistema, ela apresenta um gráfico no domínio da frequência e as informações de detecção. Essa tela é implementada através do pacote *QT Graphical User Interface* disponível no *GNU Radio*, esse pacote fornece diversos blocos para implementação de gráficos, o bloco utilizado nessa tela é o *QT GUI Frequency Sink*, é ele que apresenta o gráfico no domínio da frequência. As informações sobre o controle remoto são inseridas através de *QT GUI Labels* que recebem as informações dos módulos pelas funções apresentadas na Figura 28. Para o sistema entrar em operação basta o usuário pressionar um dos botões do controle remoto e aguardar as informações serem apresentadas na tela.

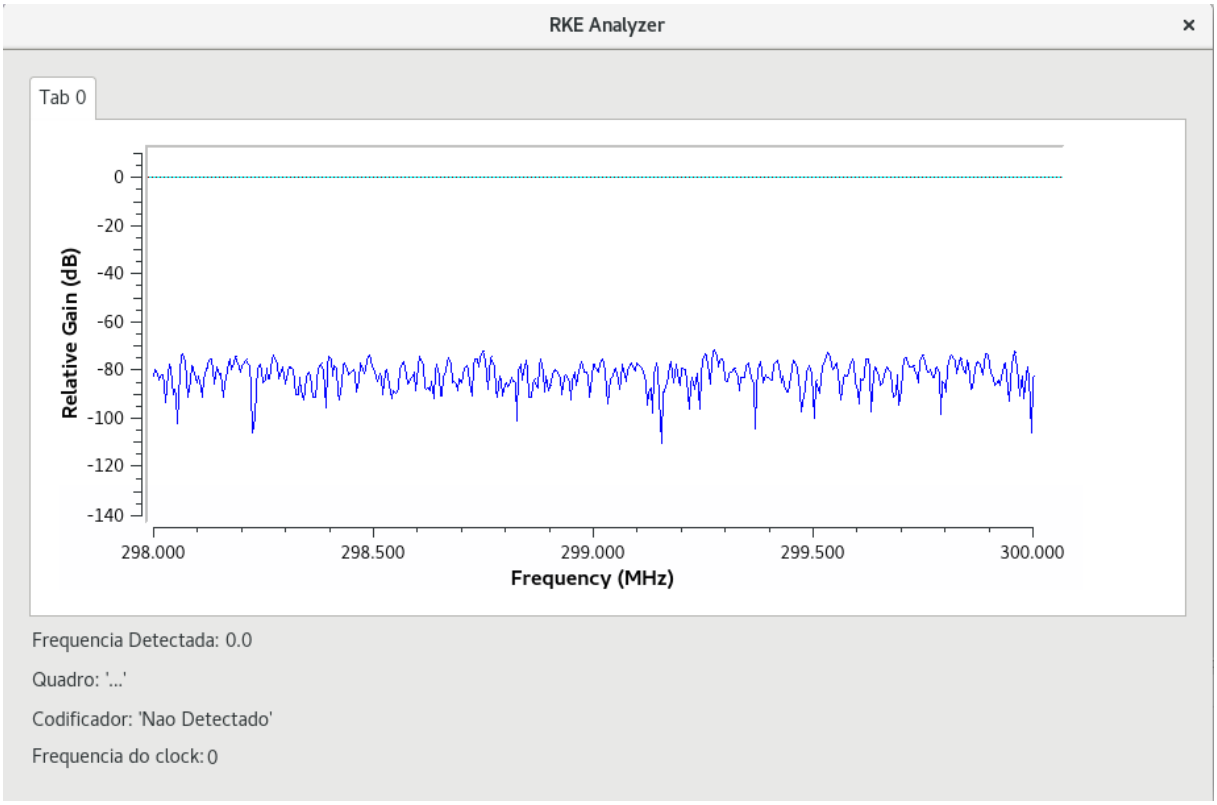


Figura 29 – Interface gráfica do analisador de controle remoto – Sem transmissão.
Fonte: Elaborada pelo autor.

3.7 Considerações do desenvolvimento

Algumas características do sistema são importantes de serem mencionadas. O *GNU Radio* facilmente consome todos os recursos da CPU, por conta disso, foram realizados diversos testes e abordagens para reduzir o seu consumo. Uma abordagem utilizada foi reduzir o número de amostras no sistema, sendo implementado o uso de um *Squelch* e de um sub-amostrador no módulo receptor, o que possibilitou o aumento taxa de amostragem do sistema, e consequentemente na largura de banda, conseguindo assim um maior sensoriamento para o módulo detector de frequência.

O módulo decodificador que além de identificar os sinais dos codificadores mencionados consegue também identificar a dos codificadores M1E-N e MC145026, pois eles possuem o mesmo protocolo do HT6026 e HT12E respectivamente.

O sistema ficou restrito às quatro faixas de frequência mais utilizadas pelos controles remotos. Em razão disso, caso um controle possua um desvio acima de $\pm$ 1 MHz de uma das frequências centrais definidas, não será possível encontrar as informações sobre ele.

O sistema detecta a sequência de transmissão de controles remotos que possuam a frequência de *clock* entre 1,33 kHz e 20 kHz. Essa limitação é devido as configurações relacionadas ao sincronizador, esses valores são definidos pelo parâmetro `Maximum Deviation` do *Symbol Sync*, que permite ao bloco resetar o *clock* com valores de 5 a 75 amostras por símbolo. No entanto, é possível aumentar essa faixa, porém o desempenho do sincronizador é reduzido quando existe um desvio máximo muito grande.

4 RESULTADOS E TESTES

Com o objetivo de apresentar o funcionamento do sistema desenvolvido e validá-lo serão apresentados nesse capítulo os testes e as análises dos resultados. Serão avaliadas as funcionalidades disponíveis pelo sistema, que são:

- Receber sinais com a modulação OOK.
- Identificar sinais codificados pelos padrões definidos pelos codificadores HT6026, HT6P20B e HT12E.
- Encontrar a frequência de operação que esteja na faixa de frequências apresentadas na Tabela 4.
- Calcular a frequência de *clock* do controle remoto emissor.
- Apresentar as informações em uma interface gráfica para o usuário.

4.1 Testes com transmissores

Os testes realizados para a validação do sistema foram divididos em três modalidades: testes com diferentes controles em variadas frequências, testes variando a distância e verificando o funcionamento do sistema e teste com relação ao tempo de resposta, ou seja, quanto tempo o sistema demora para identificar todas as informações do controle remoto.

4.1.1 Teste de validação

Para esses testes foram utilizados cinco controles remotos de diferentes marcas, dentre esses controles, três deles utilizam o codificador HT6P20B. O teste com diferentes controles com o mesmo padrão é muito interessante para validar o funcionamento do analisador para um dado codificador, os outros dois controles utilizam um codificador HT6026 e um MC145026, como já mencionado anteriormente o MC145026 possui o mesmo protocolo do HT12E.

Levando em consideração as frequências dos controles remotos, foi elaborada a Tabela 5, com ela podemos ver os valores apresentados pelos fabricantes desses dispositivos, lembrando que esses valores sofrem desvios ao longo do tempo de uso, e o conhecimento sobre os valores exatos é importante para a calibração dos controles remotos ajustáveis, resultando em melhores desempenho para o sistema.

As Figuras 30, 31 e 32 apresentam a interface gráfica para o usuário em funcionamento. O gráfico inserido na interface é a saída no domínio da frequência do bloco *RTL-SDR Source* já mencionado. Como podemos observar nas figuras, a interface apresenta as informações de interesse do controle remoto: a frequência detectada em MHz, a sequência código transmitida pelo controle remoto, o codificador utilizado e a frequência do *clock* em Hz.

Tabela 5 – Informações sobre os controles utilizados nos testes.

Controle	Codificador	Frequência	Marca
1	HT6026	292 MHz	E Sinal
2	MC145026	299,40 MHz	E Sinal
3	HT6P20B	433,96 MHz	Sulton
4	HT6P20B	433,96 MHz	FKS
5	HT6P20B	433,96 MHz	Intelbras

Ainda analisando essas figuras é possível observar as características de cada codificador, como o tamanho da sequência e sua informação, destacando o caso da Figura 30, em que o valor 'Z' é a informação de 'aberto'. Outra observação interessante de se fazer é a recepção da sequência para o codificador HT6P20B, como mencionado na subseção 2.1.1, a sequência enviada por esse codificador deve terminar em '0101', que é o seu *anticode*.

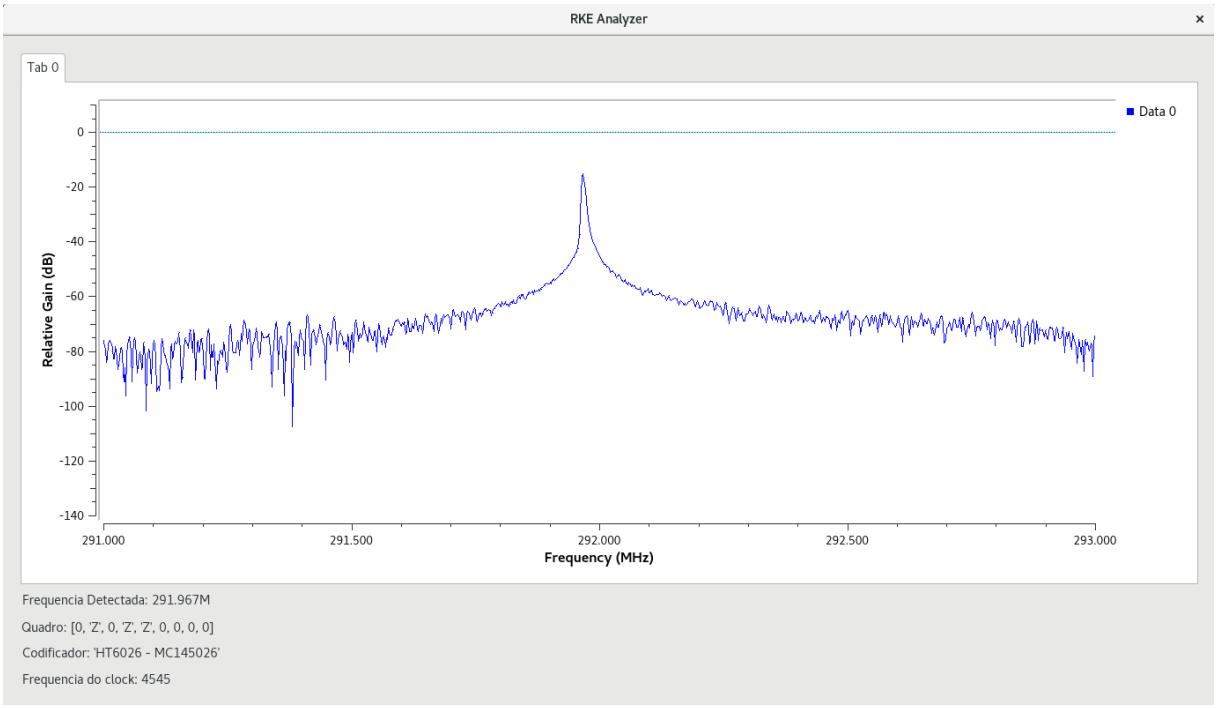


Figura 30 – Interface gráfica do analisador – Identificando codificador HT6026.
Fonte: Elaborada pelo autor.

O valor da frequência de operação detectada medida pelo analisador de controle remoto não é fixa, no entanto uma variação pequena é uma característica normal dos controles remotos. Na Tabela 6 está apresentado os valores medidos para os controles remotos, os mesmos da Tabela 5. Comparando essas duas tabelas é possível analisar o desvio da frequência entre o valor especificado pelo fabricante e o obtido pelo analisador, podendo observar que ao longo das medições os controles que utilizam ressonadores SAW, que são todos que operam em 433,92 MHz, sofrem menos variação comparado com os outros dois controles que não utilizam, com a maior variação de 420 kHz para o controle 2 e a menor para o controle 5, com 43 kHz. Ainda analisando a Tabela 6, é possível observar os valores de frequência de *clock*, em Hz, sendo essa informação fundamental para o funcionamento correto dos controles remotos, é a partir dela que é feito o

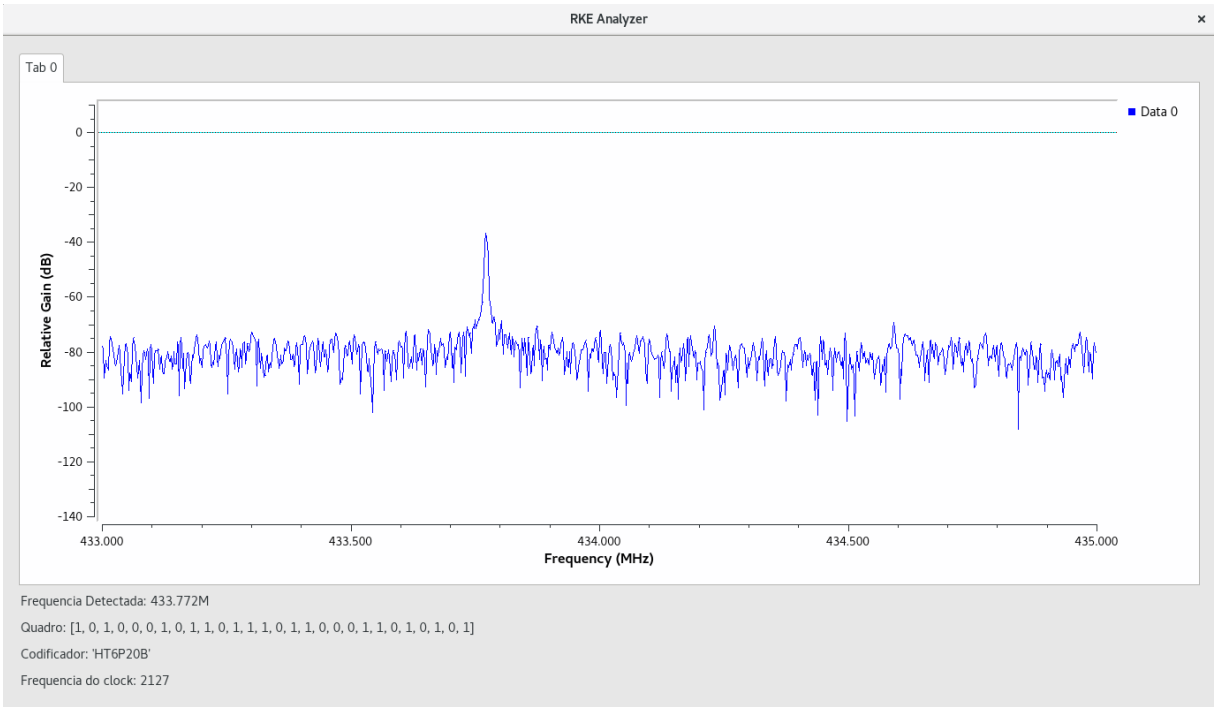


Figura 31 – Interface gráfica do analisador – Identificando codificador HT6P20B.
Fonte: Elaborada pelo autor.

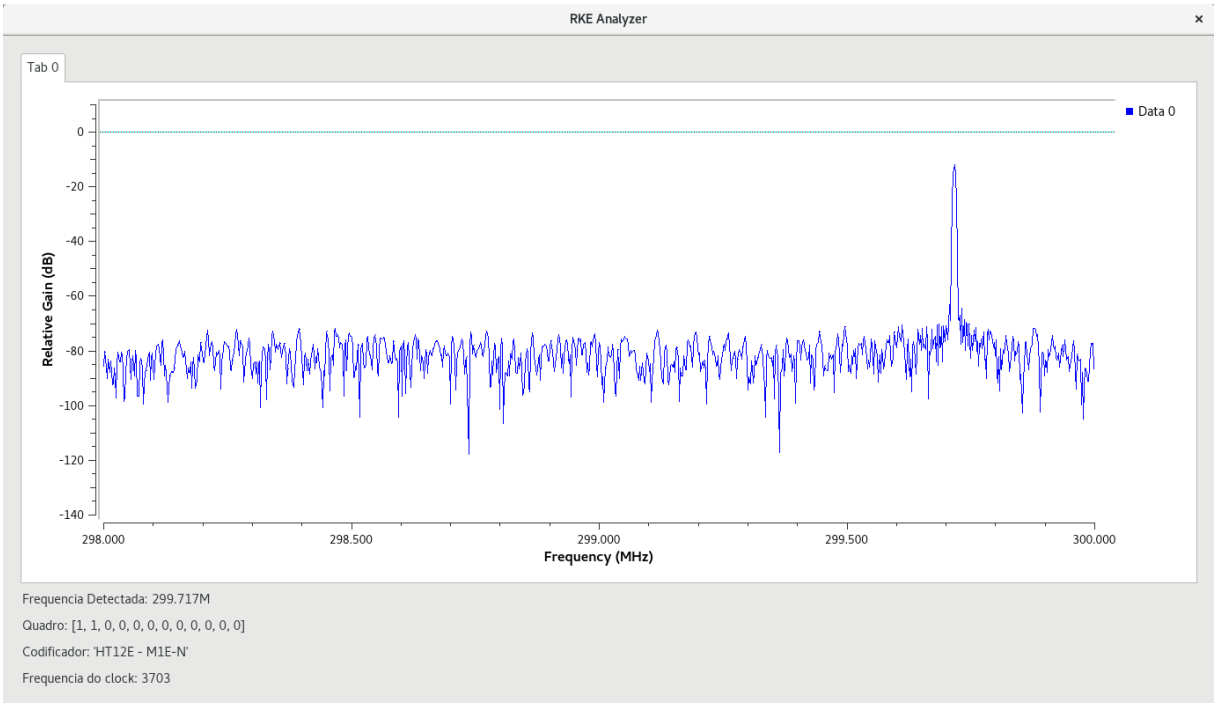


Figura 32 – Interface gráfica do analisador – Identificando os codificadores HT12E e MC145026.
Fonte: Elaborada pelo autor.

sincronismo do sistema, se existir uma variação grande o suficiente, não é possível decodificar o sinal.

Tabela 6 – Informações medidas pelo analisador de controle remoto.

Controle	Codificador	Frequência	*Clock*
1	HT6026	292,298 MHz	4545 Hz
2	MC145026	299,82 MHz	3703 Hz
3	HT6P20B	433,781 MHz	2127 Hz
4	HT6P20B	433,874 MHz	1960 Hz
5	HT6P20B	433,917 MHz	2040 Hz

4.1.2 Teste de distância máxima

Para realizar os testes de distância máxima foram utilizados dois controles, os controles 1 e 3, citados na Tabela 5. Os testes foram realizados da seguinte forma: foram definidas distâncias em visada direta, as quais seriam testados os controles remotos e verificado se o sistema identificaria todas as informações de resposta corretamente. Para uma melhor análise dos resultados, quando o sistema encontrava essas informações corretamente, era armazenado o valor da estatística de teste, calculado pela técnicas do detector de energia.

Antes de analisar os resultados é importante ressaltar alguns pontos. Primeiro, o resultado desses testes é diretamente ligado ao nível de bateria que o controle remoto possui. Segundo ponto, o valor limiar definido para identificação da presença de um sinal foi de 20 dB, ou seja, valores abaixo desse limiar não conseguiriam manter o sintonizador fixado na frequência emitida, pois ele entenderia como ausência de sinal e sintonizaria na próxima frequência.

Como o objetivo do teste era encontrar a maior distância em que o sistema funciona, foram realizados para cada medição pelo menos três tentativas, emitindo até trinta segundos o sinal em cada marca. Caso não fosse encontrada as informações ao final dessas tentativas, o teste era considerado sem sucesso. Foram realizadas tentativas com uma distância de até 20 metros.

Na Tabela 7 podemos analisar os resultados medidos. Para o controle 3 o sistema obteve uma maior distância, conseguindo um máximo de catorze metros, com um nível calculado de 22 dB. Com o controle 1 o sistema conseguiu identificar corretamente as características com um máximo de 8 metros, com um nível em 22 dB.

Tabela 7 – Valores da estatística de teste em relação à distância.

Distância (m)	0,3	0,5	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Controle 1	47	44	41	36	32	31	32	30	26	22	–	–	–	–	–	–
Controle 3	57	54	43	37	35	37	34	35	37	35	27	21	29	24	22	20

4.1.3 Teste de tempo de resposta

O cálculo do tempo de resposta do sistema utilizou três controles remotos com diferentes codificadores, frequências de operação e frequência do *clock*, objetivando encontrar o maior tempo de resposta do sistema. Os controles selecionados foram os controles 1, 2 e 3 citados na Tabela 5. Foram realizadas dez medições para cada controle remoto em cada um dos testes, a

uma distância de aproximadamente 50 cm, sendo que a cada transmissão era trocado o controle e medido o tempo para que as configurações do módulo receptor estivessem diferentes do controle testado. Antes de analisarmos os resultados é importante relembrar que, cada faixa de frequência é sensoriada por 0.2 segundos, outro fato é que o sistema só consegue decodificar a sequência após encontrar o número de amostras por símbolo do sinal transmitido, e assim sincronizar o sinal com os centros de símbolos, a função que atualiza o valor do número de amostras por símbolo é chamada a cada 0.2 segundos.

Buscando avaliar qual módulo apresentava a maior influência no tempo de resposta foram realizados quatro testes com diferentes objetivos, foram medidos os tempos para:

- Detectar somente a frequência de operação.
- Detectar somente a frequência de *clock*.
- Identificar todas as informações do controle com a frequência do sintonizador fixa.
- Identificar todas as informações do controle com o sistema funcionando normalmente.

A Tabela 8 apresenta as medições de tempo de resposta em segundos para identificar a frequência de operação. Lembrando que cada frequência é varrida por 0.2 segundos, e são utilizadas quatro faixas diferentes, o pior cenário teoricamente seria de 0.8 segundos. O tempo médio de resposta foi de 0,72 s, 1,0 s e 0,85 s para os controles 1, 2 e 3 respectivamente. Esses valores em média estão bem próximos do esperado. A Tabela 9 apresenta as medições de tempo de resposta em segundos para identificar a frequência de *clock*, nesse teste a frequência de operação foi fixada em relação ao controle remoto testado. O tempo de resposta médio foi de 0,5 s, 0,49 s e 0,43 s para os controles 1, 2 e 3 respectivamente. É possível observar que para ambos os testes que os valores são bem próximos entre os diferentes controles, isso significa que controles remotos diferentes não possuem impacto no tempo de resposta em relação ao módulo de frequência de *clock* e ao módulo de frequência de operação.

Tabela 8 – Tempo de resposta para identificar frequências de operação.

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	min	max	média
Controle 1	1.04	0.44	0.77	0.88	0.98	1.22	0.90	0.78	0.85	1.21	0.44	1.22	0.72
Controle 2	0.57	0.98	0.83	0.98	1.11	2.31	1.01	0.47	0.63	1.14	0.47	2.31	1.0
Controle 3	0.74	1.25	0.88	1.43	1.25	0.68	0.67	0.70	0.47	0.44	0.44	1.43	0.85

Tabela 9 – Tempo de resposta para identificar frequências de clock.

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	min	max	média
Controle 1	0.29	0.25	0.44	0.68	0.33	0.59	0.52	0.78	0.74	0.38	0.25	0.78	0.5
Controle 2	0.12	0.54	0.48	0.45	0.71	0.27	0.64	0.31	0.59	0.79	0.12	0.79	0.49
Controle 3	0.21	0.65	0.21	0.61	0.68	0.55	0.22	0.21	0.64	0.41	0.21	0.68	0.439

A Tabela 10 apresenta as medições de tempo de resposta em segundos do sistema com a frequência fixa identificando todas as informações do controle remoto, já a Tabela 11 apresenta as medições de tempo de resposta em segundos do sistema operando normalmente, variando

entre as quatro frequências. Analisando o resultado dessas duas tabelas e comparando com os resultados das tabelas 8 e 9, é possível fazer algumas conclusões. O tempo para decodificar um quadro completo insere um acréscimo considerável no sistema, os tempos médios das tabelas 10 e 11 aumentaram consideravelmente em comparação as tabelas 8 e 9. O controle 1 que possui o codificador HT6026 possui o tempo mais elevado nos dois últimos testes, isso se deve a lógica do decodificador que realiza o processo de decodificação para todos os codificadores até encontrar um quadro completo, e o codificador do controle 1 é o último a ser escolhido pelo bloco *decoder*. A ordem dos decodificadores selecionados são: HT12E, HT6P20B e HT6026, com isso é possível perceber que o aumento na quantidade de codificadores impactaria muito no tempo de resposta. Existe também um acréscimo de tempo devido a variação da frequência de operação, como por exemplo nas tabelas 10 e 11, o tempo médio do controle 3 teve um aumento maior que o dobro, de 0,825 s para 1,791 s.

É importante mencionar também que o tempo de resposta do sistema não é necessariamente o tempo necessário de acionamento do controle remoto, o sistema possui um *delay* interno de processamento, isso foi percebido durante os testes em que a frequência foi fixada, em que apenas um único e breve acionamento no controle retornava as informações com um atraso.

Tabela 10 – Tempo de resposta com frequências fixas.

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	min	max	média
Controle 1	1.79	1.85	3.18	4.01	2.43	3.79	2.33	1.52	2.61	1.47	1.47	4.01	2.49
Controle 2	1.38	0.88	0.64	0.52	0.89	0.71	0.85	0.55	1.12	0.68	0.52	1.38	0.822
Controle 3	0.81	1.31	0.82	0.74	0.74	0.79	0.63	0.88	0.80	0.73	0.74	1.31	0.825

Tabela 11 – Tempo de resposta do sistema completo.

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	min	max	média
Controle 1	4.99	6.51	3.37	5.0	2.78	2.16	2.87	3.55	2.02	3.19	2.02	6.51	3.64
Controle 2	1.19	1.01	1.53	1.50	1.30	1.43	1.04	1.04	1.71	1.57	1.01	1.57	1.33
Controle 3	2.5	1.24	0.96	1.79	1.21	2.57	1.49	1.19	2.59	2.37	0.96	2.59	1.791

5 CONCLUSÕES

Este trabalho teve como objetivo desenvolver um analisador de controle remoto na plataforma GNU Radio utilizando um *dongle* RTL-SDR, visando detectar e apresentar as características mais importantes para a manutenção de um controle remoto, utilizando a técnica de rádio definido por *software*.

Os objetivos gerais e específicos do sistema foram alcançados. O analisador de controle remoto desenvolvido é capaz de: receber sinais transmitidos por controles remotos que utilizam a modulação OOK, identificar a frequência de *clock*, detectar a frequência de operação, decodificar a sequência de transmissão para os codificadores apresentados (HT6026, HT12E e HT6P20B) ou seus similares que utilizem o mesmo protocolo de codificação, e por fim apresentar essas informações encontradas na interface de usuário.

Os testes realizados no sistema e apresentados no Capítulo 4 comprovam o funcionamento correto para os controles remotos compatíveis com as definições do trabalho. Com o resultado desses testes também são apresentadas análises relacionadas ao desempenho, como por exemplo: tempo de resposta e distância máxima de funcionamento. Com o teste de tempo de resposta foi possível perceber que o módulo decodificador possui um atraso de processamento significativo, devido a lógica de decodificar uma sequência recebida com diferentes decodificadores até encontrar um quadro completo. O aumento do número de decodificadores tornaria o tempo de resposta para os últimos decodificadores da lista muito grande. No entanto, para solucionar esse problema é possível utilizar processamento em paralelo, no qual cada decodificador seria um novo bloco, com um novo fluxo.

Com o teste de distância ficou evidente a falta de segurança dos controles de código fixo. Levando em consideração a distância máxima obtida nos testes de catorze metros e o tempo de resposta, principalmente para o caso de resposta com frequências fixas, como por exemplo o controle 2 da Tabela 10 que obteve uma resposta miníma de 0,52 s, mostrando que é possível obter o código de um controle remoto em um curto período.

O desenvolvimento do analisador de controle remoto utilizando o conceito de rádio definido por software se mostrou suficiente para alcançar os objetivos, o sistema foi inteiramente desenvolvido utilizando os blocos disponíveis pelo *GNU Radio* e blocos implementados em Python, permitindo abstrair todo o uso de *hardware* apenas para o *dongle* RTL-SDR.

O uso da plataforma *GNU Radio* foi fundamental para o desenvolvimento do trabalho, ela disponibilizou as ferramentas necessárias para a implementação de um sistema SDR. Porém foram encontradas algumas dificuldades durante o desenvolvimento em relação ao uso dessa plataforma, principalmente pela falta de documentação de alguns blocos de processamento disponíveis. Além disso, alguns blocos não permitiam a atualização de parâmetros de uma forma simples, como o caso do bloco *Symbol Sync*, que fez necessário o uso de fluxos com *tags* para ser possível realizar a atualização do parâmetro de *clock* do sincronizador. Outra dificuldade encontrada foi controlar

o uso da CPU pelo *GNU Radio*, sendo necessário testes com diferentes valores de parâmetros, e diversas abordagens para diminuir o uso da CPU, resultando na inserção de um *squelch* e da subamostragem no módulo receptor, que reduziu o número de amostras processadas no sistema.

5.1 Trabalhos futuros

Recomenda-se para trabalhos futuros a implementação de novas funcionalidades no sistema, como por exemplo:

- Adicionar novos decodificadores, assim como identificar decodificadores do tipo *hopping code*.
- Contar o número de quadros completos enviados por segundo.
- Receber sinais de controles remotos que possuem outras formas de modulação, como por exemplo a FSK.
- Melhorar o sensoriamento das faixas de frequência, conseguindo varrer um maior número de faixas sem perder desempenho, permitindo encontrar controles remotos que estejam com um desvio de frequência muito grande.
- Implementar novas técnicas de sensoriamento espectral, como por exemplo o método ciclo-estacionário.
- Utilizar uma abordagem mais eficiente para decodificar a sequência recebida com diversos decodificadores diferentes.
- Fazer a interface gráfica para o usuário com mais funcionalidades.

REFERÊNCIAS

AGÊNCIA NACIONAL DE TELECOMUNICAÇÕES. *Resolução n 680, de 27 de junho de 2017*. [S.l.], 2017. Citado na página 27.

ALVAREZ, P. et al. *Energy Detection and Eigenvalue Based Detection: An Experimental Study Using GNU Radio*. [S.l.], 2011. Citado na página 32.

ANTHES, J. *OOK, ASK and FSK Modulation in the Presence of an Interfering signal*. [S.l.], 2007. Citado na página 27.

ATAPATTU, S.; TELLAMBURA, C.; JIANG, H. *Energy Detection for Spectrum Sensing in Cognitive Radio*. [S.l.]: Springer, 2014. Citado 2 vezes nas páginas 32 e 33.

BALL, D.; NAIK, N.; JENKINS, P. *Spectrum Alerting System Based on Software Defined Radio and Raspberry Pi*. [S.l.], 2017. Citado na página 30.

CHEN, Y.-C.; CHIEN, T.-H. *A Simple Approach for Power Signal Frequency Determination on Virtual Instrument Platform*. 2015. Disponível em: <http://www.naturalspublishing.com/files/published/60r8y1241yf6s5.pdf>. Acesso em: 04.11.2018. Citado na página 46.

CSETE, A. *Gqrx*. [S.l.]: GitHub, 2018. <https://github.com/csete/gqrx>. Citado na página 31.

EETECH (Ed.). *Understanding I/Q Signals and Quadrature Modulation*. 2018. Disponível em: <https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/radio-frequency-demodulation/understanding-i-q-signals-and-quadrature-modulation/>. Acesso em: 10.12.2018. Citado na página 30.

FANAN, A. et al. *Comparison of Spectrum Occupancy Measurements using Software Defined Radio RTL-SDR with a Conventional Spectrum Analyzer approach*. [S.l.], 2015. Citado na página 30.

GATES, W. *Identify RF Remote Control*. 2013. Disponível em: <https://www.solidremote.com/blog/identify-rf-remote-control/>. Acesso em: 01.05.2018. Citado 2 vezes nas páginas 23 e 28.

GNU RADIO. *GNURadioCompanion*. 2017. 2017-07-17. Disponível em: <https://wiki.gnuradio.org/index.php/GNURadioCompanion>. Citado na página 31.

GNU RADIO. *What is GNU Radio?* 2017. 2017-03-20. Disponível em: <https://wiki.gnuradio.org/index.php/What_is_GNU_Radio3F>. Citado 2 vezes nas páginas 22 e 31.

GNU RADIO. *Guided Tutorial GRC*. 2018. 2018-09-15. Disponível em: <https://wiki.gnuradio.org/index.php/GNURadioCompanion>. Citado na página 36.

HOLTEK SEMICONDUCTOR INC. *HT12A/HT12E* 2^{12} *Series Encoders*. [S.l.], 2000. Citado 2 vezes nas páginas 24 e 25.

HOLTEK SEMICONDUCTOR INC. *HT6P20X Series* 2^{24} *OTP Encoder*. [S.l.], 2003. Rev. 1.40. Citado na página 26.

HOLTEK SEMICONDUCTOR INC. *HT6026 Remote Control Encoder*. [S.l.], 2009. Rev. 1.10. Citado 2 vezes nas páginas 25 e 26.

LAKE, M. *HOW IT WORKS; Remote Keyless Entry: Staying a Step Ahead of Car Thieves*. 2001. Disponível em: <https://www.nytimes.com/2001/06/07/technology/how-it-works-remote-keyless-entry-staying-a-step-ahead-of-car-thieves.html>. Citado na página 21.

LAUFER, C. *The Hobbyist's Guide to RTL-SDR*. [S.l.]: Pearson, 2015. Citado na página 30.

LINX TECHNOLOGIES. *The Basics of Remote Control and Remote Keyless Entry*. [S.l.], 2012. AN-00320. Citado na página 27.

MARNEWECK, K. *An Introduction to KEELOQ® Code Hopping*. [S.l.], 1996. Citado na página 23.

MAžEIKA, L.; DRAUDVILIENė, L. *Analysis of the zero-crossing technique in relation to measurements of phase velocities of the Lamb waves*. 2010. Disponível em: <http://www.naturalspublishing.com/files/published/60r8y1241yf6s5.pdf>. Acesso em: 04.11.2018. Citado na página 46.

MICREL INC. *MICRF220*. [S.l.], 2015. Citado na página 29.

MICROCHIP TECHNOLOGY INC. *HCS301*. [S.l.], 2001. Citado na página 23.

MUKHTAR1, N. J. et al. *Circuit Modeling of Surface Acoustic Wave (SAW) Resonator with Circular Geometry* . [S.l.], 2016. Citado na página 28.

NAGURNEY, L. S. *Software Defined Radio in the Electrical and Computer Engineering Curriculum*. [S.l.], 2009. Citado na página 30.

PRANDONI, P.; VETTERLI, M. *SIGNAL PROCESSING FOR COMMUNICATIONS*. [S.l.]: CRC Press, 2008. Citado na página 28.

RAPAPPORT, T. S. *Comunicação Sem Fio: Princípios e praticas*. [S.l.]: Pearson, 2008. Citado na página 26.

RIBAS, A. de O. P. *Sensoriamento Espetral Por Detecção de Energia om Duplo Limiar sob Canais em Desvanecimento κ–μ em Redes de Rádio Cognitivo*. Dissertação (Dissertação de Mestrado) — Universidade de Brasília, 2015. Citado 2 vezes nas páginas 32 e 33.

SDR FORUM. *SDRF Cognitive Radio Definitions*. 2007. 2007-11-08. Disponível em: <http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf>. Citado na página 29.

SELVA, A. F. B. et al. *Introduction to the Software-defined Radio Approach*. [S.l.], 2012. Citado 3 vezes nas páginas 21, 29 e 30.

SILVA, W. S. et al. *Introdução a Rádios Definidos por Software com aplicações em GNU Radio*. [S.l.], 2017. Citado 2 vezes nas páginas 29 e 31.

SKLAR, B. *Digital Communications: Fundamentals and Applications*. [S.l.]: Prentice Hall, 2001. Citado 2 vezes nas páginas 27 e 29.

Hirohide Suda e Matthew J. Lehmer. *Remote Keyless Entry System*. 2004. US 6,718,240 B1. Disponível em: <https://patentimages.storage.googleapis.com/ae/8a/3f/8df7ef2c8104f4/US6718240.pdf>. Citado 2 vezes nas páginas 21 e 23.

WALTERS, M. *Inspectrum*. [S.l.]: GitHub, 2018. <https://github.com/miek/inspectrum>. Citado na página 31.

ŠOLC, T. *Notes On M&M Clock Recovery*. 2015. Disponível em: <https://www.tablix.org/~avian/blog/archives/2015/03/notes_on_m_m_clock_recovery/>. Acesso em: 23.06.2018. Citado na página 29.