

Eduardo Noboro Tominaga

**DESENVOLVIMENTO DE EXPERIMENTOS COM O  
RASPBERRY PI PARA O ENSINO DE COMUNICAÇÕES**

Trabalho de Conclusão de Curso  
submetido ao Departamento de  
Engenharia Elétrica e Eletrônica da  
Universidade Federal de Santa  
Catarina para a obtenção do título de  
Bacharel em Engenharia Eletrônica.  
Orientador: Prof. Dr. Richard Demo  
Souza

Florianópolis  
2018

Ficha de identificação da obra elaborada pelo autor  
através do Programa de Geração Automática da Biblioteca Universitária  
da UFSC.

Tominaga, Eduardo Noboro

Desenvolvimento de Experimentos com o Raspberry  
Pi para o Ensino de Comunicações / Eduardo Noboro  
Tominaga ; orientador, Richard Demo Souza, 2018.  
231 p.

Trabalho de Conclusão de Curso (graduação) -  
Universidade Federal de Santa Catarina, Centro  
Tecnológico, Graduação em Engenharia Eletrônica,  
Florianópolis, 2018.

Inclui referências.

1. Engenharia Eletrônica. 2. Plataforma didática.  
3. Raspberry Pi. 4. Rádio Definido por Software. 5.  
Comunicações. I. Souza, Richard Demo. II.  
Universidade Federal de Santa Catarina. Graduação em  
Engenharia Eletrônica. III. Título.



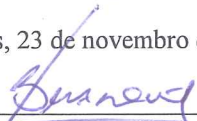


Eduardo Noboro Tominaga


**DESENVOLVIMENTO DE EXPERIMENTOS COM O  
RASPBERRY PI PARA O ENSINO DE COMUNICAÇÕES**

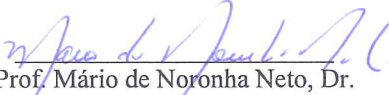
Este Trabalho foi julgado adequado para obtenção do Título de Bacharel  
em Engenharia Eletrônica e aprovado em sua forma final pela Banca  
Examinadora

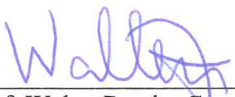
Florianópolis, 23 de novembro de 2018.

  
Prof. Jefferson Luiz Brum Marques, Phd.  
Coordenador do Curso

**Banca Examinadora:**

  
Prof. Richard Demo Souza, Dr.  
Orientador  
Universidade Federal de Santa Catarina

  
Prof. Mário de Noronha Neto, Dr.  
Instituto Federal de Santa Catarina

  
Prof. Walter Pereira Carpes Jr., Dr.  
Universidade Federal de Santa Catarina



Este trabalho é dedicado a Deus, meus pais, meus irmãos, a todos os professores que me motivaram ao longo da graduação, e a todos os meus amigos.





## AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer a Deus. Confesso que, no passado, já houve momentos em que questioneei a Sua existência. Recentemente, passei por um momento extremamente conturbado na minha vida pessoal, que me colocou num estado de profunda tristeza. Eu me vi numa situação na qual estava sem perspectivas positivas para o meu futuro. Porém, a partir de então aconteceram tantas coisas boas na minha vida (coisas pelas quais eu não procurei, apenas aconteceram de forma inesperada) que se tornou impossível para mim não acreditar na Sua existência. Hoje eu sei que todas essas coisas boas que aconteceram têm acontecido na minha vida são graças a intervenção dEle.

Gostaria também de agradecer profundamente aos meus pais, Osvaldo e Ana Paula Tominaga. Vocês sempre me apoiaram de todas as formas possíveis, e não houve um único dia sequer em que deixaram de acreditar no meu potencial. Eu jamais teria conseguido passar no vestibular e chegar ao final da graduação sem o apoio psicológico, educacional e financeiro que vocês me forneceram. Nos dias felizes e nos dias tristes, vocês sempre estiveram ao meu lado. E mesmo se eu tivesse fracassado, eu sei que vocês teriam continuado a me amar e me apoiar da mesma forma. Jamais duvidarei do amor de vocês por mim. Por isso, também dedico todas as minhas conquistas até hoje e todo o meu sucesso que ainda está por vir a vocês.

Agradeço também aos meus irmãos Rafael e Osvaldo Tominaga por terem sido meus companheiros nesses anos de graduação. Em especial, tenho muito que agradecer a você Osvaldo, pois além de ser meu irmão e colega de apartamento, também foi o meu colega na grande maioria das disciplinas da graduação. Em muitas das disciplinas difíceis do curso, eu sofri com notas baixas, o que muitas vezes me levou a considerar a desistência das disciplinas durante o semestre e até mesmo a desistência do curso. Porém, você sempre me apoiou e me motivou, sempre acreditou que eu tinha condições de recuperar as notas e ser aprovado nas disciplinas. Sempre esteve disponível para me explicar em casa os conteúdos das disciplinas, me ajudar com a resolução de exercícios e ser minha dupla em muitos trabalhos difíceis. Se hoje eu estou a um passo de me formar (e sem nenhuma reprovação no meu histórico), essa conquista se deve em grande parte a você.

Quero agradecer também aos professores Bartolomeu Ferreira Uchoa Filho, Danilo Siva e Richard Demo Souza, do Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina. Ter cursado as suas disciplinas optativas de Introdução a

Codificação, Rádio Definido por Software e Comunicações Móveis na segunda metade de 2017 foi o maior divisor de águas em toda a minha trajetória acadêmica. Vocês três foram os responsáveis por eu ter descoberto o meu amor pela área de Comunicações. Hoje eu tenho a certeza de que é com isso que eu quero trabalhar como futuro Engenheiro Eletrônico e que quero me aprofundar cada vez mais na área. Vocês definiram meus rumos profissionais, e todas as minhas futuras conquistas como profissional se deverão também a vocês.

Agradecimentos especiais para o professor Richard Demo Souza, que aceitou meus pedidos para ser meu orientador de TCC e orientador de estágio. Sou extremamente grato a você, professor, pois você nunca me negou o agendamento de uma reunião e nunca ignorou um e-mail meu. Você me indicou dezenas de artigos extremamente interessantes que contribuíram muito com a minha formação. Você sempre foi muito atencioso e preocupado com o desenvolvimento de nosso trabalho, fornecendo inúmeras sugestões de modo a torná-lo o mais interessante e didático possível. Sempre respondeu pacientemente a todas as minhas dúvidas, por mais bobas que elas fossem. Além disso, também sempre se mostrou disponível para ouvir meus anseios profissionais e acadêmicos e me dar os melhores conselhos possíveis. Por todas as nossas muitas horas de conversas, meu muito obrigado. Tenho uma enorme admiração por você como pessoa, pesquisador e professor, e por isso sempre será uma referência para mim.

Agradeço também ao meu professor de inglês, Daniel Magno. Para mim, você foi muito mais do que um professor que despertou minha paixão pelo estudo do idioma. Você é meu amigo, uma pessoa que sempre acreditou no meu potencial, sempre me motivou e contribuiu muito para que eu me tornasse um ser humano melhor. Se hoje eu sou fluente em inglês, essa conquista em grande parte foi graças a você. Você me mostrou que o inglês é muito mais do que aprendemos num livro ou numa sala de aula. O inglês é um universo fascinante a ser explorado. As tuas experiências internacionais bem sucedidas também sempre foram uma grande fonte de inspiração para mim. Sei que a fluência em inglês abrirá muitas portas para mim ao longo da minha trajetória profissional, e dedico cada uma dessas portas abertas a você.

Por último, mas não menos importante, gostaria de dedicar algumas palavras para meus amigos Vinícius Freitas, Júlia Bueno e Hadassa Brites. De todas as coisas boas que aconteceram na minha vida recentemente, ter conhecido vocês e ter me tornado amigo de vocês com certeza foi uma delas. A amizade de vocês sempre foi algo que me fez

muito bem e me deu motivação para superar meus desafios. Por todas as nossas conversas, conselhos, “rolês” e risadas, meu muito obrigado.

Vini, nós podemos não ter o mesmo sangue, mas mesmo assim você é como um irmão para mim. Eu aprendo muito contigo, me inspiro em você para tomar muitas decisões. Sempre lembrarei das nossas conversas, sobre como podíamos conversar sobre qualquer assunto com total liberdade e sermos compreendidos um pelo outro.

Ju, você é simplesmente uma das garotas mais incríveis que eu já conheci, e sei que nunca mais vou conhecer alguém como você. Poder fazer parte da tua vida, sendo teu amigo, te aconselhando e te apoiando em tuas decisões é algo que me deixa muito feliz.

Dassa, toda vez que eu te vejo não consigo conter o sorriso. Sinto uma enorme vontade de te abraçar e te proteger. Eu simplesmente me sinto muito bem na sua companhia. Você é uma das pessoas mais amáveis que já tive a oportunidade de conhecer.

Eu aprendi muito com cada um de vocês, e de coração também espero que vocês tenham aprendido algo bom comigo. Não importa o tempo ou a distância, meses ou milhares de quilômetros de distância que possam nos separar, você três serão meus amigos para sempre. Aonde quer que eu vá, me lembrarei de vocês, e estarei desejando que todos estejam bem e felizes. Nada nesse mundo poderá diminuir o carinho e a admiração que eu tenho por vocês.



Um bom professor não é aquela pessoa que apenas enche o teu copo, mas sim aquela que te mantém com sede por novos conhecimentos.  
(Autor desconhecido)



## RESUMO

Este trabalho consiste num conjunto de experimentos didáticos que podem ser aplicados em aulas de laboratório de disciplinas introdutórias de Comunicações e Processamento Digital de Sinais. Os experimentos utilizam uma plataforma didática baseada no Raspberry Pi, um computador de baixo custo e tamanho reduzido, que é utilizado aqui como um processador digital de sinais. Uma placa de som USB é conectada à placa com o objetivo de fornecer uma entrada e uma saída analógicas, isto é, atua como um ADC e um DAC. Os experimentos desenvolvidos abrangem temas como quantização, filtragem digital, modulações analógicas em amplitude e frequência, modulações digitais em banda base e modulações digitais em banda passante. Os sistemas de comunicações utilizados nos experimentos são implementados no Simulink por diagramas de blocos que, através da ferramenta de geração automática de código do MATLAB, são traduzidos em códigos executáveis e carregados no Raspberry Pi. O principal objetivo dos experimentos é permitir que os estudantes possam fixar os principais conceitos relacionados a fundamentos de comunicações analógicas e digitais ao ter contato com sinais reais, isto é, que podem ser mensurados com o uso de um osciloscópio. Diferentemente das abordagens tradicionais, que utilizam apenas simulações em aulas de laboratório, os experimentos aqui propostos tem o potencial de aprimorar o processo de aprendizagem ao elevar a motivação dos alunos.

**Palavras-chave:** Plataforma Didática. Raspberry Pi. Rádio Definido por Software. Comunicações. Processamento Digital de Sinais.





## ABSTRACT

This work consists in a set of didactic experiments that can be applied in practical classes of introductory subjects related to Communications and Digital Signal Processing. The experiments use a didactic platform based on the Raspberry Pi, a low cost and small-sized computer, that is applied here as a digital signal processor. A USB sound card is connected to the board in order to provide an analog input and an analogic output, i. e., it works as an ADC and an DAC. The experiments developed cover themes like quantization, digital filtering, amplitude and frequency analogic modulations, band base digital modulations and band pass digital modulations. The communication systems used on the experiments are implemented in Simulink with block diagrams that, by an automatic code generation tool of MATLAB, are translated in executable files that can be loaded and executed on the Raspberry Pi. The main objective of the experiments is allow students to fix the main concepts related to fundamentals of analogic and digital communications through the contact with real signals, i. e., that can be measured with an oscilloscope. Differently from the traditional approaches, that use only simulations on practical classes, the experiments proposed here have the potential to improve the learning process by the enhancement of students motivation.

**Keywords:** Didactic platform. Raspberry Pi. Software Defined Radio. Communications. Digital Signal Processing.



## LISTA DE FIGURAS

|   |    |
|---|----|
| Figura 1 – Componentes de uma arquitetura SDR simples e conceitual.<br>.....                                | 42 |
| Figura 2 – Primeira geração da arquitetura de SDR.....  | 43 |
| Figura 3 – Segunda geração da arquitetura de SDR.....   | 44 |
| Figura 4 – Terceira geração da arquitetura de SDR. ....   | 44 |
| Figura 5 – Raspberry Pi 2 Model B.....  | 45 |
| Figura 6 – Modelo do Simulink que transforma o Raspberry Pi num DSP que realiza filtragem digital. ....     | 48 |
| Figura 7 – Diagrama da plataforma didática baseada no Raspberry Pi. ....                                    | 51 |
| Figura 8 – Modelo do Simulink utilizado no Experimento 0.....   | 57 |
| Figura 9 – Janela de configurações do bloco ALSA Audio Capture.....   | 58 |
| Figura 10 – Janela de configurações do bloco ALSA Audio Playback. ....                                      | 59 |
| Figura 11 – Menu a ser acessado para a configuração do Simulink.....  | 61 |
| Figura 12 – Definição do hardware alvo. ....  | 62 |
| Figura 13 – Seleção do diretório no qual o arquivo executável será salvo. ....                              | 63 |
| Figura 14 – Parâmetros no Simulink para execução em modo externo. ....                                      | 64 |
| Figura 15 – Botão Stop utilizado para interromper a execução do experimento.....                            | 64 |
| Figura 16 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda senoidal de 1 kHz.....    | 65 |
| Figura 17 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda triangular de 1 kHz.....  | 66 |
| Figura 18 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda quadrada de 1 kHz. ....   | 66 |
| Figura 19 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda triangular de 5 kHz.....  | 67 |
| Figura 20 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda quadrada de 5 kHz. ....   | 68 |
| Figura 21 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda triangular de 10 kHz..... | 69 |
| Figura 22 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda quadrada de 10 kHz. ....  | 69 |
| Figura 23 – Modelo do Simulink utilizado para gerar um sinal senoidal na saída da placa de som USB.....     | 72 |
| Figura 24 – Janela de configuração do bloco Sine Wave.....  | 73 |
| Figura 25 – Janela de configurações do bloco Data Type Conversion..   | 75 |
| Figura 26 – Janela de configurações do bloco Matrix concatenate.....  | 76 |
| Figura 27 – Forma de onda senoidal gerada pelo Raspberry Pi. ....   | 77 |

|   |    |
|---|----|
| Figura 28 – Espectro do sinal senoidal gerado pelo Raspberry Pi. ....   | 78 |
| Figura 29 – Modelo do Simulink utilizado para gerar uma forma de onda triangular na saída da placa de som USB. ....                             | 78 |
| Figura 30 – Janela de configurações do bloco <i>Triangle Generator</i> . ....   | 79 |
| Figura 31 – Janela de configurações do bloco <i>Gain</i> . ....   | 79 |
| Figura 32 – Janela de configurações do bloco <i>Buffer</i> . ....   | 80 |
| Figura 33 – Forma de onda triangular gerada pelo Raspberry Pi. ....   | 80 |
| Figura 34 – Espectro da forma de onda triangular gerado pelo Raspberry Pi. ....   | 81 |
| Figura 35 – Modelo do Simulink utilizado para gerar uma forma de onda quadrada na saída da placa de som USB. ....                               | 82 |
| Figura 36 – Janela de configurações do bloco <i>Pulse Generator</i> . ....  | 82 |
| Figura 37 – Forma de onda quadrada gerada pelo Raspberry Pi. ....   | 83 |
| Figura 38 – Período da forma de onda quadrada gerada pelo Raspberry Pi. ....  | 84 |
| Figura 39 – Espectro da forma de onda quadrada gerada pelo Raspberry Pi. ....   | 84 |
| Figura 40 – Modelo do Simulink utilizado para gerar diferentes formas de onda com o Raspberry Pi. ....  | 85 |
| Figura 41 – Janela de configurações do bloco <i>Waveform Generator</i> . ..   | 86 |
| Figura 42 – Onda dente de serra gerada pelo Raspberry Pi. ....  | 87 |
| Figura 43 – Soma de senóides gerada pelo Raspberry Pi. ....   | 87 |
| Figura 44 – Espectro da onda dente de serra gerada pelo Raspberry Pi. ....  | 88 |
| Figura 45 – Espectro da soma de senóides gerada pelo Raspberry Pi. .  | 88 |
| Figura 46 – Modelo do Simulink utilizado para reproduzir um arquivo .wav na saída da placa de som USB. ....                                     | 90 |
| Figura 47 – Janela de configurações do bloco <i>Signal from Workspace</i> . ....  | 91 |
| Figura 48 – Sinal de ECG gerado pelo Raspberry Pi. ....   | 92 |
| Figura 49 – Espectro do sinal de ECG gerado pelo Raspberry Pi. ....   | 92 |
| Figura 50 – Modelo do Simulink utilizado para transformar o Raspberry Pi num filtro digital FIR. ....   | 94 |
| Figura 51 – Componentes internos do subsistema “Filtro FIR”, utilizado no modelo do Simulink que transforma o Raspberry Pi num filtro FIR. .... | 95 |
| Figura 52 – Modelo do Simulink utilizado para transformar o Raspberry Pi num filtro digital do tipo IIR. ....                                   | 96 |
| Figura 53 – Componentes internos do subsistema “Filtro IIR”, utilizado no modelo do Simulink que transforma o Raspberry Pi num filtro FIR. .... | 97 |
| Figura 54 – Modelo do Simulink utilizado para transformar o Raspberry Pi num filtro digital. ....   | 98 |

|   |     |
|---|-----|
| Figura 55 – Janela de configurações do bloco Digital Filter Design (FDATool).....   | 100 |
| Figura 56 – Aba <i>Realize Model</i> na ferramenta FDATool, acessada para a implementação automática de filtros em diagramas de blocos. ....    | 101 |
| Figura 57 – Modelo do Simulink utilizado para realizar a filtragem de sinais de áudio. ....   | 103 |
| Figura 58 – Localização do botão <i>Deploy to Hardware</i> no Simulink. ...   | 70  |
| Figura 59 – Relação de processos em execução no Raspberry Pi. ....  | 71  |
| Figura 60 – Modelo do Simulink que transforma o Raspberry Pi num quantizador. ....  | 104 |
| Figura 61 – Janela de configurações do bloco <i>Multiport Selector</i> . ....   | 105 |
| Figura 62 – Componentes internos do subsistema AGC que pode ser observado na Figura 60. ....  | 105 |
| Figura 63 – Janela de configuração do bloco <i>Quantizer</i> . ....   | 106 |
| Figura 64 – Sinal de saída observado quando o Raspberry Pi foi configurado como um quantizador com intervalo de quantização igual a 1. ....     | 107 |
| Figura 65 – Sinal de saída observado quando o Raspberry Pi foi configurado como um quantizador com intervalo de quantização igual a 0.5. ....   | 108 |
| Figura 66 – Sinal de saída observado quando o Raspberry Pi foi configurado como um quantizador com intervalo de quantização igual a 0.333. .... | 109 |
| Figura 67 – Sinal de saída observado quando o Raspberry Pi foi configurado como um quantizador com intervalo de quantização igual a 0.333. .... | 110 |
| Figura 68 – Sinal de voz reproduzido pelo Raspberry Pi. ....  | 111 |
| Figura 69 – Espectro do sinal de voz (sem limitação de frequência) reproduzido pelo Raspberry Pi. ....  | 112 |
| Figura 70 – Espectro do sinal de voz (com largura de banda limitada a 3.5 kHz) reproduzido pelo Raspberry Pi. ....                              | 113 |
| Figura 71 – Modelo do Simulink utilizado para transformar o Raspberry Pi num quantizador de sinais de áudio. ....                               | 114 |
| Figura 72 – Sinal de voz quantizado pelo Raspberry Pi com o uso de intervalo de quantização igual a 0.1 (1). ....                               | 115 |
| Figura 73 – Sinal de voz quantizado pelo Raspberry Pi com o uso de intervalo de quantização igual a 0.1 (2). ....                               | 115 |
| Figura 74 – Modelo do Simulink que gera um sinal modulado DSB-SC. ....  | 117 |
| Figura 75 – Sinal modulado DSB-SC gerado pelo Raspberry Pi. ....  | 118 |
| Figura 76 – Espectro do sinal DSB-SC gerado pelo Raspberry Pi. ....   | 118 |

|   |     |
|---|-----|
| Figura 77 – Utilização de cursores no espectro do sinal DSB-SC. ....  | 119 |
| Figura 78 – Modelo do Simulink que gera um sinal modulado DSB-SC cujo sinal de mensagem é uma gravação de voz. ....             | 120 |
| Figura 79 – Sinal modulado DSB-SC gerado pelo Raspberry Pi, cujo sinal de mensagem é um sinal de voz. ....                      | 121 |
| Figura 80 – Espectro do sinal modulado DSB-SC gerado pelo Raspberry Pi, cujo sinal de mensagem é um sinal de voz. ....          | 122 |
| Figura 81 – Modelo do Simulink que gera um sinal modulado AM... ..  | 123 |
| Figura 82 – Sinal modulado AM gerado pelo Raspberry Pi. ....  | 124 |
| Figura 83 – Espectro do sinal AM gerado pelo Raspberry Pi. ....   | 125 |
| Figura 84 – Utilização de cursores no espectro do sinal AM (1). ....  | 125 |
| Figura 85 – Utilização de cursores no espectro do sinal AM (2). ....  | 126 |
| Figura 86 – Modelo do Simulink que gera um sinal modulado AM cujo sinal de mensagem é uma gravação de voz. ....                 | 127 |
| Figura 87 – Sinal modulado AM gerado pelo Raspberry Pi, cujo sinal de mensagem é um sinal de voz. ....                          | 127 |
| Figura 88 – Espectro do sinal modulado AM gerado pelo Raspberry Pi, cujo sinal de mensagem é um sinal de voz. ....              | 128 |
| Figura 89 – Modelo do Simulink que gera um sinal modulado FM. ..  | 128 |
| Figura 90 – Janela de configurações do bloco <i>Discrete-Time VCO</i> ....  | 129 |
| Figura 91 – Sinal FM gerado pelo Raspberry Pi. ....   | 130 |
| Figura 92 – Espectro do sinal FM gerado pelo Raspberry Pi. ....   | 131 |
| Figura 93 – Modelo do Simulink que transforma o Raspberry Pi num modulador FM cujo sinal de mensagem é um sinal de voz. ....    | 132 |
| Figura 94 – Sinal FM gerado pelo Raspberry Pi (no domínio do tempo), cujo sinal de mensagem é um sinal de voz. ....             | 133 |
| Figura 95 – Espectro do sinal FM gerado pelo Raspberry Pi (no domínio do tempo), cujo sinal de mensagem é um sinal de voz. .... | 134 |
| Figura 96 – Modelo do Simulink que transforma o Raspberry Pi num modulador em banda base 2-PAM com pulso retangular. ....       | 136 |
| Figura 97 – Janela de configurações do bloco <i>Bernoulli Binary Generator</i> . ....   | 136 |
| Figura 98 – Janela de configurações do bloco <i>M-PAM Baseband Modulator</i> . ....   | 137 |
| Figura 99 – Diagrama de constelação da modulação 2-PAM. ....  | 138 |
| Figura 100 – Janela de configurações do bloco <i>Complex to Real-Imag.</i> .....  | 138 |
| Figura 101 – Janela de configurações do bloco <i>FIR Interpolation</i> . ....   | 139 |
| Figura 102 – Componentes internos do subsistema AGC. ....   | 140 |
| Figura 103 – Sinal 2-PAM em banda base, com pulso de transmissão retangular NRZ, gerado pelo Raspberry Pi. ....                 | 140 |

|  |     |
|--|-----|
| Figura 104 – Espectro do sinal 2-PAM em banda base, com pulso de transmissão retangular NRZ, gerado pelo Raspberry Pi. ....                | 141 |
| Figura 105 – Sinal 2-PAM em banda base, com pulso de transmissão retangular RZ, gerado pelo Raspberry Pi. ....                             | 142 |
| Figura 106 – Espectro do sinal 2-PAM em banda base, com pulso de transmissão retangular RZ, gerado pelo Raspberry Pi. ....                 | 143 |
| Figura 107 – Sinal 2-PAM em banda base, com pulso de transmissão retangular do tipo Manchester, gerado pelo Raspberry Pi.....              | 145 |
| Figura 108 – Espectro do sinal 2-PAM em banda base, com pulso de transmissão retangular do tipo Manchester, gerado pelo Raspberry Pi. .... | 146 |
| Figura 109 – Diagrama de constelação da modulação 4-PAM. ....  | 147 |
| Figura 110 – Sinal 4-PAM em banda base, com pulso de transmissão retangular NRZ, gerado pelo Raspberry Pi. ....                            | 148 |
| Figura 111 – Espectro do sinal 4-PAM em banda base, com pulso de transmissão retangular NRZ, gerado pelo Raspberry Pi. ....                | 149 |
| Figura 112 – Modelo do Simulink que transforma o Raspberry Pi num modulador 2-PAM em banda base com pulso cosseno levantado.....           | 150 |
| Figura 113 – Janela de configurações do bloco <i>Raised Cosine Transmit Filter</i> .....   | 151 |
| Figura 114 – Sinal 2-PAM em banda base, com pulso de transmissão do tipo cosseno levantado, gerado pelo Raspberry Pi. ....                 | 152 |
| Figura 115 – Espectro do sinal 2-PAM em banda base, com pulso de transmissão do tipo cosseno levantado, gerado pelo Raspberry Pi. ....     | 153 |
| Figura 116 – Sinal 4-PAM em banda base, com pulso de transmissão do tipo cosseno levantado, gerado pelo Raspberry Pi. ....                 | 154 |
| Figura 117 – Espectro do sinal 4-PAM em banda base, com pulso de transmissão do tipo cosseno levantado, gerado pelo Raspberry Pi. ....     | 155 |
| Figura 118 – Modelo do Simulink que transforma o Raspberry Pi num modulador OOK.....   | 157 |
| Figura 119 – Sinal modulado OOK gerado pelo Raspberry Pi. ....   | 158 |
| Figura 120 – Espectro do sinal OOK gerado pelo Raspberry Pi. ....  | 159 |
| Figura 121 – Modelo do Simulink utilizado para transformar o Raspberry Pi num modulador 2-ASK. ....  | 160 |
| Figura 122 – Sinal 2-ASK com pulso de transmissão retangular gerado pelo Raspberry Pi. ....  | 161 |
| Figura 123 – Sinal 2-ASK com pulso de transmissão do tipo cosseno levantado gerado pelo Raspberry Pi.....                                  | 162 |
| Figura 124 – Espectro do sinal 2-ASK com pulso de transmissão do tipo cosseno levantado gerado pelo Raspberry Pi. ....                     | 163 |
| Figura 125 – Sinal 4-ASK gerado pelo Raspberry Pi.....   | 165 |

Figura 126 – Sinal 4-ASK com pulso de transmissão do tipo cosseno levantado gerado pelo Raspberry Pi..... 166

Figura 127 – Modelo do Simulink utilizado para transformar o Raspberry Pi num modulador BPSK. .... 168

Figura 128 – Janela de configurações do bloco *BPSK Modulator Baseband*..... 168

Figura 129 – Diagrama de constelação da modulação BPSK. .... 169

Figura 130 – Sinal BPSK com pulso de transmissão retangular gerado pelo Raspberry Pi..... 170

Figura 131 – Espectro do sinal BPSK com pulso de transmissão retangular gerado pelo Raspberry Pi..... 171

Figura 132 – Sinal BPSK com pulso de transmissão cosseno levantado gerado pelo Raspberry Pi..... 172

Figura 133 – Espectro do sinal BPSK com pulso de transmissão cosseno levantado gerado pelo Raspberry Pi..... 173

Figura 134 – Modelo do Simulink utilizado para transformar o Raspberry Pi num modulador BFSK. .... 174

Figura 135 – Componentes internos do subsistema “FSK”, utilizado no modelo do Simulink que transforma o Raspberry Pi num modulador BFSK. .... 175

Figura 136 – Janela de configurações do bloco *Saturation*. .... 176

Figura 137 – Sinal BFSK gerado pelo Raspberry Pi..... 177

Figura 138 – Espectro do sinal BFSK gerado pelo Raspberry Pi (1).. 178

Figura 139 – Espectro do sinal BFSK gerado pelo Raspberry Pi (2).. 179

Figura 140 – Modelo do Simulink utilizado para transformar o Raspberry Pi num modulador 4-FSK..... 180

Figura 141 – Componentes internos do subsistema 4-FSK que pode ser observado na Figura 140..... 181

Figura 142 – Sinal 4-FSK gerado pelo Raspberry Pi..... 182

Figura 143 – Espectro do sinal 4-FSK gerado pelo Raspberry Pi (1). 183

Figura 144 – Espectro do sinal 4-FSK gerado pelo Raspberry Pi (2). 183

Figura 145 – Modelo do Simulink que transforma o Raspberry Pi num modulador QPSK..... 186

Figura 146 – Janela de configurações do bloco *QPSK Modulator Baseband*..... 187

Figura 147 – Constelação QPSK adotada no experimento. .... 187

Figura 148 – Componentes internos do subsistema “BB to IF”, presente no modelo do Simulink que transforma o Raspberry Pi num modulador QPSK. .... 188

Figura 149 – Sinal QPSK gerado pelo Raspberry Pi (com o uso de pulsos de transmissão retangulares NRZ)..... 190



|  |     |
|--|-----|
| Figura 150 – Espectro do sinal QPSK gerado pelo Raspberry Pi (com uso de pulsos de transmissão retangulares NRZ). .....                                | 191 |
| Figura 151 – Sinal QPSK gerado pelo Raspberry Pi (com o uso de pulsos de transmissão do tipo Cosseno Levantado). .....                                 | 192 |
| Figura 152 – Espectro do sinal QPSK gerado pelo Raspberry Pi (com uso de pulsos de transmissão do tipo Cosseno Levantado). .....                       | 193 |
| Figura 153 – Bloco <i>M-PSK Modulator Baseband</i> , utilizado no modelo do Simulink que transforma o Raspberry Pi num modulador M-PSK. ....           | 194 |
| Figura 154 – Janela de configurações do bloco <i>M-PSK Modulator Baseband</i> . .....  | 195 |
| Figura 155 – Constelação 8-PSK adotada no experimento. ....  | 195 |
| Figura 156 – Sinal 8-PSK gerado pelo Raspberry Pi, com o uso do pulso de transmissão retangular NRZ. ....  | 197 |
| Figura 157 – Sinal 8-PSK gerado pelo Raspberry Pi, com o uso do pulso de transmissão retangular NRZ. ....  | 198 |
| Figura 158 – Bloco <i>Rectangular QAM Modulator Baseband</i> , utilizado no modelo do Simulink que transforma o Raspberry Pi num modulador M-QAM. .... | 200 |
| Figura 159 – Janela de configurações do bloco <i>M-QAM Modulator Baseband</i> . .....  | 200 |
| Figura 160 – Constelação de símbolos 16-QAM utilizada no experimento. ....   | 201 |
| Figura 161 – Sinal 16-QAM com pulso de transmissão cosseno levantado gerado pelo Raspberry Pi. ....  | 202 |
| Figura 162 – Obtenção do pacote de suporte ao hardware no MATLAB. ....   | 210 |
| Figura 163 – Selecionar uma ação. ....   | 210 |
| Figura 164 – Seleção do pacote de suporte ao hardware do Raspberry Pi. ....  | 211 |
| Figura 165 – Confirmação da instalação do pacote de suporte ao hardware. ....  | 211 |
| Figura 166 – Requisição de log in na conta registrada da Mathworks. ....   | 212 |
| Figura 167 – Inserção de informações para log in na conta da Mathworks. ....   | 212 |
| Figura 168 – Licença para utilização do pacote de suporte ao hardware. ....  | 213 |
| Figura 169 – Licenças para utilização do software de terceiros. ....   | 213 |
| Figura 170 – Confirmação da instalação do pacote de suporte ao hardware. ....  | 214 |
| Figura 171 – Instalação do pacote de suporte ao hardware finalizada. ....  | 215 |

|   |     |
|---|-----|
| Figura 172 – Seleção do pacote de suporte ao hardware a ser configurado. ....   | 216 |
| Figura 173 – Seleção do modelo do Raspberry Pi a ser utilizado. ....  | 216 |
| Figura 174 – Configuração da rede para comunicação com o Raspberry Pi. ....   | 217 |
| Figura 175 – Inserção de um cartão micro SD no computador. ....   | 218 |
| Figura 176 – Passos a serem realizados para a conexão do Raspberry Pi. ....   | 219 |
| Figura 177 – Esquema utilizado para estabelecer a conexão entre o Raspberry Pi e o computador. ....   | 220 |
| Figura 178 – Configurações de rede no Windows. ....   | 221 |
| Figura 179 – Rede Ethernet estabelecida entre o computador e o Raspberry Pi. ....   | 222 |
| Figura 180 – Propriedades da rede Ethernet. ....  | 222 |
| Figura 181 – Propriedades do protocolo de IP Versão 4 (TCP/IPv4). ....  | 223 |
| Figura 182 – Configuração da rede e detecção do Raspberry Pi. ....  | 224 |
| Figura 183 – Teste da conexão com o Raspberry Pi. ....  | 225 |
| Figura 184 – Erro ao testar conexão com o Raspberry Pi. ....  | 225 |
| Figura 185 – Mensagem indicando sucesso na conexão com o Raspberry Pi. ....   | 226 |
| Figura 186 – Mensagem indicando o fim dos procedimentos de configuração. ....   | 226 |
| Figura 187 – Teste da conexão entre o Raspberry Pi e o MATLAB na Command Window. ....   | 227 |
| Figura 188 – Comandos utilizados para invocar o Shell do Linux na Command Window. ....  | 227 |
| Figura 189 – Shell do Linux invocado na Command Window. ....  | 227 |
| Figura 190 – Utilização do comando “aplay -l” no Shell do Linux para checar a presença da placa de som USB externa como dispositivo de saída. ....            | 228 |
| Figura 191 – Utilização do comando “arecord -l” no Shell do Linux para checar a presença da placa de som USB externa como dispositivo de entrada. ....        | 229 |
| Figura 192 – Utilização do comando “cat /proc/asound/modules” para verificar a ordem de prioridade dos dispositivos de áudio conectados ao Raspberry Pi. .... | 230 |
| Figura 193 – Utilização do comando “sudo nano /etc/modprobe.d/alsa-base.conf” no Shell do Linux para a criação de um arquivo de configuração. ....            | 231 |
| Figura 194 – Trecho de código a ser inserido no arquivo de configuração. ....   | 231 |

|   |     |
|---|-----|
| Figura 195 – Linha de código “load card-specific configuration files (on request)”, presente no arquivo “/usr/share/alsa/alsa.conf”. .....        | 232 |
| Figura 196 – Utilização do comando “aplay -l” no Shell do Linux para checar a ordem de prioridade dos dispositivos de reprodução de áudio. ....   | 233 |
| Figura 197 – Utilização do Alsamixer, invocado no Shell do Linux, para definir os volumes da entrada e da saída da placa de som USB externa. .... | 234 |



## LISTA DE QUADROS

|   |     |
|---|-----|
| Quadro 1 – Mapeamento de bits em símbolos e pulsos transmitidos para a modulação 4-FSK..... | 182 |
| Quadro 2 – Mapeamento de bits em símbolos para a constelação QPSK.....                      | 188 |
| Quadro 3 – Mapeamento de bits em símbolos para a modulação 8-PSK.....                       | 196 |
| Quadro 4 – Parâmetros utilizados para a configuração da rede. ....                          | 218 |



## LISTA DE ABREVIATURAS E SIGLAS

- ADC – *Analogic to Digital Converter* (Conversor Analógico-Digital)  
ASK – *Amplitude Shift Keying* (Chaveamento por Deslocamento de Amplitude)  
BB – *Baseband* (Banda Base)  
BER – *Bit Error Rate* (Taxa de Erro de Bits)  
BLE – *Bluetooth Low Energy* (Bluetooth de Baixo Consumo)  
CSI – *Camera Serial Interface* (Interface Serial para Câmera)  
DAC – *Digital to Analogic Converter* (Conversor Digital Analógico)  
DDC – *Direct Digital Downconversion*  
DSI – *Display Serial Interface* (Interface Serial para Display)  
DSP – *Digital Signal Processor* (Processador Digital de Sinais)  
ECG – Eletrocardiograma  
FIR – *Finite Impulse Response* (Resposta ao Impulso Infinita)  
FSK – *Frequency Shift Keying* (Chaveamento por Deslocamento de Frequência)  
GPIO – *General Purpose Input and Output* (Entradas e saídas de propósito geral)  
IF – *Intermediate Frequency* (Frequência Intermediária)  
IIR – *Infinite Impulse Response* (Resposta ao Impulso Infinita)  
LO – *Local Oscillator* (Oscilador Local)  
MBD – *Model Based Design* (Projeto Baseado em Modelo)  
PAM – *Pulse Amplitude Modulation* (Modulação por Amplitude Pulso)  
PC – *Personal Computer* (Computador Pessoal)  
PID – *Process Identification* (Identificação do Processo)  
PSK – *Phase Shift Keying* (Chaveamento por Deslocamento de Fase)  
SDR – *Software Defined Radio* (Rádio Definido por Software)





## SUMÁRIO

|              |  |           |
|--------------|--|-----------|
| <b>1</b>     | <b>INTRODUÇÃO .....</b>  | <b>37</b> |
| 1.1          | OBJETIVOS .....  | 39        |
| <b>1.1.1</b> | <b>Objetivo geral .....</b>  | <b>39</b> |
| <b>1.1.2</b> | <b>Objetivos específicos .....</b>                                 | <b>39</b> |
| <b>2</b>     | <b>REVISÃO DO ESTADO DA ARTE .....</b>                             | <b>41</b> |
| 2.1          | RÁDIO DEFINIDO POR SOFTWARE .....                                  | 41        |
| <b>2.1.1</b> | <b>Arquitetura ideal de um Rádio Definido por Software.....</b>    | <b>41</b> |
| <b>2.1.2</b> | <b>Evolução das Arquiteturas de Rádio Definido por Software 42</b> |           |
| 2.2          | RASPBERRY PI.....  | 45        |
| 2.3          | PROJETO SIMULINK DEFINED RADIO .....                               | 47        |
| <b>3</b>     | <b>PLATAFORMA DIDÁTICA BASEADA NO RASPBERRY PI .....</b>           | <b>51</b> |
| 3.1          | RECURSOS NECESSÁRIOS .....   | 51        |
| <b>3.1.1</b> | <b>Recursos de Hardware.....</b>                                   | <b>51</b> |
| 3.1.1.1      | Computador Pessoal.....  | 52        |
| 3.1.1.2      | Cabo micro-USB – USB .....   | 52        |
| 3.1.1.3      | Cartão de memória micro SD.....                                    | 52        |
| 3.1.1.4      | Cabo Ethernet e Adaptador USB – Ethernet.....                      | 53        |
| 3.1.1.5      | Placa de som USB externa .....                                     | 53        |
| 3.1.1.6      | Cabos e Adaptadores.....   | 53        |
| 3.1.1.7      | Gerador de Funções .....   | 53        |
| 3.1.1.8      | Osciloscópio/Analisador de Espectro.....                           | 54        |
| <b>3.1.2</b> | <b>Recursos de Software.....</b>                                   | <b>54</b> |
| 3.2          | PROCEDIMENTOS PRELIMINARES .....                                   | 55        |
| <b>4</b>     | <b>EXPERIMENTOS DIDÁTICOS COM O RASPBERRY PI 57</b>                |           |
| 4.1          | EXPERIMENTO INTRODUTÓRIO .....                                     | 57        |
| <b>4.1.1</b> | <b>Bloco ALSA Audio Capture.....</b>                               | <b>57</b> |

|       |  |            |
|-------|--|------------|
| 4.1.2 | <b>Bloco ALSA Audio Playback .....</b>                                 | <b>59</b>  |
| 4.1.3 | <b>Configurações necessárias no Simulink .....</b>                     | <b>60</b>  |
| 4.1.4 | <b>Execução do experimento.....</b>                                    | <b>63</b>  |
| 4.2   | <b>RASPBERRY PI COMO UM GERADOR DE FUNÇÕES.</b>                        | <b>72</b>  |
| 4.2.1 | <b>Bloco Sine Wave.....</b>  | <b>73</b>  |
| 4.2.2 | <b>Bloco Data Type Conversion.....</b>                                 | <b>74</b>  |
| 4.2.3 | <b>Bloco Matrix Concatenate.....</b>                                   | <b>75</b>  |
| 4.2.4 | <b>Geração do sinal senoidal.....</b>                                  | <b>76</b>  |
| 4.2.5 | <b>Geração de uma forma de onda triangular com o Raspberry Pi.....</b> | <b>78</b>  |
| 4.2.6 | <b>Geração de uma forma de onda quadrada com o Raspberry Pi.....</b>   | <b>81</b>  |
| 4.2.7 | <b>Geração de outras formas de onda com o Raspberry Pi.</b>            | <b>85</b>  |
| 4.2.8 | <b>Geração de sinais a partir de arquivos .wav.....</b>                | <b>89</b>  |
| 4.3   | <b>RASPBERRY PI COMO UM FILTRO DIGITAL .....</b>                       | <b>93</b>  |
| 4.3.1 | <b>Raspberry Pi como um filtro digital do tipo FIR.....</b>            | <b>93</b>  |
| 4.3.2 | <b>Raspberry Pi como um filtro digital do tipo IIR.....</b>            | <b>96</b>  |
| 4.3.3 | <b>Raspberry Pi como qualquer tipo de filtro digital .....</b>         | <b>98</b>  |
| 4.3.4 | <b>Sugestões de experimentos práticos .....</b>                        | <b>101</b> |
| 4.3.5 | <b>Procedimentos “Deploy to Hardware” .....</b>                        | <b>70</b>  |
| 4.4   | <b>RASPBERRY PI COMO UM QUANTIZADOR .....</b>                          | <b>104</b> |
| 4.4.1 | <b>Quantização de um sinal senoidal com o Raspberry Pi.</b>            | <b>104</b> |
| 4.4.2 | <b>Reprodução de sinais de áudio com o Raspberry Pi.....</b>           | <b>110</b> |
| 4.4.3 | <b>Quantização de sinais de áudio com o Raspberry Pi.....</b>          | <b>113</b> |
| 4.5   | <b>MODULAÇÕES EM AMPLITUDE COM O RASPBERRY PI</b>                      | <b>116</b> |
| 4.5.1 | <b>Raspberry Pi como um modulador DSB-SC .....</b>                     | <b>116</b> |
| 4.5.2 | <b>Raspberry Pi como um modulador AM.....</b>                          | <b>122</b> |
| 4.6   | <b>MODULAÇÃO EM FREQUÊNCIA COM O RASPBERRY PI</b>                      | <b>128</b> |

|          |   |     |
|----------|---|-----|
| 4.7      | MODULAÇÕES EM BANDA BASE COM O RASPBERRY PI             | 135 |
| 4.7.1    | Modulação 2-PAM com pulso de transmissão retangular NRZ | 135 |
| 4.7.2    | Modulação 2-PAM com pulso de transmissão retangular RZ  | 141 |
| 4.7.3    | Modulação 2-PAM com pulso de transmissão Manchester     | 144 |
| 4.7.4    | Modulação 4-PAM com pulso de transmissão retangular     | 146 |
| 4.7.5    | Modulação 2-PAM com pulso Cosseno Levantado .....       | 149 |
| 4.7.6    | Modulação 4-PAM com pulso Cosseno Levantado .....       | 153 |
| 4.8      | MODULAÇÃO ASK COM O RASPBERRY PI .....                  | 155 |
| 4.8.1    | Geração de um sinal OOK com o Raspberry Pi.....         | 156 |
| 4.8.2    | Geração de um sinal 2-ASK com o Raspberry Pi.....       | 159 |
| 4.8.3    | Geração de um sinal 4-ASK com o Raspberry Pi.....       | 163 |
| 4.9      | MODULAÇÕES BPSK, BFSK e 4-FSK COM O RASPBERRY PI.....   | 166 |
| 4.9.1    | Geração de um sinal BPSK com o Raspberry Pi.....        | 166 |
| 4.9.2    | Geração de um sinal BFSK com o Raspberry Pi.....        | 173 |
| 4.9.3    | Geração de um sinal 4-FSK com o Raspberry Pi.....       | 179 |
| 4.10     | MODULAÇÕES PSK E QAM COM O RASPBERRY PI                 | 184 |
| 4.10.1   | Modulações QPSK e 8-PSK com o Raspberry Pi.....         | 184 |
| 4.10.1.1 | Geração de um sinal QPSK com o Raspberry Pi .....       | 185 |
| 4.10.1.2 | Geração de um sinal 8-PSK com o Raspberry Pi.....       | 194 |
| 4.10.2   | Geração de um sinal 16-QAM com o Raspberry Pi.....      | 198 |
| 4.10.3   |   | 202 |
| 5        | CONCLUSÃO .....   | 203 |
|          | REFERÊNCIAS .....                                       | 207 |
|          | APÊNDICE A .....  | 209 |

|     |   |     |
|-----|---|-----|
| 5.1 | INSTALAÇÃO DO PACOTE DE SUPORTE AO HARDWARE ..... | 209 |
| 5.2 | INSTALAÇÃO DO RASPBIAN E CONFIGURAÇÃO DA REDE     | 215 |
| 5.3 | CONFIGURAÇÃO DO ADAPTADOR USB PARA ETHERNET ..... | 220 |
| 5.4 | INICIALIZAÇÃO DO RASPBERRY PI.....                | 224 |
| 5.5 | CONFIGURAÇÃO DA PLACA DE SOM USB EXTERNA          | 228 |

## 1 INTRODUÇÃO

A principal motivação para a elaboração deste trabalho é o artigo de Bazzi, Pasolini e Zabini (2017). Segundo esses autores, os Rádios Definidos por Software (ou SDRs, do inglês *Software Defined Radios*) estão se tornando a tecnologia dominante nos sistemas de comunicações, porém o ensino de SDR em cursos de graduação ainda enfrenta alguns desafios.

O projeto de um SDR requer um amplo conhecimento em diferentes tópicos como arquitetura de sistemas de comunicações, processamento digital de sinais, linguagens de programação e outros aspectos específicos de hardware (BAZZI, PASOLINI e ZABINI, 2017). Ainda de acordo com os autores, fornecer aos estudantes uma formação sólida em SDR é uma tarefa desafiadora, e o ensino efetivo desse tema não pode separar os fundamentos teóricos dos experimentos práticos com dispositivos e instrumentos reais.

Bazzi, Pasolini e Zabini (2017) também acrescentam que experimentos práticos de SDR raramente são oferecidos em cursos de graduação por dois motivos principais: o custo elevado do hardware necessário, como DSPs (*Digital Signal Processors*) ou kits FPGA (*Field Programmable Gate Array*); e a complexidade e exigência de tempo desses experimentos. Devido ao segundo motivo, professores de processamento de sinais e comunicações não se aprofundam na sintaxe específica dos códigos utilizados para implementar um dado subsistema ou algoritmo. Eles preferem focar em metodologias de projeto, algoritmos de processamento de sinais e análise de sinais nos domínios do tempo e da frequência. Nesse contexto, atividades práticas de SDR raramente são oferecidas aos estudantes.

As duas dificuldades apresentadas anteriormente podem ser superadas graças a recente introdução no mercado de dispositivos programáveis e de propósito geral com duas características principais: custo relativamente baixo e a possibilidade de fazer esses dispositivos funcionarem com software gerado automaticamente a partir de um modelo funcional (como um diagrama de blocos) do sistema a ser implementado (BAZZI, PASOLINI e ZABINI, 2017).

A placa Raspberry Pi é provavelmente o exemplo mais relevante desses novos dispositivos. As principais características dessa placa serão apresentadas na seção Revisão do Estado da Arte. A MathWorks (2018), empresa desenvolvedora dos softwares MATLAB & Simulink, fornece gratuitamente um pacote de suporte ao hardware do Raspberry Pi. Esse

pacote é um software *Add On* utilizado especificamente na conexão entre a placa Raspberry Pi e os softwares MATLAB & Simulink.

O Simulink permite a modelagem gráfica e simulação de um sistema a ser implementado, e a tradução do modelo (diagrama de blocos) em software, que pode ser então carregado e executado no dispositivo graças à geração automática de código provida pelo pacote de suporte ao hardware. Isso permite que estudantes possam projetar e implementar algoritmos complexos sem a necessidade de conhecerem alguma linguagem de programação específica (BAZZI, PASOLINI e ZABINI, 2017).

Uma plataforma didática de baixo custo e baseada no Raspberry Pi para ensino de processamento digital de sinais e comunicações é proposta no artigo de Bazzi, Pasolini e Zabini (2017). Conforme é colocado no artigo, nessa plataforma didática o Raspberry Pi é utilizado de maneira não convencional e inovadora. Dispositivos periféricos, como monitor, teclado e mouse não são empregados. Com o uso de cabos e conectores adequados, o Raspberry Pi é utilizado como um DSP de baixo custo que executa algoritmos de processamento digital de sinais e sistemas de comunicação baseados em SDRs. Mais detalhes sobre essa plataforma didática serão apresentados na seção Recursos Necessários.

Este Trabalho de Conclusão de Curso consiste em inicialmente replicar a plataforma didática baseada no uso do Raspberry Pi em conjunto com MATLAB & Simulink proposta por Bazzi, Pasolini e Zabini (2017), e em seguida desenvolver um conjunto de experimentos práticos que possa ser utilizado em aulas de laboratório de disciplinas introdutórias da área de Comunicações. Alguns dos experimentos a serem desenvolvidos serão adaptações dos experimentos do Projeto *Simulink Defined Radio*, mantido por Pasolini, Bazzi e Mirabella (2018). Porém, outros experimentos sobre temas não abordados pelo projeto *Simulink Defined Radio* também serão desenvolvidos.

Para uma plena compreensão deste trabalho, recomenda-se que o leitor tenha conhecimentos prévios sobre sistemas de comunicações e processamento digital de sinais.

Nas disciplinas de Princípios de Sistemas de Comunicações, Fundamentos de Comunicação Digital e Processamento Digital de Sinais oferecidas aos cursos de Engenharia Elétrica e Engenharia Eletrônica da Universidade Federal de Santa Catarina, as atuais aulas de laboratório consistem na realização de simulações utilizando MATLAB & Simulink. Espera-se que o conjunto de experimentos práticos desenvolvido para este trabalho possa ser utilizado futuramente nessas

aulas de laboratório de modo a fortalecer o aprendizado e a motivação dos alunos, que poderão ter um vislumbre maior da ligação entre teoria e prática ao poder visualizar e mensurar sinais utilizando instrumentos reais.

## 1.1 OBJETIVOS

### 1.1.1 Objetivo geral

Elaborar um conjunto de experimentos práticos a ser utilizado em aulas de laboratório de disciplinas de Sinais e Sistemas, Processamento Digital de Sinais, Comunicações e correlatas utilizando uma plataforma didática de baixo custo baseada no Raspberry Pi.

### 1.1.2 Objetivos específicos

- Reproduzir a plataforma didática baseada no Raspberry Pi proposta por Bazzi, Pasolini e Zabini (2017).
- Reproduzir os experimentos didáticos propostos por Pasolini, Bazzi e Mirabella (2016), que utilizam a plataforma baseada em Raspberry Pi e os softwares MATLAB & Simulink.
- Adaptar os experimentos propostos por Pasolini, Bazzi e Mirabella (2016) e elaborar novos experimentos, para futuramente serem utilizados em aulas de laboratório de disciplinas oferecidas na UFSC.
- Utilizar a geração de código automática do pacote de suporte ao hardware do Raspberry Pi para implementar algoritmos de processamento digital de sinais e sistemas de comunicações sem o uso de linguagens de programação específicas.
- Elaborar experimentos que permitam aos estudantes observar e mensurar sinais nos domínios do tempo e da frequência com o uso de instrumentos reais.
- Aprimorar o método de ensino das aulas de laboratório de disciplinas de Comunicações, fortalecendo a ligação entre teoria e prática, e assim contribuir com o aprendizado e a motivação dos estudantes.





## 2 REVISÃO DO ESTADO DA ARTE

### 2.1 RÁDIO DEFINIDO POR SOFTWARE

Segundo Mitola (1995) *apud* Stewart *et al* (2015), o termo Rádio Definido por Software (em inglês *Software Defined Radio*) refere-se a sistemas de rádio nos quais a maioria das funções associadas à camada física são implementadas em software através do uso de algoritmos de processamento digital de sinais. Tuttlebee (2002) *apud* Stewart *et al* (2015) acrescenta que um receptor SDR ideal possuiria apenas um pequeno *front-end* analógico, composto apenas por uma antena e um amostrador de alta taxa de amostragem (na ordem de GHz) capaz de capturar e digitalizar uma banda muito ampla de sinais de rádio. As funções de demodulação, sincronização, decodificação ou descryptografia seriam realizadas em software com o uso de um DSP dedicado de alto desempenho.

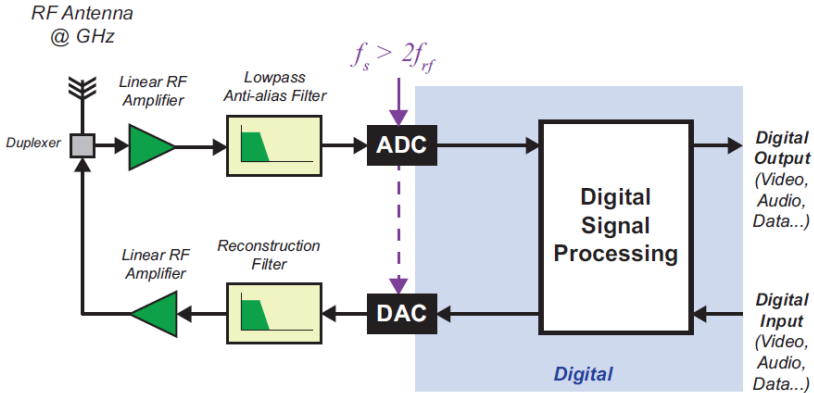
Goeller e Tate (2014) *apud* Stewart *et al* (2015) citam que nos últimos anos os SDRs têm se apresentado e sido anunciados como a solução futura para todos os receptores de RF. Stewart *et al* (2015) ainda acrescentam que é correto dizer que os SDRs têm se mostrado uma solução promissora há alguns anos, mas somente recentemente (por volta de 2014) se tornou uma solução de baixo custo e largamente disponível. Haghghat (2002) *apud* Stewart *et al* (2015) finaliza dizendo que no passado os SDRs estavam associados somente a aplicações de pesquisa e fins militares, devido ao seu (historicamente) alto custo de implementação.

#### 2.1.1 Arquitetura ideal de um Rádio Definido por Software

Stewart *et al* (2015) apresenta a arquitetura ideal de um SDR, que pode ser observada na Figura 1. Conforme ilustrado na Figura, essa arquitetura é composta basicamente por uma seção de RF (constituída por uma antena, amplificadores e filtros); um conversor analógico-digital (*Analog-to-Digital Converter*, ou simplesmente ADC) e um conversor digital-analógico (*Digital-to-Analog Converter*, ou simplesmente DAC), ambos de alto desempenho; e um computador ou DSP de alto desempenho. Do teorema de Nyquist (e supondo amostragem e reconstrução ideal), se a amostragem for realizada a uma frequência  $f_s = 4$  GHz, o espectro em banda base resultante estaria entre 0 e 2 GHz (ou  $f_s/2$ ). Dessa forma, o SDR estaria apto a transmitir e

receber sinais em todas as frequências até 2 GHz, com a modulação e demodulação sendo realizadas no domínio digital.

Figura 1 – Componentes de uma arquitetura SDR simples e conceitual.



Fonte: Stewart *et al* (2015).

Conforme indicam Stewart *et al* (2015), ADCs e DACs que operam com taxas de amostragem de ordem de GHz serão disponíveis num futuro próximo, e assim a arquitetura ilustrada na Figura 1 será realizável. Os autores apontam que no final dos anos 1990, os melhores (e mais caros) ADCs e DACs operavam com frequências de amostragem da ordem de 100 kHz e resolução de 16 bits. Atualmente, existem ADCs e DACs de custos razoáveis que operam com frequências de amostragem da ordem de 100 MHz e resolução de 14 bits. Também já existem dispositivos com frequência de amostragem da ordem de GHz e resolução de 8 bits, porém de custo elevado e ainda não disponíveis comercialmente.

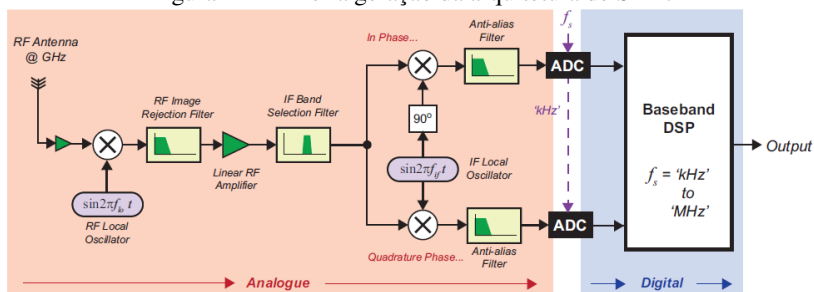
### 2.1.2 Evolução das Arquiteturas de Rádio Definido por Software

Stewart *et al* (2015) apresentam de forma muito interessante a evolução de um receptor em quadratura de rádio, na qual é possível observar o crescimento do papel desempenhado pelo processamento digital de sinais (o caminho para o Rádio Definido por Software). A seguir será apresentado um breve resumo dessa evolução.

A primeira geração de rádios digitais surgiu em meados dos anos 1990. Essa arquitetura inicial pode ser observada na Figura 2. A parte analógica do receptor realiza a conversão do sinal de RF para uma

frequência intermediária (IF, do inglês *intermediate frequency*) com o uso de um oscilador local (LO, do inglês *local oscillator*), e então um segundo LO realiza a conversão do sinal em IF para um sinal em banda base. O sinal em banda base é então amostrado por um ADC que opera com taxa de amostragem da ordem de dezenas de kHz, e por fim um DSP é utilizado para processar o sinal digitalizado e recuperar a informação transmitida.

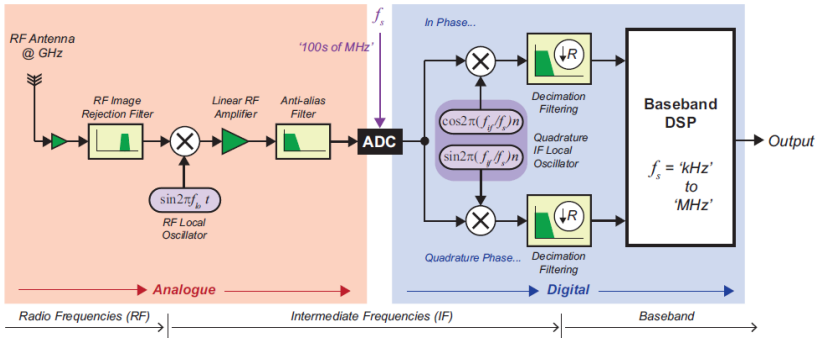
Figura 2 – Primeira geração da arquitetura de SDR.



Fonte: Stewart *et al* (2015).

Na segunda geração de rádios digitais, que surgiu no início dos anos 2000, a amostragem e digitalização dos sinais começaram a ser realizada em alguns dispositivos em IF. Por exemplo, uma frequência intermediária da ordem de 40 MHz poderia ser suportada por um ADC com frequência de amostragem de 125 MHz. O diagrama de blocos da segunda arquitetura de SDR pode ser observado na Figura 3. O primeiro estágio de DSP dessa arquitetura envolve a utilização de *Direct Digital Downconversion* (DDC) para transladar sinais de IF para banda base usando demodulação e filtragem por dizimação. Posteriores operações de DSP são realizadas com o sinal em banda base. Como apontam Stewart *et al* (2015), mais funcionalidades são implementadas no domínio digital, o que confere uma maior flexibilidade para o SDR.

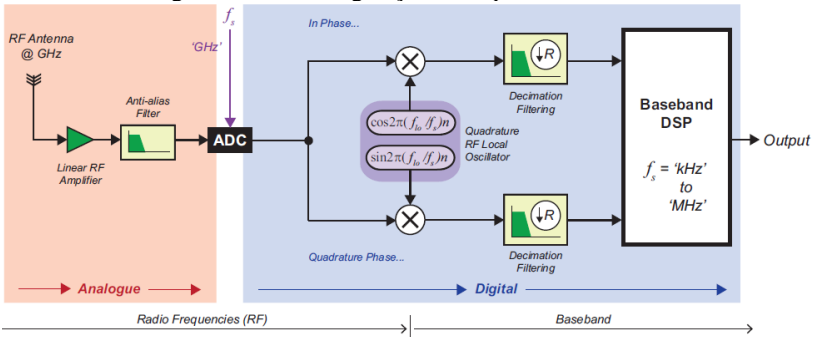
Figura 3 – Segunda geração da arquitetura de SDR.



Fonte: Stewart *et al* (2015).

Por fim, a terceira geração de rádio digitais, apontada por Stewart *et al* (2015) como o atual estado da arte de Rádio Definido por Software, envolve a amostragem direta de sinais de RF e sua conversão para banda base em apenas um único estágio com o uso de DSP, conforme pode ser observado no diagrama de blocos da Figura 4. Os autores acrescentam que essa arquitetura seria realizável hoje graças à disponibilidade de amostradores com frequências da ordem de GHz, que poderiam permitir a realização da arquitetura ideal de SDR ilustrada na Figura 1.

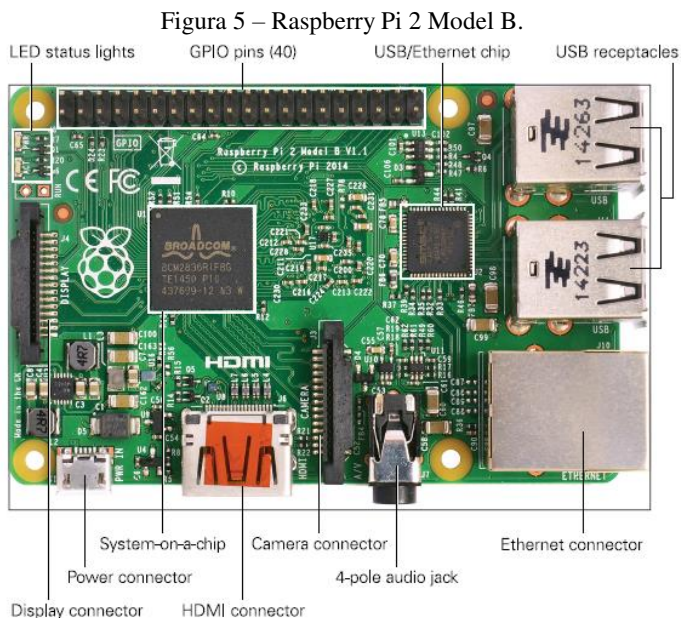
Figura 4 – Terceira geração da arquitetura de SDR.



Fonte: Stewart *et al* (2015).

## 2.2 RASPBERRY PI

O Raspberry Pi é um computador de baixo custo (seu preço é de cerca de £25) e de pequenas dimensões (aproximadamente do tamanho de um cartão de crédito) (UPTON, 2016). Na Figura 5 podemos observar a fotografia de um Raspberry Pi 2 Model B com as indicações de seus principais componentes.



Fonte: Upton *et al* (2016).

Conforme é apontado por Upton *et al* (2016), o Raspberry Pi tem maior poder computacional, mais memória e maior capacidade de armazenamento do que os primeiros computadores pessoais. Porém, não apresenta o desempenho, displays *high-end*, fontes de energia *built-in* e capacidade de disco rígido dos desktops e laptops atuais.

As limitações do Raspberry Pi podem ser superadas através da conexão entre o pequeno computador e dispositivos periféricos. Alguns exemplos de dispositivos que podem ser conectados são discos rígidos, displays HDMI e sistemas de som de alta qualidade (UPTON *et al*, 2016). Ainda na Figura 5 é possível observar um conector Ethernet e portas USB que permitem a conexão do Raspberry Pi com muitos outros dispositivos, como mouses, teclados e até mesmo outros computadores.

Upton *et al* (2016) dizem que o Raspberry Pi pode ser utilizado nos mais diversos projetos, sendo que na internet é possível encontrar a documentação de milhares deles. Alguns exemplos de aplicações citadas pelos autores são: automação residencial, segurança residencial, central multimídia, estação de monitoramento de clima, controlador de robô, controlador de drone, servidor web, servidor de email, controlador de *web cam* e computador automotivo embarcado.

A versão do computador a ser utilizada neste trabalho é a Raspberry Pi 3. Segundo a Raspberry Pi Foundation (2018a), a terceira geração substituiu o Raspberry Pi 2 Model B em fevereiro de 2016. Ainda de acordo com a organização, algumas das principais novidades da terceira geração em comparação com a segunda são:

- CPU ARMv8 quad-core 1.2 GHz 64 bits;
- 802.11n wireless LAN (*Local Area Network*);
- Bluetooth 4.1;
- Bluetooth Low Energy (BLE).

Também de acordo com a Raspberry Pi Foundation (2018a), os recursos que a terceira geração do Raspberry Pi tem em comum com a segunda são:

- 4 portas USB;
- 40 pinos GPIO (*general purpose input/output*);
- Porta full HDMI;
- Porta Ethernet;
- Jack de 3,5 mm combinado para áudio e vídeo;
- Interface para câmera (CSI, *câmera serial interface*);
- Interface para display (DSI, *display serial interface*);
- Slot para cartão micro SD;
- Núcleo para vídeo VideoCore IV 3D Graphics.

A Raspberry Pi Foundation (2018b) diz que o sistema operacional recomendado para o Raspberry Pi é o Raspian, uma versão do GNU/Linux especialmente projetada para funcionar bem no pequeno computador. O sistema operacional e arquivos são armazenados no cartão micro SD que deve ser conectado ao Raspberry Pi (RASPBERRY PI FOUNDATION, 2018c). Upton *et al* (2016) indicam que o tamanho mínimo usualmente recomendado para o cartão micro SD a ser utilizado é de 8 GB.

A Raspberry Pi Foundation (2018d) também informa que a alimentação do Raspberry Pi deve ser feita através de uma fonte microB USB com capacidade de fornecer até 2A de corrente.

Os pinos de GPIO permitem a interface entre o Raspberry Pi e o mundo real, visto que através desses pinos o pequeno computador pode ser conectado a diversos dispositivos periféricos (UPTON *et al*, 2016). Upton *et al* (2016) informam que alguns dos pinos podem ser configurados para serem entradas ou saídas digitais de propósito geral, enquanto que outros pinos tensão DC (de 5V ou 3,3 V) ou referência (0V).

A Sparkfun (2018) fornece um diagrama e uma tabela que explicam detalhadamente como cada pino de GPIO do Raspberry Pi pode ser utilizado. Além disso, de acordo com a Adafruit (2018), o Raspberry Pi não possui uma entrada analógica (o propósito da placa é controlar apenas entradas e saídas digitais). Assim, se for necessário realizar a leitura de um sinal analógico, é necessário utilizar um ADC externo. Esse fato justifica a utilização da placa de som USB externa neste trabalho.

## 2.3 PROJETO SIMULINK DEFINED RADIO

O Projeto Simulink Defined Radio consiste num conjunto de materiais didáticos (uma apostila e um conjunto de projetos desenvolvidos no Simulink) fornecidos gratuitamente por Pasolini, Bazzi e Mirabella (2018) em seu website. Segundo os autores do projeto, os experimentos práticos, que utilizam a plataforma didática baseada em Raspberry Pi e visam ser utilizados em cursos de telecomunicações, provêm uma forma viável e de baixo custo de introduzir estudantes, hobbistas e entusiastas de tecnologia ao paradigma de Rádio Definido por Software.

Os responsáveis pelo Projeto Simulink Defined Radio são os professores Gianni Pasolini, Alessandro Bazzi e Mirko Mirabella, que atuam na Universidade de Bolonha (Itália). Esse projeto nasceu no *Wireless Communication Laboratory* (WiLab), localizado na Itália. Esse laboratório mantém parcerias com as Universidades de Bolonha e de Ferrara e com o *National Research Council* (CNR).

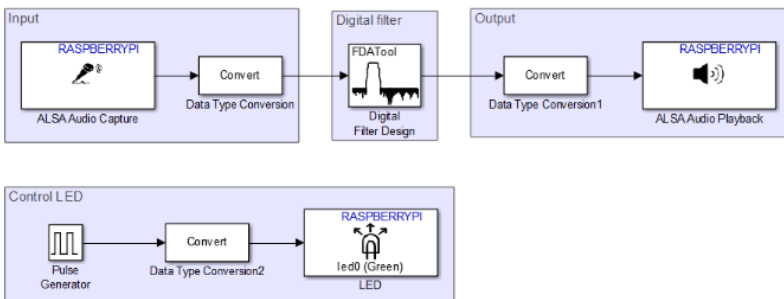
Pasolini, Bazzi e Mirabella (2018) afirmam que software para o Raspberry Pi pode ser gerado por usuários sem experiência em programação graças ao paradigma MBD (*Model Based Design*), que permite a modelagem e simulação de sistemas dinâmicos através de um ambiente gráfico. Ainda de acordo com esse paradigma, a fase de desenvolvimento de software consiste na escolha e na interconexão gráfica de blocos predefinidos que executam funções específicas, e cujos

comportamentos são controlados através de valores de parâmetros definidos pelo usuário.

Ainda de acordo com Pasolini, Bazzi e Mirabella (2018), o Simulink (desenvolvido pela Mathworks) é o mais famoso ambiente gráfico de MBD. Além disso, graças ao pacote de suporte ao hardware do Raspberry Pi desenvolvido pela Mathworks, placas Raspberry Pi são completamente suportadas pelo Simulink, que permite não apenas a modelagem e simulação, mas também a geração automática de código. Ao final da modelagem e simulação, o Simulink converte o modelo projetado em um arquivo executável que pode ser carregado na placa Raspberry Pi.

Como exemplo da aplicação do paradigma MBD, na Figura 6 podemos observar um modelo do Simulink que é utilizado para transformar o Raspberry Pi num DSP que realiza filtragem digital. O bloco *Alsa Audio Capture* é responsável pela leitura do sinal obtido após a digitalização do sinal analógico aplicado à entrada da placa de som externa conectada ao Raspberry. A filtragem do sinal é realizada utilizando-se o bloco *Digital Filter Design* do Simulink, que permite o projeto do filtro digital sem a necessidade do uso de linguagens de programação específicas. O bloco *Alsa Audio Playback* é responsável por enviar um sinal digital à placa de som, que por sua vez o converterá num sinal analógico e o apresentará à sua saída. Os dois blocos *Data Type Conversion* são responsáveis por realizarem as conversões necessárias dos tipos de dados das amostras. Os blocos *Pulse Generator*, *Data Type Conversion* e *LED* localizados na parte inferior da Figura 6 são responsáveis por fazer com que um LED do Raspberry Pi pisque durante seu funcionamento.

Figura 6 – Modelo do Simulink que transforma o Raspberry Pi num DSP que realiza filtragem digital.



Fonte: Pasolini, Bazzi e Mirabella (2018).



Pasolini, Bazzi e Mirabella (2018) apresentam em seu site a relação de todos os recursos de hardware e software necessários para a realização dos experimentos (a relação completa pode ser encontrada na seção Recursos Necessários deste trabalho). Os autores utilizam o Raspberry Pi 2 para a execução dos experimentos, porém Bazzi, Pasolini e Zabini (2017) afirmam que esses também podem ser realizados com o Raspberry Pi 3.

Os autores fornecem em seu website os arquivos necessários para a execução dos seguintes experimentos:

- Raspberry Pi 2 como um gerador de funções;
- Raspberry Pi 2 como um filtro digital;
- Modulações banda base com o Raspberry Pi 2;
- Modulações 2-ASK e 4-ASK com o Raspberry Pi 2;
- Modulação QPSK com o Raspberry Pi 2;
- Modulação 2-FSK com o Raspberry Pi 2;
- Raspberry Pi 2 como um modulador OFDM.

Uma das etapas do trabalho a ser executado corresponde à reprodução dos experimentos listados acima. Tendo esses como inspiração, outros experimentos sobre temas relacionados a disciplinas de Princípios de Sistemas de Comunicações e Fundamentos de Comunicação Digital (que não foram abordados pelo *Projeto Simulink Defined Radio*) são desenvolvidos.



### 3 PLATAFORMA DIDÁTICA BASEADA NO RASPBERRY PI

#### 3.1 RECURSOS NECESSÁRIOS

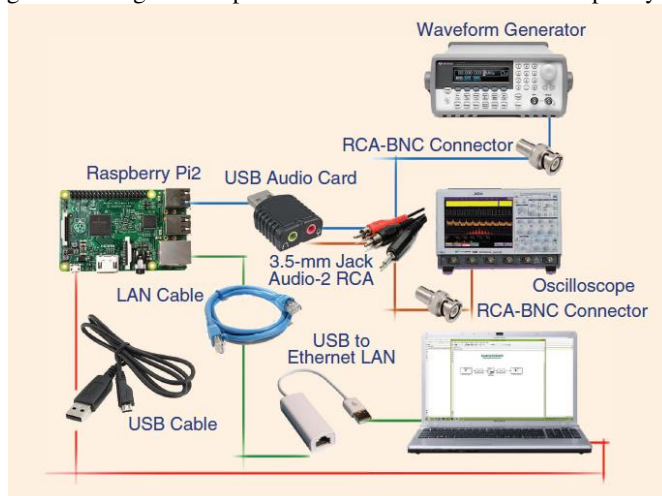
##### 3.1.1 Recursos de Hardware

Os recursos de hardware necessários para a reprodução da plataforma didática proposta por Bazzi, Pasolini e Zabini (2017) são:

- Um Raspberry Pi;
- Um computador pessoal (desktop ou notebook);
- Um cabo micro-USB – USB;
- Um cartão de memória micro SD de no mínimo 8 GB e seu adaptador SD.
- Um cabo Ethernet;
- Um adaptador USB – Ethernet;
- Uma placa de som externa USB;
- Dois cabos adaptadores P2 macho – RCA macho;
- Dois conectores adaptadores RCA fêmea – BNC macho;
- Um gerador de funções;
- Um osciloscópio/analizador de espectro.

O diagrama da plataforma didática baseada no Raspberry Pi pode ser observado na Figura 7.

Figura 7 – Diagrama da plataforma didática baseada no Raspberry Pi.



Fonte: Bazzi, Pasolini e Zabini (2017).

A plataforma didática proposta por Bazzi, Pasolini e Zabini (2017) utiliza um Raspberry Pi 2, conforme está ilustrado na Figura 7. Porém, os autores afirmam que a plataforma didática também pode ser replicada com o Raspberry Pi 3, que foi a placa adotada neste trabalho.

Conforme afirmam Pasolini, Bazzi e Mirabella (2016), alguns dos equipamentos empregados na plataforma (gerador de funções e osciloscópio) são caros demais para serem adquiridos por estudantes ou hobbistas. Assim, os experimentos didáticos propostos nesse trabalho não poderão ser reproduzidos em casa. Entretanto, geradores de funções e osciloscópios são equipamentos facilmente encontrados em laboratórios de eletrônica e telecomunicações de universidades.

A seguir serão apresentados alguns comentários importantes sobre os recursos de hardware necessários.

#### 3.1.1.1 Computador Pessoal

Para a execução dos experimentos, é necessário um computador com os softwares MATLAB e Simulink instalados. Para que esses softwares apresentem um desempenho adequado, recomenda-se a utilização de um computador com processador Intel Core i5 ou i7 (ou similares) e com memória RAM de no mínimo 8 GB.

#### 3.1.1.2 Cabo micro-USB – USB

O cabo micro-USB tem a função de conectar o Raspberry Pi a uma das entradas USB do computador. A única função dessa conexão é o fornecimento de alimentação DC de 5 V ao Raspberry Pi. Segundo Bazzi, Pasolini e Zabini (2017), uma fonte externa DC poderia ser utilizada para o mesmo propósito, porém essa fonte pode inserir interferências indesejadas no sinal de saída da placa.

#### 3.1.1.3 Cartão de memória micro SD

O cartão de memória micro SD tem a função de armazenar o Raspbian, o sistema operacional do Raspberry Pi, e demais arquivos necessários para a execução dos experimentos. Detalhes sobre os procedimentos de instalação do Raspbian serão fornecidos no Apêndice deste trabalho.

#### 3.1.1.4 Cabo Ethernet e Adaptador USB – Ethernet

O cabo Ethernet tem a função de realizar a transferência de dados entre o computador e o Raspberry Pi. Esse cabo é conectado ao computador através de um adaptador USB-Ethernet ou diretamente na entrada Ethernet se essa estiver disponível.

#### 3.1.1.5 Placa de som USB externa

Para o desenvolvimento dos experimentos didáticos propostos, tanto uma entrada analógica quanto uma saída analógica são necessárias. Ambas são obtidas através do uso da placa de som externa USB conectada ao Raspberry Pi. A entrada para microfone é utilizada como a entrada analógica, enquanto que a saída para fones de ouvido é utilizada como a saída analógica. Assim, a placa desempenha um papel duplo: conversor analógico-digital para o sinal de entrada e conversor digital-analógico para o sinal de saída (BAZZI, PASOLINI e ZABINI, 2017).

Como a placa de som externa foi originalmente concebida para sinais de áudio, sua taxa de amostragem máxima está limitada a 48000 amostras/s. Portanto, a banda dos sinais a serem operados pela plataforma proposta deve estar entre 0 e 24 kHz (a frequência da portadora dos sinais em banda passante deve ser da ordem de 15~20 kHz e a largura de banda da ordem de poucos kHz). Para fins didáticos, essa limitação não é um problema. Por exemplo, para o caso de uma transmissão digital, a limitação na largura de banda apenas implica que a taxa de bits deve ser mantida apropriadamente baixa, sem nenhuma modificação requerida na arquitetura do sistema (BAZZI, PASOLINI e ZABINI, 2017).

#### 3.1.1.6 Cabos e Adaptadores

Um dos cabos P2 macho – RCA macho e um dos adaptadores RCA fêmea – BNC macho são utilizados para conectar a saída do gerador de funções à entrada da placa de som externa, enquanto que o outro cabo e o outro adaptador são utilizados para conectar a saída da placa de som externa à entrada do osciloscópio.

#### 3.1.1.7 Gerador de Funções

Um gerador de funções básico tem a capacidade de gerar diferentes sinais periódicos (por exemplo, ondas senoidais, quadradas ou

triangulares) com características definidas pelo usuário (com amplitude e frequência). Em alguns dos experimentos a serem desenvolvidos com o Raspberry Pi, o gerador de funções será empregado para gerar sinais de entrada para os sistemas.

### 3.1.1.8 Osciloscópio/Analisador de Espectro

O osciloscópio será empregado em todos os experimentos para observar o comportamento de sinais no domínio do tempo. Alguns osciloscópios também possuem a função de analisador de espectro, isto é, permitem a análise do sinal no domínio da frequência. O osciloscópio utilizado neste trabalho possui essa função, que será utilizada em muitos dos experimentos.

### 3.1.2 Recursos de Software

De acordo com Bazzi, Pasolini e Zabini (2017), os únicos recursos de software necessários para o desenvolvimento da plataforma são os softwares MATLAB & Simulink com os seguintes pacotes instalados: *Signal Processing Toolbox*, *DSP System Toolbox*, *Communications System Toolbox* e *Raspberry Pi Support Package*.

Bazzi, Pasolini e Zabini (2017) acrescentam que os softwares listados acima compreendem todos os blocos necessários para modelar a camada física de um sistema de comunicações: geração de dados, codificação de canal, modulação, filtragem, demodulação, sincronização de portadora, sincronização de símbolo, etc. Esses softwares também possuem ferramentas para o projeto e análise de filtros FIR (*finite impulse response*) e IIR (*infinite impulse response*), instrumentos virtuais para a observação de diagramas de constelação, diagramas de olho e espectros e ferramentas para obter métricas de desempenho como BER (*bit error rate*) e VEM (*vector error magnitude*).

Graças ao pacote de suporte do Raspberry Pi, os modelos projetados no Simulink (diagramas de blocos) podem ser traduzidos em código de baixo nível que pode ser carregado na placa e executado. Após esses procedimentos, o Raspberry Pi pode ser desconectado do computador e conectado à instrumentos reais de medição (BAZZI, PASOLINI e ZABINI, 2017).

### 3.2 PROCEDIMENTOS PRELIMINARES

Após todos os recursos de hardware e software terem sido providenciados, é necessário executar uma série de procedimentos preliminares para garantir o correto funcionamento da plataforma didática. Esses procedimentos devem ser executados na seguinte ordem:

- 1) Instalação do pacote de suporte ao hardware do Raspberry Pi para MATLAB e Simulink.
- 2) Instalação do Raspbian (sistema operacional do Raspberry Pi) num cartão micro SD.
- 3) Configuração da rede Ethernet para comunicação entre o PC e o Raspberry Pi;
- 4) Configuração do adaptador USB para Ethernet;
- 5) Inicialização do Raspberry Pi (o que inclui a invocação do Shell do Raspbian através do MATLAB);
- 6) Configuração da placa de som USB externa para ser o dispositivo primário de áudio do Raspberry Pi.

No Apêndice A deste trabalho é possível encontrar um tutorial passo a passo e ilustrado de como realizar cada um dos procedimentos relacionados acima. A versão do MATLAB utilizada neste tutorial é a R2016a. Para outras versões do MATLAB, a ordem dos procedimentos ou as janelas de configurações apresentadas ao usuário podem ser diferentes. Porém, a execução de cada um desses procedimentos ainda deve ser semelhante, e cada um deles é indispensável.





## 4 EXPERIMENTOS DIDÁTICOS COM O RASPBERRY PI

### 4.1 EXPERIMENTO INTRODUTÓRIO

O primeiro experimento a ser realizado com o Raspberry Pi em conjunto com os softwares MATLAB e Simulink tem dois objetivos básicos: testar o funcionamento de todos os recursos de hardware e software sendo utilizados, e também ensinar aos estudantes os passos para compilar e executar no Raspberry Pi modelos gerados no Simulink. Os passos apresentados a seguir deverão ser executados também em todos os outros experimentos deste trabalho.

No Simulink, o modelo mostrado na Figura 8 deve ser implementado. Esse modelo tem a seguinte função: gerar na saída da placa de som USB um sinal exatamente igual ao aplicado à entrada da placa de som, sendo o sinal de entrada proveniente do gerador de funções. O sinal gerado na saída da placa de som USB poderá então ser observado com o uso do osciloscópio.

Figura 8 – Modelo do Simulink utilizado no Experimento 0.



Fonte: O autor (2018).

No modelo do Simulink observado na Figura 8, podem ser observados dois blocos: *ALSA Audio Capture* e *ALSA Audio Playback*. A seguir serão apresentadas breves explicações sobre esses blocos, bem como as configurações de seus parâmetros. Todos os experimentos com o Raspberry Pi que serão apresentados posteriormente utilizarão pelo menos um desses blocos. A diferença dos próximos experimentos em comparação com este consiste na adição de outros blocos do Simulink para o processamento dos sinais de entrada e de saída do Raspberry Pi.

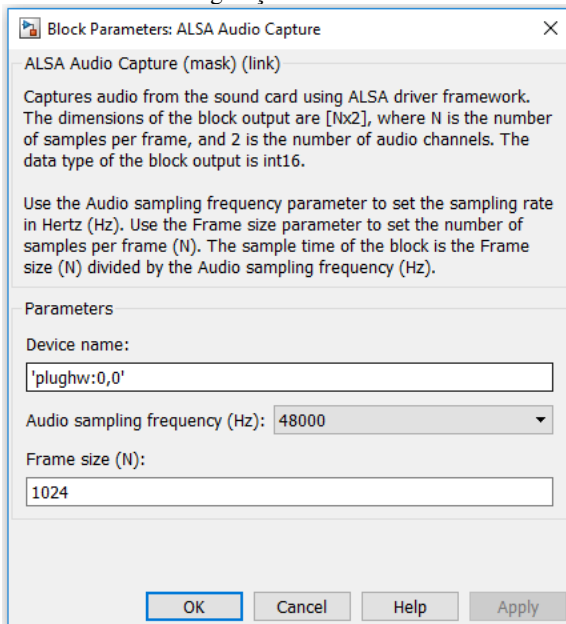
#### 4.1.1 Bloco ALSA Audio Capture

O bloco **ALSA Audio Capture** representa a entrada analógica do Raspberry Pi (entrada da placa de som USB). O bloco está relacionado ao driver de som ALSA do Raspberry Pi, que gerencia todos os

dispositivos de áudio conectados à placa, incluindo a placa de som USB (PASOLINI, BAZZI e MIRABELLA, 2016).

A janela de configurações do bloco *ALSA Audio Capture* pode ser observada na Figura 9. Nessa janela também são apresentadas algumas informações úteis sobre o bloco.

Figura 9 – Janela de configurações do bloco ALSA Audio Capture.



Fonte: O autor (2018).

Como pode ser observado na Figura 9, o bloco *ALSA Audio Capture* gera um sinal de saída de dimensão  $[N \times 2]$ , onde  $N$  é o número de amostras por quadro (*frame*) e 2 é o número de canais de áudio (a placa de som USB opera com sinais de áudio do tipo estéreo). Em outras palavras, o sinal à saída desse bloco é o sinal gerado pelo conversor ADC presente na placa de som USB. No caso deste experimento, o valor  $N = 1024$  é adotado. Cada amostra é representada no formato *int16*, isto é, um número inteiro de 15 bits + 1 bit de sinal (PASOLINI, BAZZI e MIRABELLA, 2016).

A taxa de amostragem é definida pelo parâmetro *Audio sampling frequency*. Nesse experimento, e em todos os demais que utilizarão esse

bloco, a taxa de amostragem a ser utilizada é de 48 kHz, que é o valor mais alto possível para a placa de som USB sendo utilizada.

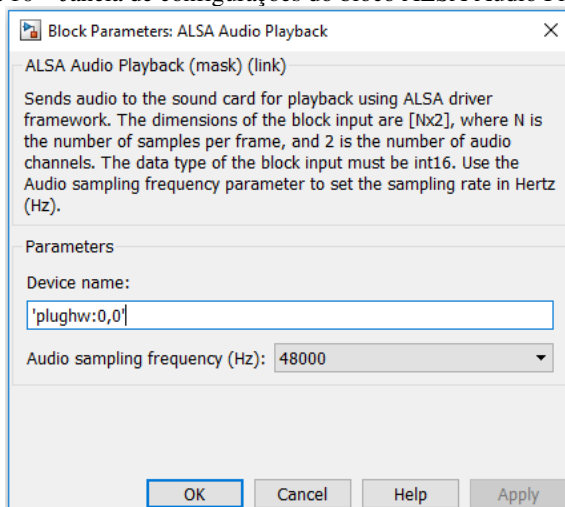
A identificação do dispositivo de entrada de som deve ser inserida no campo *Device name* respeitando a sintaxe “plughw:card,device” (PASOLINI, BAZZI e MIRABELLA, 2016). Os parâmetros “*card*” e “*device*” da entrada analógica da placa de som USB podem ser obtidos após a utilização do comando `arecord -l` no Shell do Linux. Se a configuração da placa de som foi realizada corretamente, então o parâmetro *Device name* sempre terá a forma “plughw:0,0”. Os autores também acrescentam que é essencial inicializar os parâmetros da placa de som USB utilizando o comando `alsamixer` no Shell.

#### 4.1.2 Bloco ALSA Audio Playback

O bloco **ALSA Audio Playback** representa a saída analógica do Raspberry Pi (saída da placa de som USB). Assim como o bloco *ALSA Audio Capture*, esse bloco está relacionado ao driver de som ALSA do Raspberry Pi.

A janela de configurações do bloco *ALSA Audio Playback* pode ser observada na Figura 10. Nessa janela também são apresentadas algumas informações úteis sobre o bloco.

Figura 10 – Janela de configurações do bloco ALSA Audio Playback.



Fonte: O autor (2018).

Como pode ser observado na Figura 10, o sinal de entrada do bloco ALSA Audio Playback deve ter dimensão  $[N \times 2]$ , onde  $N$  é o número de amostras por quadro e 2 é o número de canais de áudio. O sinal apresentado a esse bloco será o sinal de entrada do conversor DAC da placa de som USB.

No caso específico deste experimento, temos que  $N = 1024$ . Além disso, cada amostra apresentada a esse bloco deve ter o formato *int16*. O parâmetro *Audio sampling frequency* está relacionado à taxa de amostragem, cujo valor adotado é o máximo de 48 kHz.

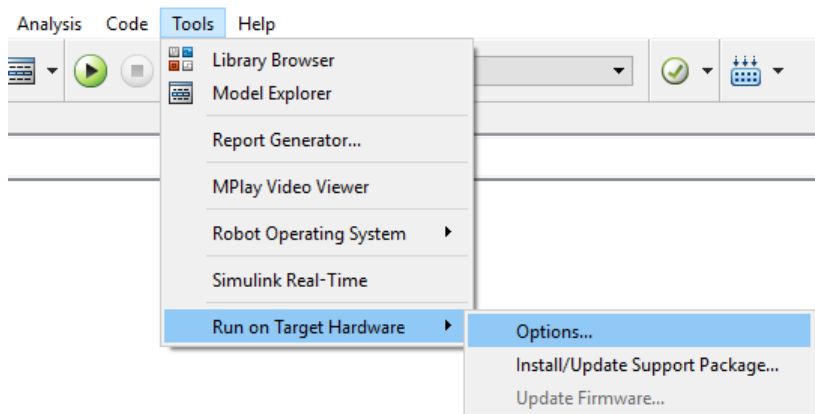
A identificação do dispositivo de saída de som deve ser inserida no campo *Device name* respeitando a sintaxe “*plughw:card,device*” (PASOLINI, BAZZI e MIRABELLA, 2016). Os parâmetros “*card*” e “*device*” da entrada analógica da placa de som USB podem ser obtidos após a utilização do comando `aplay -l` no Shell do Linux. Se a configuração da placa de som foi realizada corretamente, então o parâmetro *Device name* sempre terá a forma “*plughw:0,0*”.

#### 4.1.3 Configurações necessárias no Simulink

Após a implementação do modelo do Simulink ter sido finalizada, é necessário fornecer ao programa as informações necessárias para que ele transforme o modelo implementado num arquivo executável que de fato possa ser executado no Raspberry Pi (PASOLINI, BAZZI e MIRABELLA, 2016).

Primeiramente, no ambiente do Simulink, é necessário acessar o menu **Tools** → **Run on Target Hardware** → **Options**, conforme pode ser observado na Figura 11.

Figura 11 – Menu a ser acessado para a configuração do Simulink.

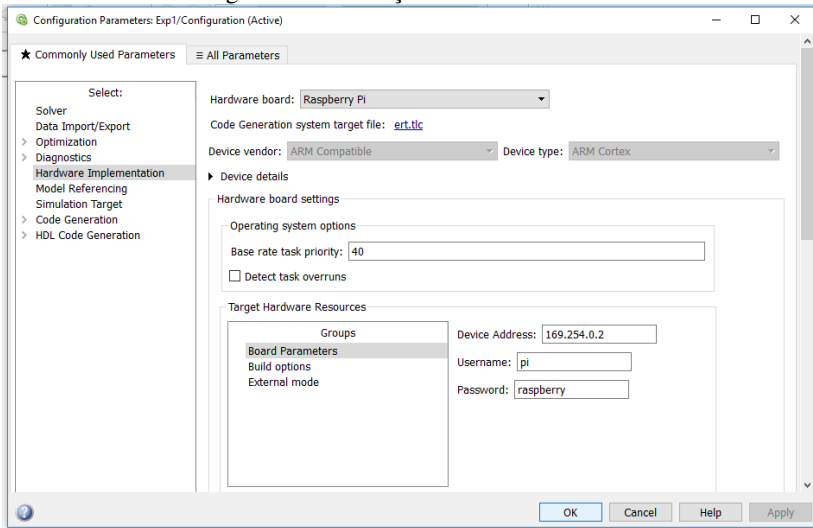


Fonte: O autor (2018).

Na sequência, aparecerá a janela de configuração que pode ser observada na Figura 12. Nessa janela, o hardware alvo, isto é, a placa na qual o modelo elaborado no Simulink será executado, é definido. Primeiramente, na aba **Hardware Implementation**, devemos definir o parâmetro **Hardware Board** como sendo **Raspberry Pi**.

Em seguida, no campo *Target Hardware Resources*, deve-se selecionar *Board Parameters*. Com isso, aparecerão os campos dos parâmetros **Device Address**, **Username** e **Raspberry Pi**, que devem ser definidos de forma coerente com a definição dos parâmetros de mesmo nome durante os Procedimentos Preliminares.

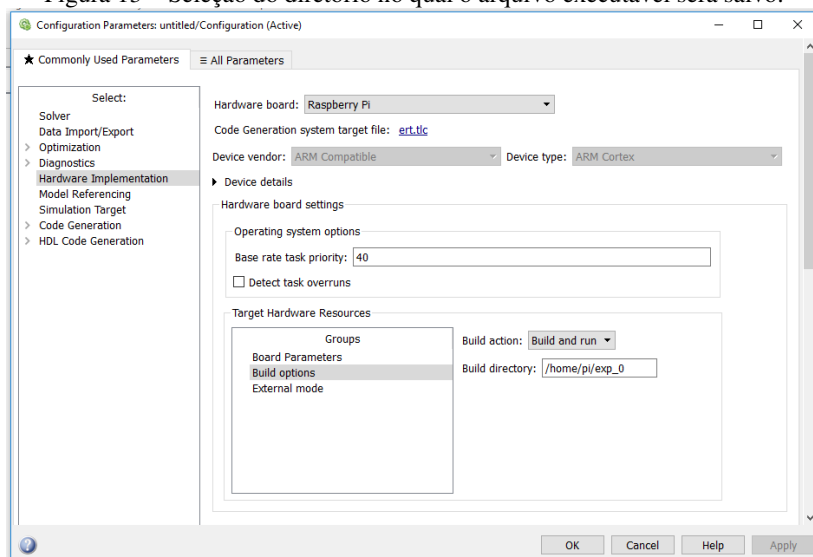
Figura 12 – Definição do hardware alvo.



Fonte: O autor (2018).

Por fim, devemos voltar ao campo **Target Hardware Resources** e seleccionar a opção **Build Options**, como pode ser observado na Figura 13.

Figura 13 – Seleção do diretório no qual o arquivo executável será salvo.



Fonte: O autor (2018).

No campo do parâmetro **Build directory** é possível definir o diretório da memória do Raspberry Pi no qual o arquivo executável do experimento será salvo. Recomenda-se que cada executável gerado nos experimentos seja salvo num diretório diferente. Após a realização de todas as configurações apresentadas nessa subseção, deve-se clicar em **Apply** e em seguida em **OK**. Com isso, a janela de configurações será fechada, e o Simulink estará pronto para a execução do experimento.

#### 4.1.4 Execução do experimento

Pasolini, Bazzi e Mirabella (2016) apresentam diferentes formas de executar no Raspberry Pi os modelos desenvolvidos no Simulink. Basicamente, os arquivos gerados após a compilação e salvos no Raspberry podem ser executados através do Simulink, através de comandos no MATLAB ou através de comandos no Shell do Linux.

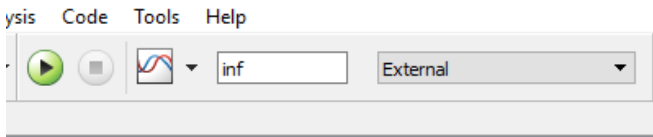
Há dois diferentes modos de execução dos experimentos desenvolvidos. O modo no qual o arquivo executável é salvo no Raspberry Pi e ele pode ser desconectado do PC para a execução do experimento é denominado modo *Deploy to Hardware* (em outras palavras, esse modo de execução permite que o Raspberry Pi atue como um dispositivo *stand-alone*). Já no modo de execução *External Mode*, o

arquivo executável é salvo no PC e é necessário manter o Raspberry Pi conectado a ele para a execução do experimento. Nesta subseção, os procedimentos para a realização do segundo modo serão apresentados, mas posteriormente também serão apresentados os procedimentos necessários para o primeiro.

Durante a execução no modo externo, é mantida uma comunicação entre o Raspberry Pi e o Simulink. Conseqüentemente, é possível interromper a execução ou alterar em tempo real parâmetros do sistema implementado através do ambiente do Simulink.

Para a execução no modo externo, é necessário definir o parâmetro do Simulink **Simulation mode** como **External** e **Simulation stop time** como **inf**, como mostrado na Figura 14.

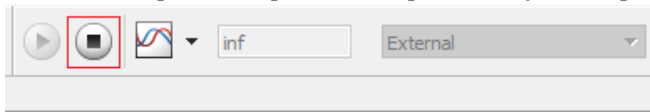
Figura 14 – Parâmetros no Simulink para execução em modo externo.



Fonte: O autor (2018).

Em seguida, é preciso apertar o botão **Run** (botão verde que aparece na Figura 14). Após alguns instantes, e se não houverem erros de compilação ou na comunicação com o Raspberry Pi, o modelo implementado no Simulink começará a funcionar no Raspberry Pi. Para interromper a execução do experimento, basta apertar o botão Stop que pode ser visto na Figura 15.

Figura 15 – Botão Stop utilizado para interromper a execução do experimento.



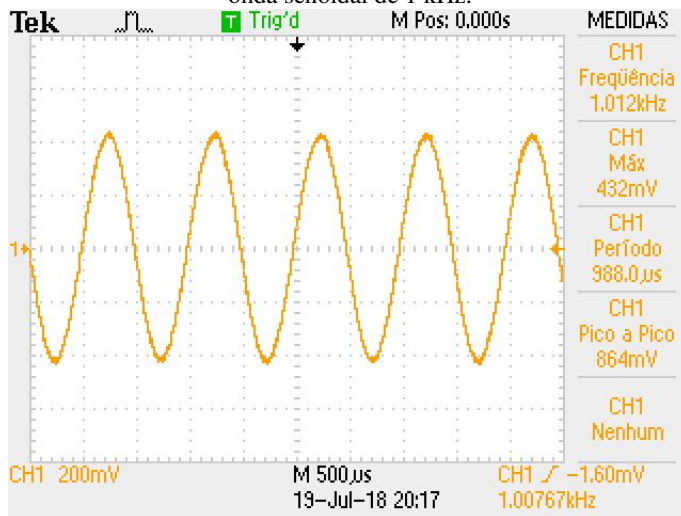
Fonte: O autor (2018).

Antes da execução do experimento, a saída do gerador de funções deve ser conectada à entrada analógica da placa de som USB, e a saída analógica da placa de som USB deve ser conectada a um dos canais do osciloscópio. Com o modelo desenvolvido no Simulink em execução no Raspberry Pi, diferentes sinais provenientes do gerador de funções



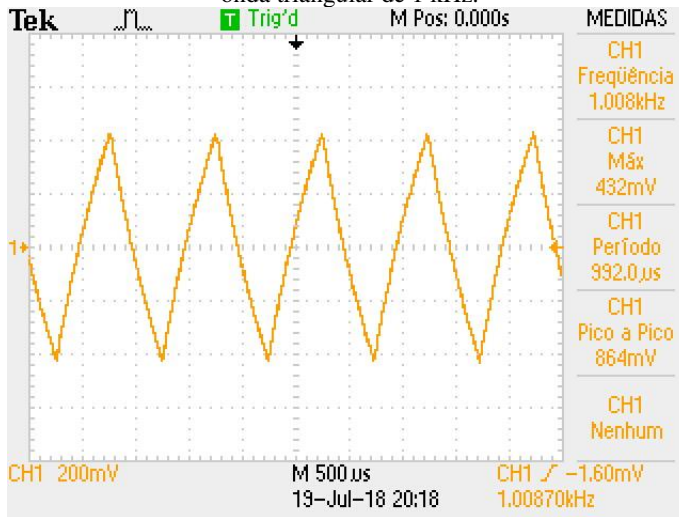
podem ser observados no osciloscópio após serem processados pelo Raspberry Pi e sua placa de som. Na Figura 16, na Figura 17 e na Figura 18, podemos observar o sinal de saída quando o sinal de entrada é uma onda senoidal, uma onda triangular e uma onda quadrada, respectivamente, tendo todos esses sinais a frequência aproximada de 1 kHz.

Figura 16 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda senoidal de 1 kHz.



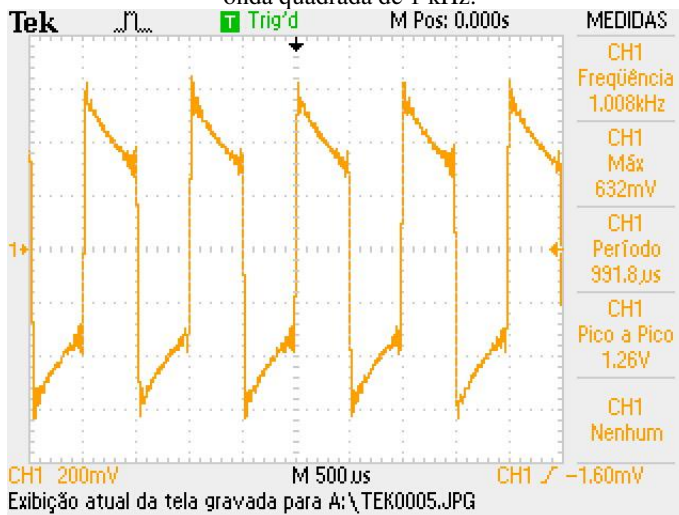
Fonte: O autor (2018).

Figura 17 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda triangular de 1 kHz.



Fonte: O autor (2018).

Figura 18 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda quadrada de 1 kHz.

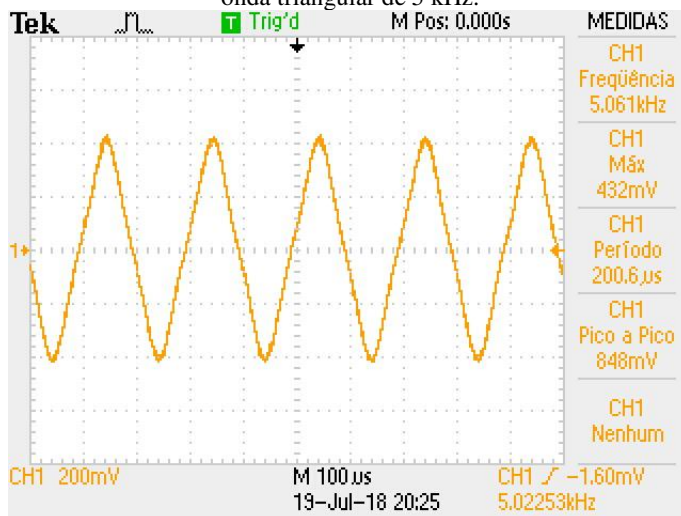


Fonte: O autor (2018).

Na Figura 16 e na Figura 17, podemos observar que o sinal senoidal e a onda triangular não foram distorcidos ao serem processados pela placa de som USB. Porém, o sinal que pode ser observado na Figura 18 evidencia que a onda quadrada foi severamente distorcida. Devido à limitação de frequência de 24 kHz da placa de som, todas as componentes do sinal de entrada acima dessa frequência são atenuadas. No caso da onda quadrada, muitas de suas componentes de alta frequência (que contribuem para que o sinal de fato possua a forma quadrada), foram suprimidas.

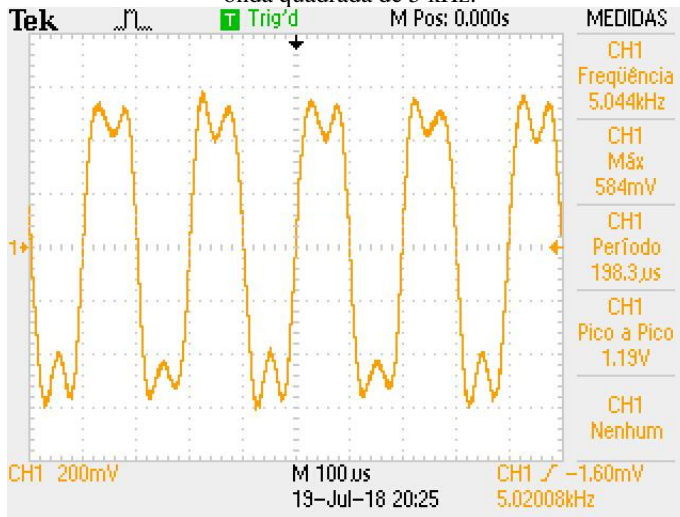
O efeito da limitação de frequência da placa de som fica ainda mais evidente quando sinais de entrada com frequências maiores são aplicados. Na Figura 19 e na Figura 20, podemos observar o sinal de saída quando os sinais de entrada são uma onda triangular de 5 kHz e uma onda quadrada de 5 kHz, respectivamente.

Figura 19 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda triangular de 5 kHz.



Fonte: O autor (2018).

Figura 20 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda quadrada de 5 kHz.

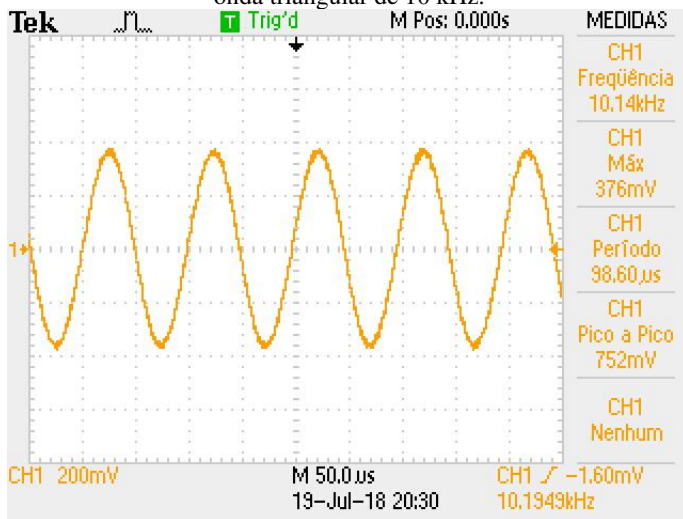


Fonte: O autor (2018).

Na Figura 19, podemos observar que algumas componentes de alta frequência do sinal de entrada triangular foram suprimidas, fazendo com que o sinal de saída fosse uma onda triangular com uma forma arredondada. Já na Figura 20, podemos observar que a placa de som USB suprimiu quase todas as componentes harmônicas da onda quadrada, fazendo com que o sinal de saída fosse apenas a soma de duas componentes senoidais.

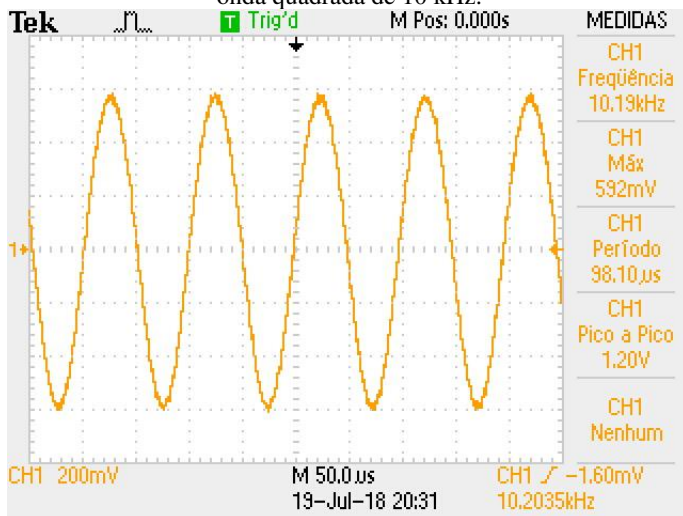
Quando ondas triangulares e quadradas de frequências ainda maiores são utilizadas como sinais de entrada, o efeito da limitação de frequência da placa de som é ainda mais extremo: o sinal de saída passa a ser somente a componente fundamental desses sinais. Na Figura 21, podemos observar o sinal de saída quando uma onda triangular de 10 kHz é utilizada como sinal de entrada. Já na Figura 22, podemos observar o sinal de saída quando uma onda quadrada de 10 kHz é utilizada como sinal de entrada.

Figura 21 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda triangular de 10 kHz.



Fonte: O autor (2018).

Figura 22 – Sinal de saída do Experimento 0, quando o sinal de entrada é uma onda quadrada de 10 kHz.



Fonte: O autor (2018).

Com a execução desse experimento inicial, é possível concluir que, para o correto funcionamento de outros experimentos mais complexos, os sinais de entrada a serem processados pela placa de som conectada ao Raspberry Pi não devem ter componentes significativas acima de 24 kHz. Também é importante observar que, para a execução desse experimento, o sinal de entrada aplicado deve ter amplitude suficientemente baixa para evitar a saturação do sinal de saída da placa de som.

#### 4.1.5 Procedimentos “Deploy to Hardware”

O modo de execução externa é útil para testes durante a implementação dos sistemas de comunicação, porém seria mais interessante possibilitar a atuação do Raspberry Pi como um dispositivo *stand-alone*. Ao implementar um sistema de comunicação no Simulink, converter o modelo em código executável, carregar o código na memória do Raspberry Pi, desconectar o Raspberry Pi do computador com o MATLAB e mensurar os sinais com o uso do osciloscópio, os estudantes podem se motivar mais por estarem manipulando um hardware que implementa um sistema de comunicações real, e não simplesmente um modelo utilizado em simulações no computador.

O objetivo dessa subseção é apresentar os procedimentos necessários para o modo de execução *Deploy to Hardware*. Os procedimentos que serão apresentados na sequência podem ser executados para todos os outros experimentos apresentados neste trabalho.

Após a implementação do modelo do Simulink ter sido finalizada, e uma vez que o Simulink tenha sido adequadamente configurado para gerar um arquivo executável para o Raspberry Pi (ver subseção 4.1.3), é necessário clicar no botão **Deploy to Hardware** localizado na parte superior da janela do Simulink, conforme pode ser observado na Figura 23.

Figura 23 – Localização do botão *Deploy to Hardware* no Simulink.



Após o usuário clicar no botão, o modelo do Simulink será transformado num código executável, que é salvo na memória do Raspberry Pi em conjunto com alguns arquivos auxiliares. Na sequência, a execução do experimento será iniciada automaticamente. Se o Raspberry Pi for desconectado do computador, o experimento permanecerá sendo executado.

Para interromper a execução de um experimento que foi inicializado através dos procedimentos de *Deploy to Hardware*, é necessário acessar o Shell do Linux através da *Command Window* do MATLAB, identificar o processo associado ao experimento e forçar o seu encerramento.

Primeiramente, a relação de todos os processos em execução do Raspberry Pi pode ser obtida com o uso do comando `top` no Shell. Após esse comando ter sido executado, é possível observar a janela mostrada na Figura 24. Para cada processo em execução, é atribuído um número de identificação (PID, do inglês *Process Identification*).

Figura 24 – Relação de processos em execução no Raspberry Pi.

```

pi@raspberrypi-tominaga: ~
top - 10:51:11 up 2:34, 1 user, load average: 0.26, 0.09, 0.14
Tasks: 94 total, 1 running, 93 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.1 us, 0.2 sy, 0.0 ni, 97.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 883052 total, 223784 used, 659268 free, 15184 buffers
KiB Swap: 102396 total, 0 used, 102396 free. 168324 cached Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 4134 root        rt   0 22036  3568 2880 S   9.3   0.4   0:24.45 Exp2.elf
 4160 pi          20   0   5112   2416 2080 R   0.3   0.3   0:00.14 top
    1 root        20   0 23824  3816 2680 S   0.0   0.4   0:05.24 systemd
    2 root        20   0     0     0     0 S   0.0   0.0   0:00.00 kthreadd
    3 root        20   0     0     0     0 S   0.0   0.0   0:00.32 ksoftirqd/0
    5 root        0 -20   0     0     0 S   0.0   0.0   0:00.00 kworker/0:+
    7 root        20   0     0     0     0 S   0.0   0.0   0:00.34 rcu_sched
    8 root        20   0     0     0     0 S   0.0   0.0   0:00.00 rcu_bh
    9 root        rt   0     0     0     0 S   0.0   0.0   0:00.01 migration/0
   10 root        rt   0     0     0     0 S   0.0   0.0   0:00.02 migration/1
   11 root        20   0     0     0     0 S   0.0   0.0   0:00.02 ksoftirqd/1
   13 root        0 -20   0     0     0 S   0.0   0.0   0:00.00 kworker/1:+
   14 root        rt   0     0     0     0 S   0.0   0.0   0:00.01 migration/2
   15 root        20   0     0     0     0 S   0.0   0.0   0:00.05 ksoftirqd/2
   17 root        0 -20   0     0     0 S   0.0   0.0   0:00.00 kworker/2:+
   18 root        rt   0     0     0     0 S   0.0   0.0   0:00.01 migration/3
   19 root        20   0     0     0     0 S   0.0   0.0   0:00.03 ksoftirqd/3

```

Fonte: O autor (2018).

O processo `Exp2.elf` localizado na primeira linha da relação de processos mostrada na Figura 24 corresponde ao experimento sendo executado. O PID associado a esse processo é 4134. Uma vez que o PID

do experimento foi identificado, a janela com a relação de todos os processos pode ser fechada ao pressionar-se a tecla `q`.

Para forçar o encerramento do processo, e com isso interromper o experimento em execução, é necessário utilizar o comando `sudo kill PID` no Shell do Linux, onde “PID” corresponde ao PID do processo do experimento em execução (nesse caso, o comando `sudo kill 4134` deve ser utilizado).

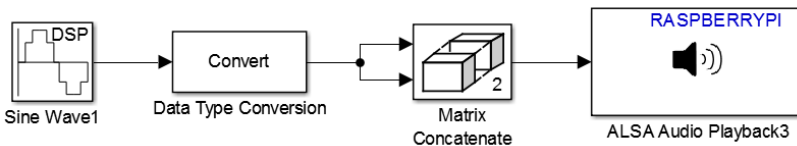
## 4.2 RASPBERRY PI COMO UM GERADOR DE FUNÇÕES

O segundo experimento a ser realizado com a plataforma didática baseada no Raspberry Pi consiste em transformar a placa num gerador de funções. Esse experimento é uma adaptação do primeiro experimento proposto por Pasolini, Bazzi e Mriabella (2016). No trabalho desses autores, eles utilizaram o Raspberry Pi para gerar apenas um sinal de tensão senoidal. No experimento adaptado para este trabalho, o Raspberry Pi também será utilizado para gerar outras formas de onda.

O principal objetivo deste experimento é permitir que os alunos se familiarizem com a execução em tempo real de modelos desenvolvidos no Simulink e também com a análise de sinais nos domínios do tempo e da frequência utilizando o osciloscópio. Além disso, esse experimento pode ser utilizado como um exercício sobre a representação de sinais periódicos através da Série de Fourier.

O diagrama de blocos mostrado na Figura 25, que foi implementado no Simulink, tem a função de transformar o Raspberry Pi num gerador de um sinal senoidal. Esse diagrama de blocos foi adaptado do trabalho de Pasolini, Bazzi e Mirabella (2016).

Figura 25 – Modelo do Simulink utilizado para gerar um sinal senoidal na saída da placa de som USB.



Fonte: O autor (2018).

Além do bloco *ALSA Audio Playback*, que foi apresentado no experimento anterior, três outros blocos podem ser observados na Figura 25: *Sine Wave*, *Data Type Conversion* e *Matrix Concatenate*. Esses

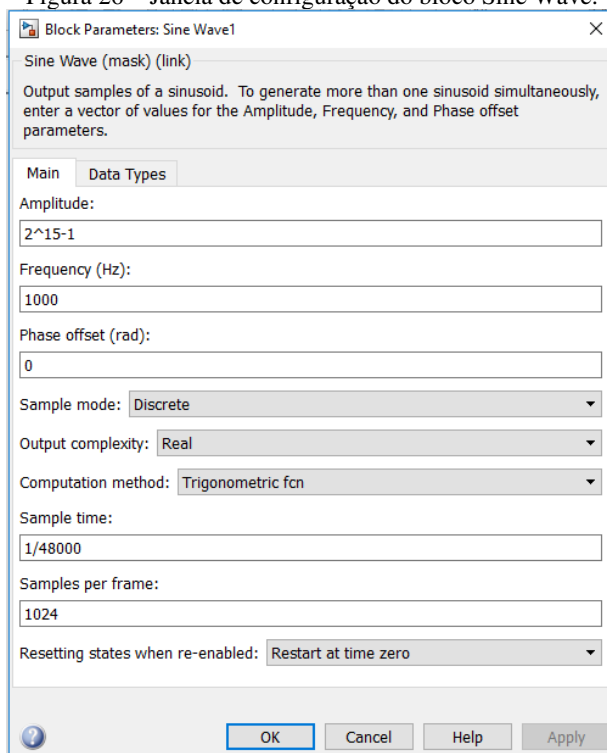


blocos serão utilizados com frequência nos próximos experimentos, e por esse motivo serão apresentadas a seguir explicações sobre eles.

### 4.2.1 Bloco Sine Wave

O bloco **Sine Wave** é utilizado para gerar uma senóide em sua saída. Sua janela de configurações pode ser observada na Figura 26. Nessa janela, podemos configurar diversos parâmetros do sinal, como amplitude, frequência, *offset* de fase e período de amostra (*Sample Time*).

Figura 26 – Janela de configuração do bloco Sine Wave.



Fonte: O autor (2018).

Uma vez que a máxima taxa de amostragem suportada pela placa de som USB é de 48000 amostras/s, o período de amostragem definido para a senóide é de 1/48000 s. Isso limita o valor máximo de frequência

da senóide a ser gerada, que não deve passar de 24 kHz. Conforme pode ser observado na Figura 26, o valor de frequência definido para a senóide é de 1 kHz.

As amostras do sinal a ser processado pela placa de som devem ter o formato *int16*. Nessa representação de 15 bits + 1 bit de sinal, o maior número inteiro possível é igual a  $2^{15} - 1$ . Portanto, esse é o valor adotado como amplitude da senóide gerada.

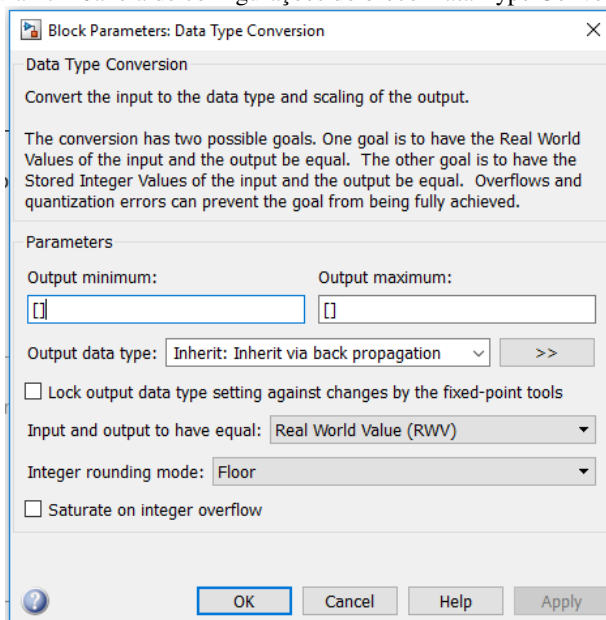
Para a geração de uma senóide em tempo discreto, o parâmetro *Sample mode* deve ser definido como *Discrete*. Além disso, para a geração de uma senóide e não uma exponencial complexa, o parâmetro *Output complexity* deve ser definido como *Real*.

O parâmetro *Samples per frame* define o número de amostras contidas nos quadros que compõe o sinal de saída do bloco. Nesse experimento, utilizamos quadros de  $N = 1024$  amostras. Como afirmam Pasolini, Bazzi e Mirabella (2016), para aumentar a eficiência do processamento do Raspberry Pi (e em geral de todos os sistemas embarcados), o processamento por quadros é preferível em relação ao processamento de *streams* de amostras.

#### **4.2.2 Bloco Data Type Conversion**

O bloco Data Type Conversion converte o tipo de dado de um sinal de entrada em qualquer outro tipo de dado do Simulink. Sua janela de configurações pode ser observada na Figura 27.

Figura 27 – Janela de configurações do bloco Data Type Conversion.



Fonte: O autor (2018).

Conforme mencionado anteriormente, o bloco *ALSA Audio Playback* deve receber amostras do tipo *int16*. Porém, as amostras na saída do bloco *Sine Wave* são do tipo *Real*. Nas configurações do bloco *Data Type Conversion*, ao definirmos o parâmetro *Output data type* como sendo *Inherit: inherit via back propagation*, o bloco automaticamente converte o tipo de dado das amostras de saída no tipo de dado requerido pelos estágios seguintes do processamento do sinal (PASOLINI, BAZZI e MIRABELLA, 2016).

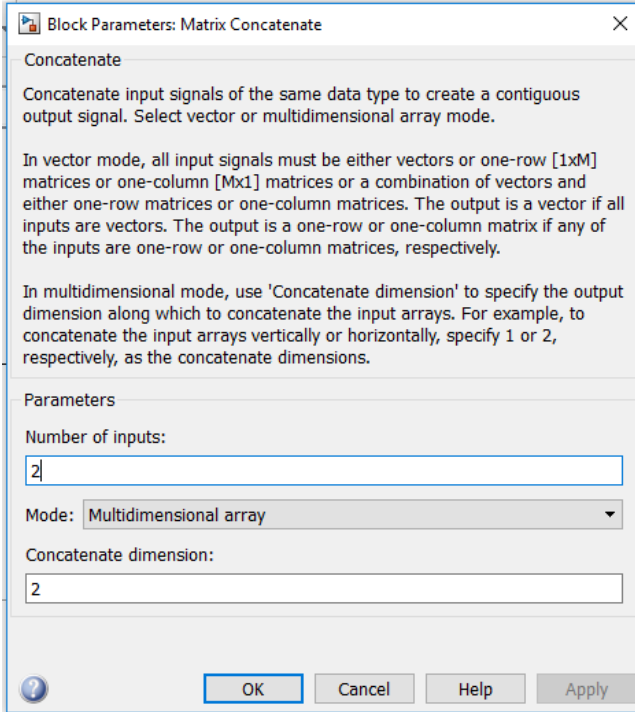
#### 4.2.3 Bloco Matrix Concatenate

O bloco *Matrix Concatenate* tem a função de concatenar múltiplos sinais de entrada num único sinal de saída. O sinal de saída pode ser tanto um vetor quanto um *array* multidimensional (PASOLINI, BAZZI e MIRABELLA, 2016).

No diagrama de blocos mostrado na Figura 25 (e nos experimentos posteriores), o bloco *Matrix concatenate* tem a função de produzir um sinal de dois canais (isto é, um sinal estéreo) a partir de um sinal de entrada com apenas um canal (mono). Essa operação é

necessária visto que o bloco ALSA Audio Playback requer como sinal de entrada um sinal estéreo (PASOLINI, BAZZI e MIRABELLA, 2016). A janela de configurações do bloco *Matrix concatenate* pode ser observada na Figura 28, na qual podem ser vistas informações adicionais sobre o bloco.

Figura 28 – Janela de configurações do bloco Matrix concatenate

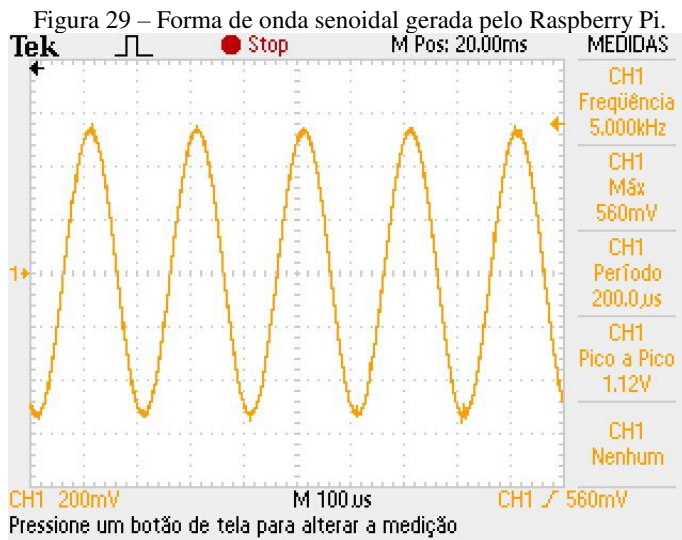


Fonte: O autor (2018).

Como o resultado da operação do bloco *Matrix concatenate* mostrado na Figura 25, o sinal estéreo em sua saída apresenta dimensão [1024x2], sendo que os sinais em cada um de seus canais são idênticos.

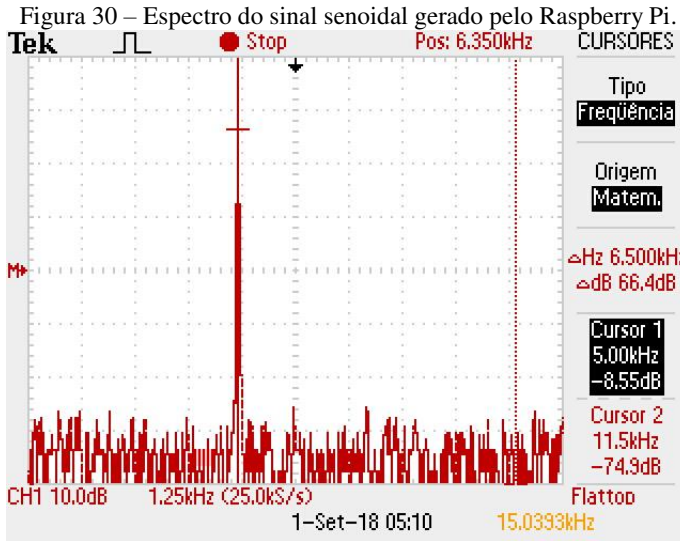
#### 4.2.4 Geração do sinal senoidal

O modelo do Simulink mostrado na Figura 25 foi configurado para que o Raspberry Pi gerasse uma senóide de 5 kHz na saída da placa de som USB. Após o início da execução do modelo do Simulink, foi possível observar na tela do osciloscópio o sinal mostrado na Figura 29.



Fonte: O autor (2018).

Utilizando a função matemática FFT do osciloscópio, é possível observar o espectro (de magnitude) do sinal senoidal mostrado anteriormente na Figura 29. Esse espectro, por sua vez, pode ser observado na Figura 30.



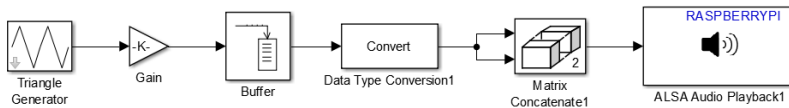
Fonte: O autor (2018).

Na Figura 30, a ferramenta Cursor do osciloscópio é utilizada para mostrar que, conforme esperado, o espectro do sinal gerado pelo Raspberry Pi é constituído por apenas uma componente impulsiva na frequência de 5 kHz.

#### 4.2.5 Geração de uma forma de onda triangular com o Raspberry Pi

O modelo do Simulink mostrado na Figura 31 é utilizado para gerar uma forma de onda triangular na saída da placa de som. As únicas diferenças desse diagrama de blocos em relação ao mostrado na Figura 25 é a substituição do bloco *Sine Wave* pelo bloco *Triangle Generator* e a inclusão de um bloco de ganho e de um bloco *Buffer*.

Figura 31 – Modelo do Simulink utilizado para gerar uma forma de onda triangular na saída da placa de som USB.

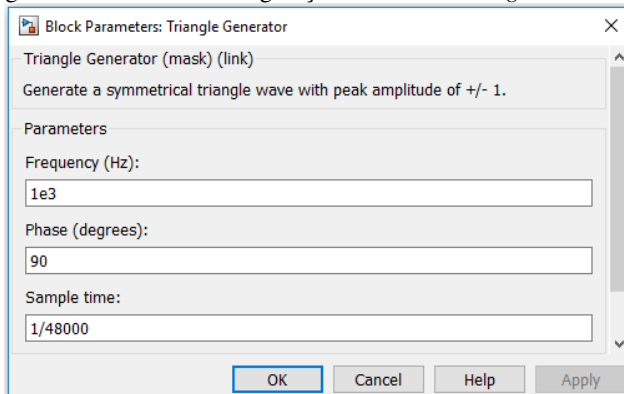


Fonte: O autor (2018).

A janela de configurações do bloco *Triangle Generator* pode ser observada na Figura 32. Esse bloco gera uma onda triangular simétrica

com amplitude unitária. Para a realização deste experimento, a frequência da onda triangular foi definida com 1 kHz, e o período de amostragem como 1/48000 s.

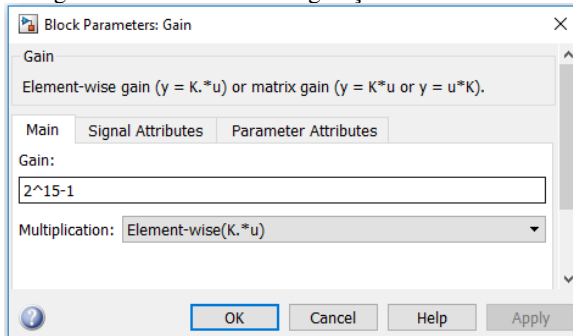
Figura 32 – Janela de configurações do bloco *Triangle Generator*.



Fonte: O autor (2018).

A janela de configurações do bloco *Gain* pode ser observada na Figura 33. A função desse bloco no modelo do Simulink mostrado na Figura 31 é multiplicar todas as amostras da onda triangular por um valor de ganho constante igual a  $2^{15} - 1$ .

Figura 33 – Janela de configurações do bloco *Gain*.

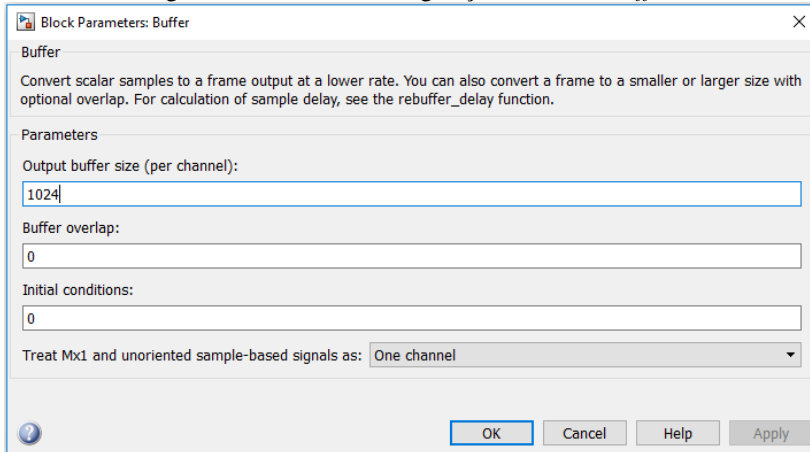


Fonte: O autor (2018).

A janela de configurações do bloco *Buffer* pode ser observada na Figura 34. Esse bloco tem a função de agrupar as amostras do sinal de

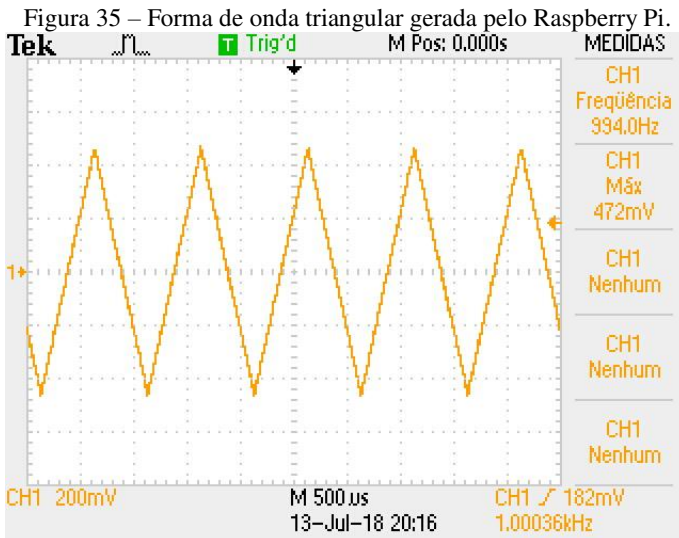
entrada em quadros, que neste caso possuem o tamanho  $N = 1024$  amostras.

Figura 34 – Janela de configurações do bloco *Buffer*.



Fonte: O autor (2018).

Após início da execução do modelo do Simulink, é possível observar na tela do osciloscópio o sinal mostrado na Figura 35.

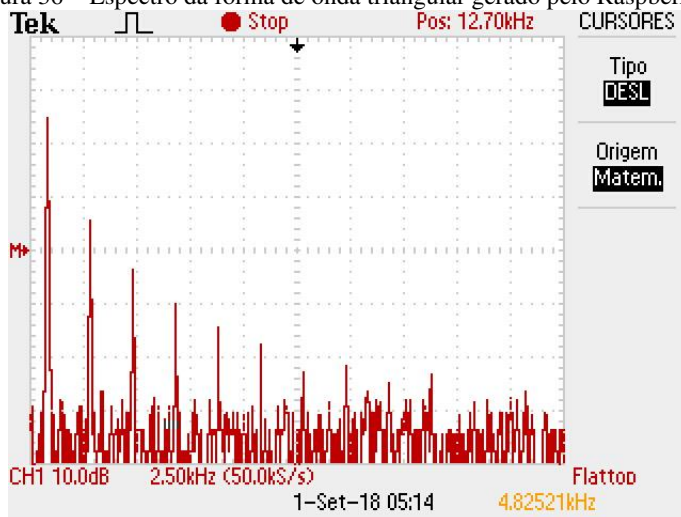


Fonte: O autor (2018).



Utilizando a ferramenta FFT do osciloscópio, é possível observar o espectro da forma de onda triangular gerada pelo Raspberry Pi. Esse espectro, por sua vez, pode ser observado na Figura 36.

Figura 36 – Espectro da forma de onda triangular gerado pelo Raspberry Pi.



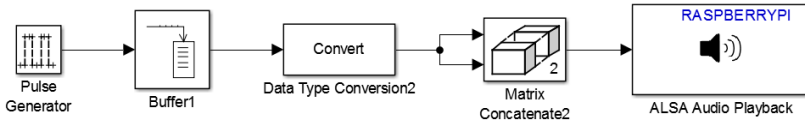
Fonte: O autor (2018).

Conforme pode ser observado na Figura 36, o espectro da forma de onda triangular é constituído por componentes discretas em frequências múltiplas de 1 kHz, que é a frequência fundamental do sinal.

#### 4.2.6 Geração de uma forma de onda quadrada com o Raspberry Pi

O modelo do Simulink mostrado na Figura 37 é utilizado para gerar uma forma de onda quadrada na saída da placa de som USB. As principais mudanças em relação ao diagrama de blocos mostrado na Figura 25 é a substituição do bloco *Sine Wave* pelo bloco *Pulse Generator* e a adição do mesmo bloco *Buffer* utilizado no gerador de onda triangular.

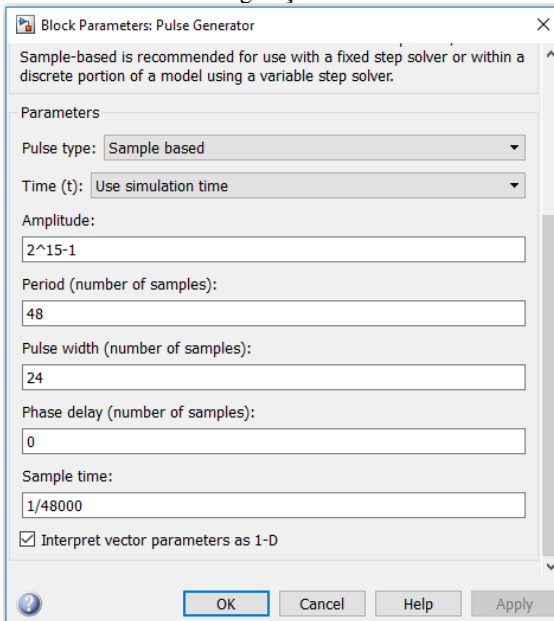
Figura 37 – Modelo do Simulink utilizado para gerar uma forma de onda quadrada na saída da placa de som USB.



Fonte: O autor (2018).

A janela de configurações do bloco *Pulse Generator* pode ser observada na Figura 38.

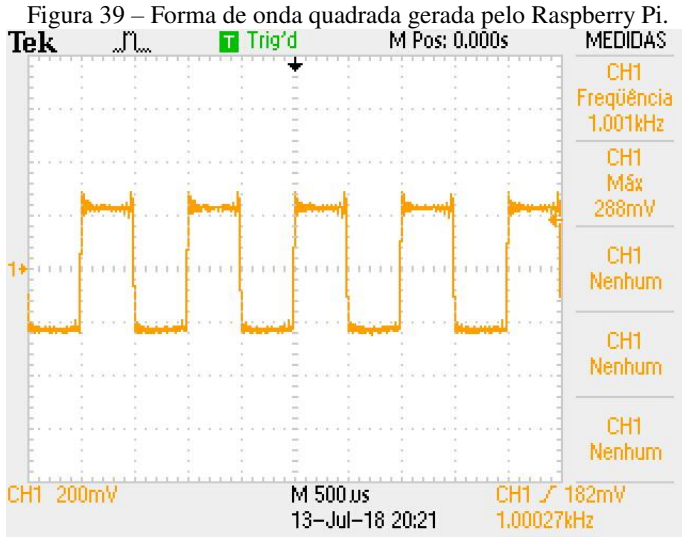
Figura 38 – Janela de configurações do bloco *Pulse Generator*.



Fonte: O autor (2018).

O bloco *Pulse Generator* foi configurado para gerar uma onda quadrada com amplitude igual a  $2^{15} - 1$  e período de amostragem de  $1/48000$  segundos (isto é, geração de 48000 amostras/s). Para definir uma forma de onda quadrada com frequência de 1 kHz e *duty cycle* de 50%, o número de amostras por período foi definido como 48, e a largura do pulso foi definida como de 24 amostras.

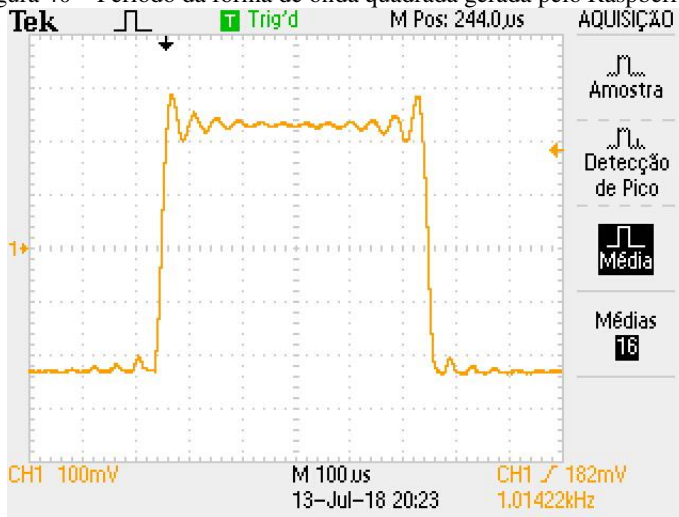
Após o início da execução do modelo do Simulink mostrado na Figura 37, foi possível observar na tela do osciloscópio o sinal mostrado na Figura 39.



Fonte: O autor (2018).

Na Figura 40 podemos observar com mais detalhes um período da forma de onda quadrada mostrada na Figura 39. Nessa Figura, podemos observar claramente o efeito do Fenômeno de Gibbs na forma de onda quadrada. Esse efeito surge sempre que o número de harmônicas de um sinal periódico não for infinito.

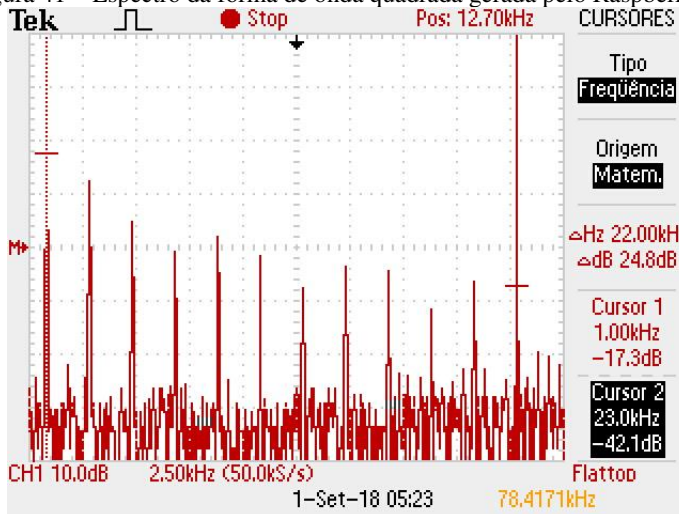
Figura 40 – Período da forma de onda quadrada gerada pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da ferramenta FFT do osciloscópio, é possível observar o espectro da forma de onda quadrada gerada pelo Raspberry Pi. Esse espectro, por sua vez, pode ser observado na Figura 41.

Figura 41 – Espectro da forma de onda quadrada gerada pelo Raspberry Pi.



Fonte: O autor (2018).

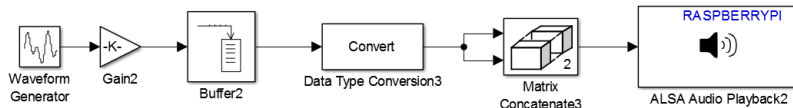
Conforme pode ser observado na Figura 41, o espectro da forma de onda quadrada gerada pelo Raspberry Pi é constituído por componentes impulsivas em frequências múltiplas de 1 kHz, que é a frequência fundamental do sinal. Além disso, dois cursores foram adicionados ao espectro: um desses cursores indica a frequência fundamental, e o outro indica a frequência de 23 kHz, que corresponde à componente de mais alta frequência que compõe a onda quadrada.

Uma forma de onda quadrada ideal teria um número infinito de componentes. Porém, o número de harmônicas dos sinais gerados neste experimento é limitado pela placa de som USB, que suprime as componentes dos sinais gerados acima de 24 kHz. Essa limitação de frequência é a responsável pelo fenômeno de Gibbs observado.

#### 4.2.7 Geração de outras formas de onda com o Raspberry Pi

O modelo do Simulink mostrado na Figura 42 é utilizado para gerar diferentes formas de onda na saída da placa de som.

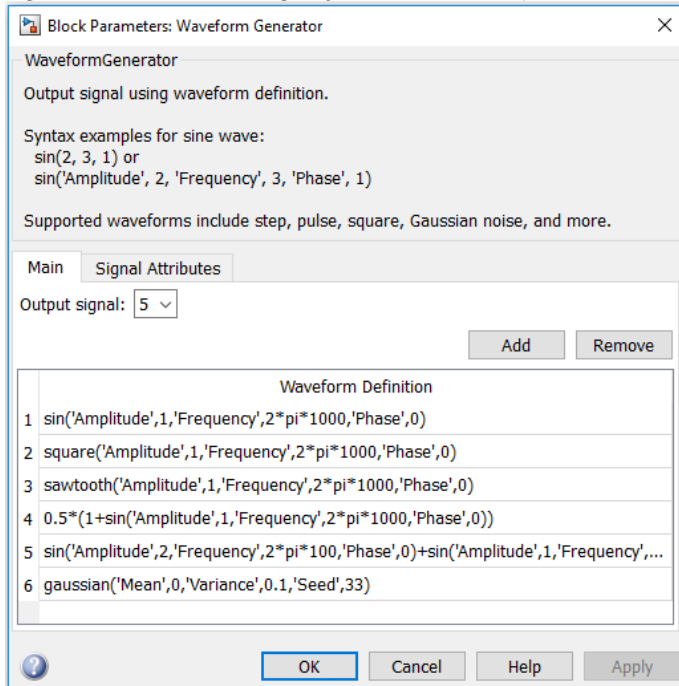
Figura 42 – Modelo do Simulink utilizado para gerar diferentes formas de onda com o Raspberry Pi.



Fonte: O autor (2018).

A forma de onda a ser gerada é proveniente do bloco do Simulink *Waveform Generator*. A janela de configurações desse bloco pode ser observada na Figura 43.

Figura 43 – Janela de configurações do bloco *Waveform Generator*.



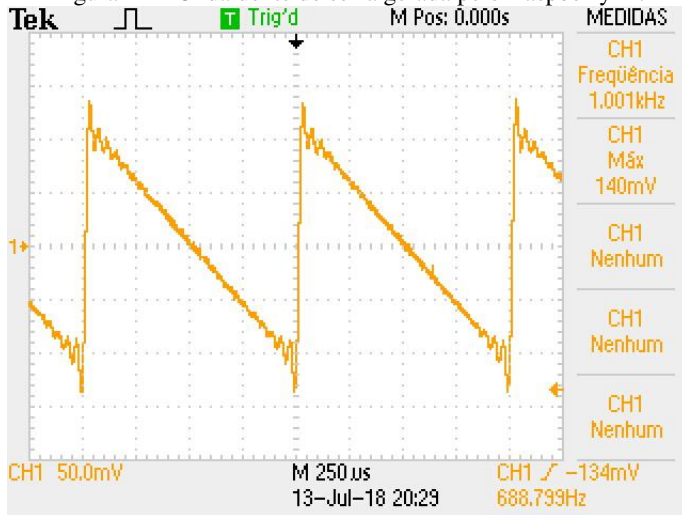
Fonte: O autor (2018).

Na janela de configurações do bloco *Waveform Generator*, é possível definir diferentes formas de onda, como ondas senoidais, triangulares ou quadradas (formas de onda tradicionais, que podem ser geradas por um gerador de funções comum). Também é possível gerar sinais que não são gerados por um gerador de funções comum, como dente de serra, ruído gaussiano, soma de senóides, etc. Utilizando combinações dessas formas de onda, com seus devidos parâmetros ajustados, é possível gerar uma infinidade de diferentes formas de onda.

Os blocos *Gain* e *Buffer* que podem ser observados na Figura 42 são exatamente iguais aos blocos mostrados anteriormente na Figura 31.

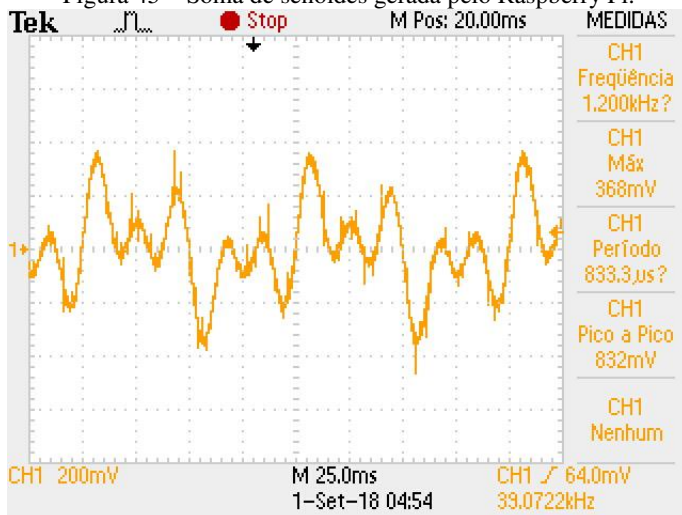
Alguns exemplos de formas de onda geradas na saída da placa de som após a execução do modelo do Simulink da Figura 42 podem ser observadas na Figura 44 e na Figura 45. Na primeira Figura, o bloco *Waveform Generator* foi configurado para gerar uma onda dente de serra, enquanto que na segunda Figura o bloco foi configurado para gerar uma soma de três senóides.

Figura 44 – Onda dente de serra gerada pelo Raspberry Pi.



Fonte: O autor (2018).

Figura 45 – Soma de senóides gerada pelo Raspberry Pi.

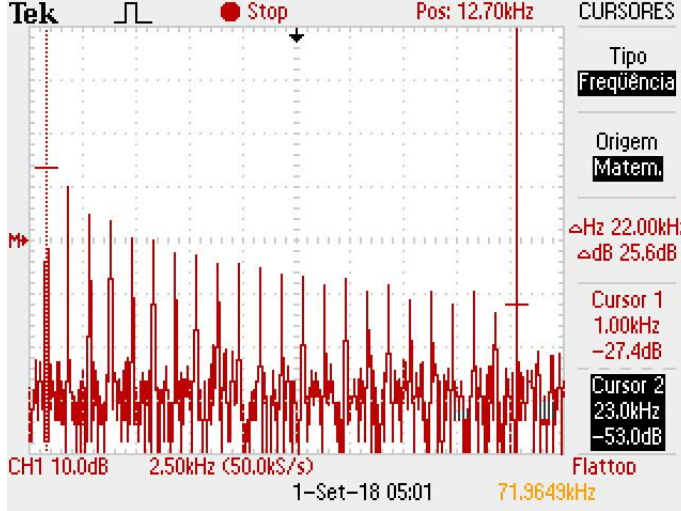


Fonte: O autor (2018).

Com o uso da ferramenta FFT do osciloscópio, é possível observar os espectros da onda dente de serra e do sinal composto pela

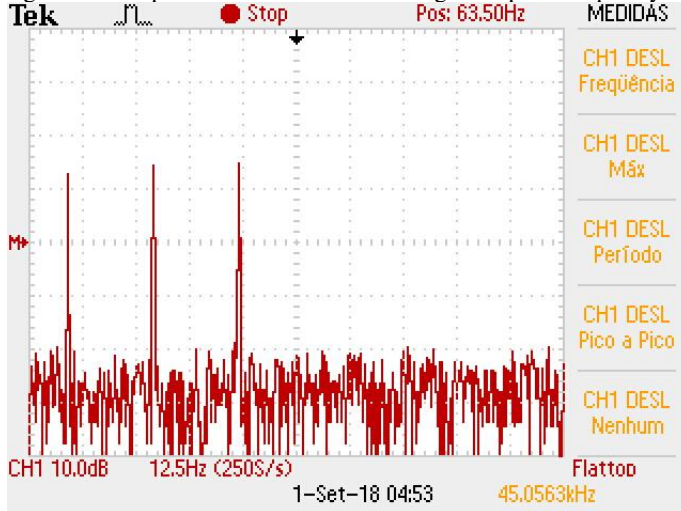
soma de senóides. Esses espectros podem ser observados na Figura 46 e na Figura 47, respectivamente.

Figura 46 – Espectro da onda dente de serra gerada pelo Raspberry Pi.



Fonte: O autor (2018).

Figura 47 – Espectro da soma de senóides gerada pelo Raspberry Pi.



Fonte: O autor (2018).



Na Figura 46 é possível observar o espectro onda dente de serra gerada pelo Raspberry Pi. Assim como no espectro da forma de onda quadrada mostrado anteriormente, no espectro da onda dente de serra é possível observar o efeito da limitação de frequência da placa de som USB.

#### 4.2.8 Geração de sinais a partir de arquivos .wav

Nessa subseção, serão apresentados os procedimentos para a reprodução de um arquivo .wav na saída da placa de som USB. Primeiramente, para fins de ilustração, o Raspberry Pi será transformado num gerador de um sinal de ECG (eletrocardiograma) artificial, sendo que as amostras do sinal gerado serão provenientes de um arquivo .wav.

As amostras do sinal de ECG real (isto é, gerado pela atividade elétrica do coração de uma pessoa, e não sintetizado através de uma soma de senóides) foram obtidas na plataforma PhysioNet (GOLDBERGER *et al.*, 2000). De forma pública e gratuita, essa plataforma fornece diversas bases de dados contendo amostras de sinais fisiológicos de pacientes reais.

A base de dados consultada para este trabalho foi a MIT-BIH *Normal Sinus Rhythm Database*. Essa base de dados inclui 18 registros de longa duração de indivíduos encaminhados ao Laboratório de Arritmia do Beth Israel Hospital de Boston (EUA). Os indivíduos incluídos nessa base de dados não apresentaram arritmias significativas.

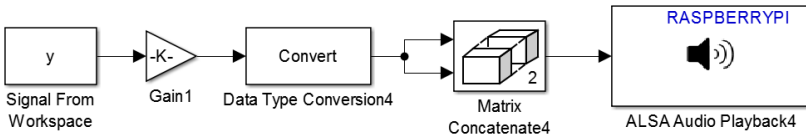
Nessa base de dados, foi obtido um arquivo do MATLAB (.m) contendo amostras de uma gravação de longa duração do ECG de um paciente. Essas amostras foram obtidas a uma taxa de amostragem de 128 Hz. Uma parte das amostras correspondendo a um trecho de aproximadamente 40 segundos da gravação foi selecionada, a amplitude do sinal foi normalizada (isto é, a amplitude máxima do sinal passou a ser unitária), e em seguida esse trecho foi convertido para um arquivo .wav. Com o uso de uma interpolação por um fator 375, a partir do primeiro arquivo .wav foi gerado um segundo arquivo com amostras a uma taxa de 48 kHz. Esse arquivo foi salvo com o nome `ecg_48kHz.wav`.

O arquivo .wav deve estar salvo no mesmo diretório onde se encontra o modelo do Simulink a ser utilizado. Na sequência, na *Command Window* do MATLAB, o comando `[y, Fs]=audioread('ecg_48kHz.wav');` deve ser utilizado.

Esse comando gera duas novas variáveis no *Workspace* do MATLAB: uma variável  $y$ , contendo as amostras do sinal contido no arquivo `.wav`, e uma variável  $F_s$ , contendo o valor da taxa de amostragem do sinal (que nesse caso é de 48 kHz).

Após a realização dos procedimentos anteriores, o modelo do Simulink que pode ser observado na Figura 48 é utilizado para reproduzir na saída da placa de som USB o arquivo `.wav`.

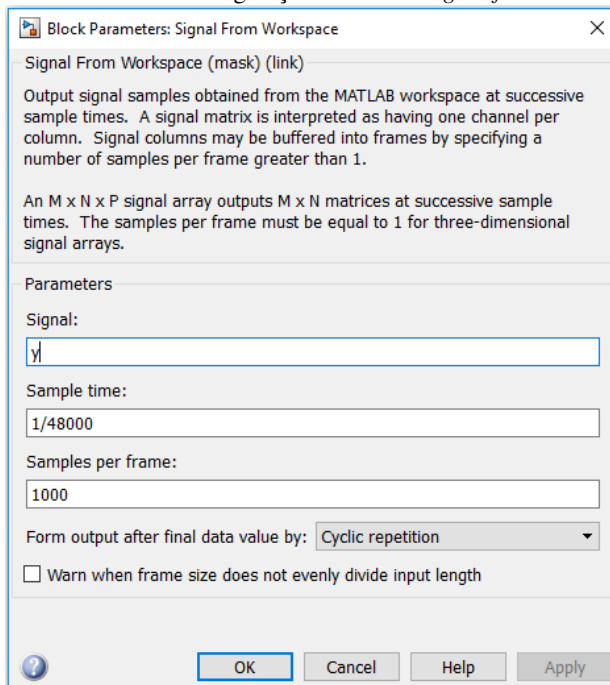
Figura 48 – Modelo do Simulink utilizado para reproduzir um arquivo `.wav` na saída da placa de som USB.



Fonte: O autor (2018).

O bloco *Signal From Workspace*, que pode ser observado na Figura 48, é utilizado para importar as amostras de uma variável armazenada no *Workspace* do MATLAB. A janela de configurações desse bloco pode ser observada na Figura 49.

Figura 49 – Janela de configurações do bloco *Signal from Workspace*.



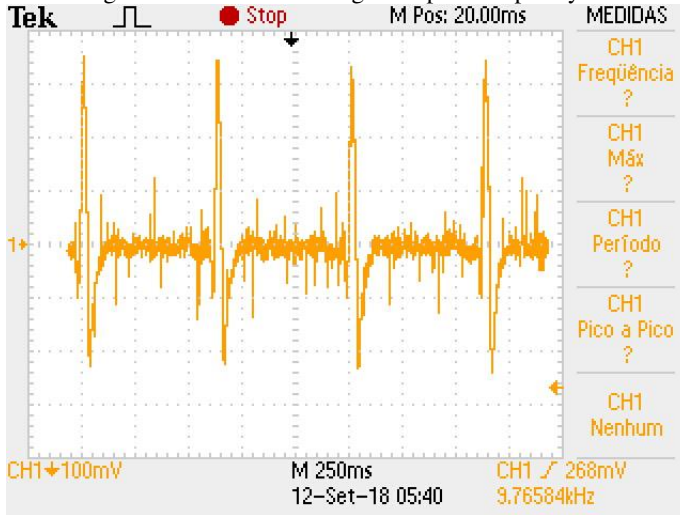
Fonte: O autor (2018).

O bloco *Signal from Workspace* é configurado para importar as amostras do sinal armazenado na variável  $y$  (que contém o sinal proveniente do arquivo `.wav`). A taxa de amostragem utilizada é de 48 kHz (parâmetro *Sample time* igual a  $1/48000$  s) e o sinal de saída do bloco é constituído por quadros de 1000 amostras. Além disso, o parâmetro *Form output after final data value by:* deve ser definido como *Cyclic repetition*. Dessa forma, sempre que o sinal sendo reproduzido chegar ao seu fim, ele será novamente iniciado.

O bloco *Gain* que pode ser observado na Figura 48 tem a função de aplicar um ganho igual a  $2^{15} - 1$  ao sinal a ser reproduzido.

Após o início da execução do modelo do Simulink, é possível observar na tela do osciloscópio o sinal mostrado na Figura 50.

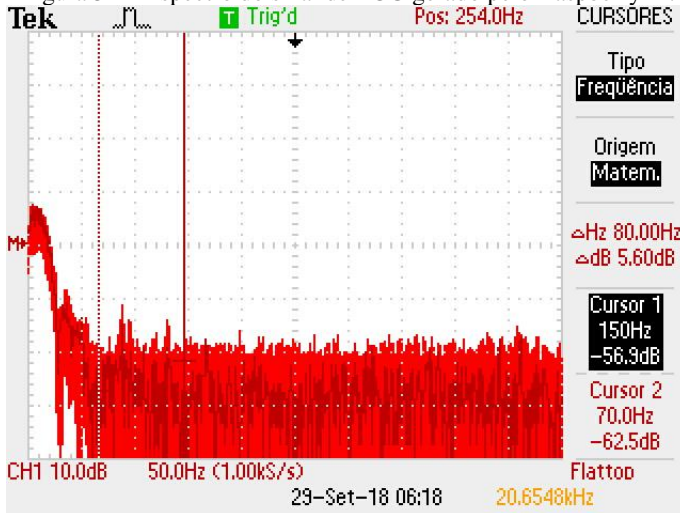
Figura 50 – Sinal de ECG gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da ferramenta FFT do osciloscópio, é possível observar o espectro do sinal de ECG sendo reproduzido pelo Raspberry Pi. Esse espectro, por sua vez, pode ser observado na Figura 51.

Figura 51 – Espectro do sinal de ECG gerado pelo Raspberry Pi.



Fonte: O autor (2018).

O espectro mostrado na Figura 51 foi obtido com o uso da função de persistência do osciloscópio digital. Essa função permite que os valores máximos de cada componente de frequência do sinal sendo mensurado permaneçam na tela, o que (em alguns casos) pode facilitar a análise do sinal no domínio da frequência. Sendo assim, conforme pode ser observado no espectro mostrado na Figura 51, a maior parte da potência do sinal de ECG concentra-se nas componentes de baixa frequência (abaixo de 70 Hz).

O Raspberry Pi configurado como um gerador de um sinal de ECG pode ser utilizado em aulas práticas de disciplinas relacionadas a Engenharia Biomédica. O sinal gerado pode ser utilizado para o teste do funcionamento de cadeias de aquisição de ECG. Após a cadeia ser testada com esse sinal e apresentar um funcionamento correto, ela pode ser utilizada para a aquisição de sinais de ECG de pacientes reais.

### 4.3 RASPBERRY PI COMO UM FILTRO DIGITAL

O experimento proposto nesta seção pode ser utilizado em aulas introdutórias de Processamento Digital de Sinais para o ensino dos principais conceitos básicos relacionados ao projeto de filtros digitais.

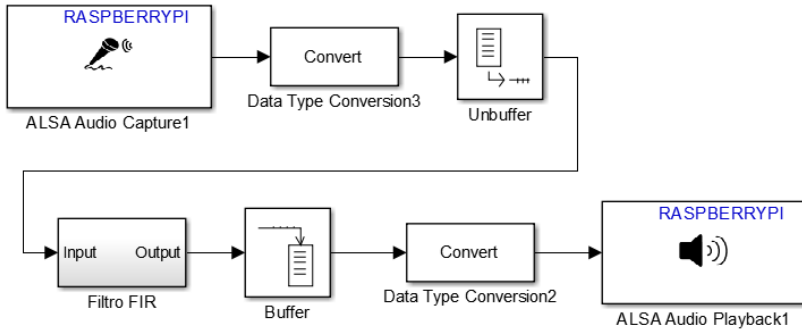
O primeiro objetivo do experimento é apresentar aos estudantes a diferença entre as topologias (em diagrama de blocos) de filtros digitais dos tipos FIR e IIR. Em seguida, o objetivo será apresentar aos estudantes a ferramenta do MATLAB *FDATool*, que pode ser utilizada para transformar o Raspberry Pi em qualquer tipo de filtro digital.

#### 4.3.1 Raspberry Pi como um filtro digital do tipo FIR

O modelo do Simulink que pode ser observado na Figura 52 é utilizado para transformar o Raspberry Pi num filtro digital do tipo FIR passa-baixas de ordem 6, com frequência de corte de 10 kHz. Os blocos *ALSA Audio Capture*, *Data Type Conversion*, *Unbuffer*, *Buffer* e *ALSA Audio Playback* que podem ser observados na Figura já foram apresentados anteriormente.

Os quadros à saída do bloco *ALSA Audio Capture* apresentam tamanho de 1024 amostras. O bloco *Unbuffer* converte a sequência de quadros numa sequência de amostras a ser filtrada. Na sequência, o bloco *Buffer* organiza as amostras resultantes da filtragem em quadros de tamanho de 1024 amostras, que são apresentados à entrada do bloco *ALSA Audio Playback*.

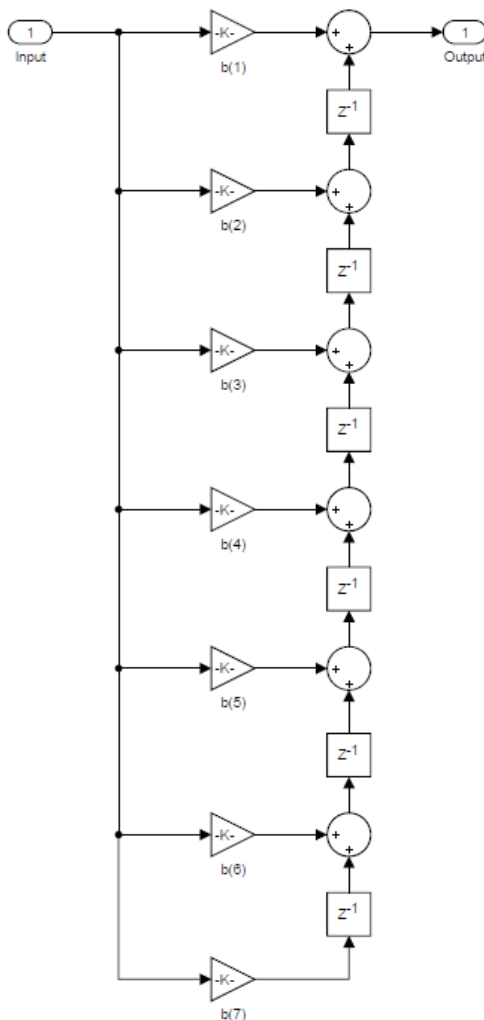
Figura 52 – Modelo do Simulink utilizado para transformar o Raspberry Pi num filtro digital FIR.



Fonte: O autor (2018).

Na Figura 53 é possível observar os componentes internos do subsistema “Filtro FIR” que pode ser observado na Figura 52. Esse diagrama de blocos implementa o filtro digital FIR passa-baixas, ordem 6, com frequência de corte de 10 kHz. A obtenção desse diagrama de blocos (incluindo os coeficientes do filtro) será explicada posteriormente.

Figura 53 – Componentes internos do subsistema “Filtro FIR”, utilizado no modelo do Simulink que transforma o Raspberry Pi num filtro FIR.



Fonte: O autor (2018).

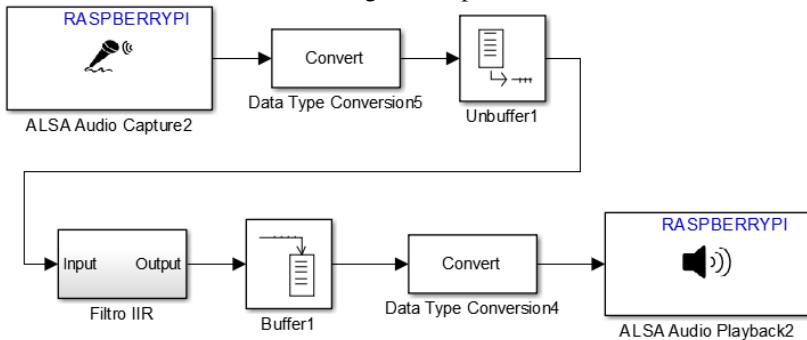
Após o início da execução do experimento, os estudantes podem aplicar sinais senoidais de diferentes frequências (provenientes do gerador de funções) na entrada da placa de som USB, e simultaneamente observar o sinal à saída da placa de som USB (com o uso do osciloscópio). Ao fazer uma varredura de frequências, os estudantes

poderão constatar que o modelo do Simulink mostrado na Figura 52 de fato transforma o Raspberry Pi num filtro digital passa-baixas com frequência de corte em 10 kHz.

### 4.3.2 Raspberry Pi como um filtro digital do tipo IIR

O modelo do Simulink que pode ser observado na Figura 54 é utilizado para transformar o Raspberry Pi num filtro digital do tipo IIR passa-baixas de ordem 4, com frequência de corte de 10 kHz. Esse modelo é muito semelhante ao mostrado anteriormente na Figura 52. A única diferença entre os dois modelos é a substituição do subsistema “Filtro FIR” pelo subsistema “Filtro IIR”.

Figura 54 – Modelo do Simulink utilizado para transformar o Raspberry Pi num filtro digital do tipo IIR.

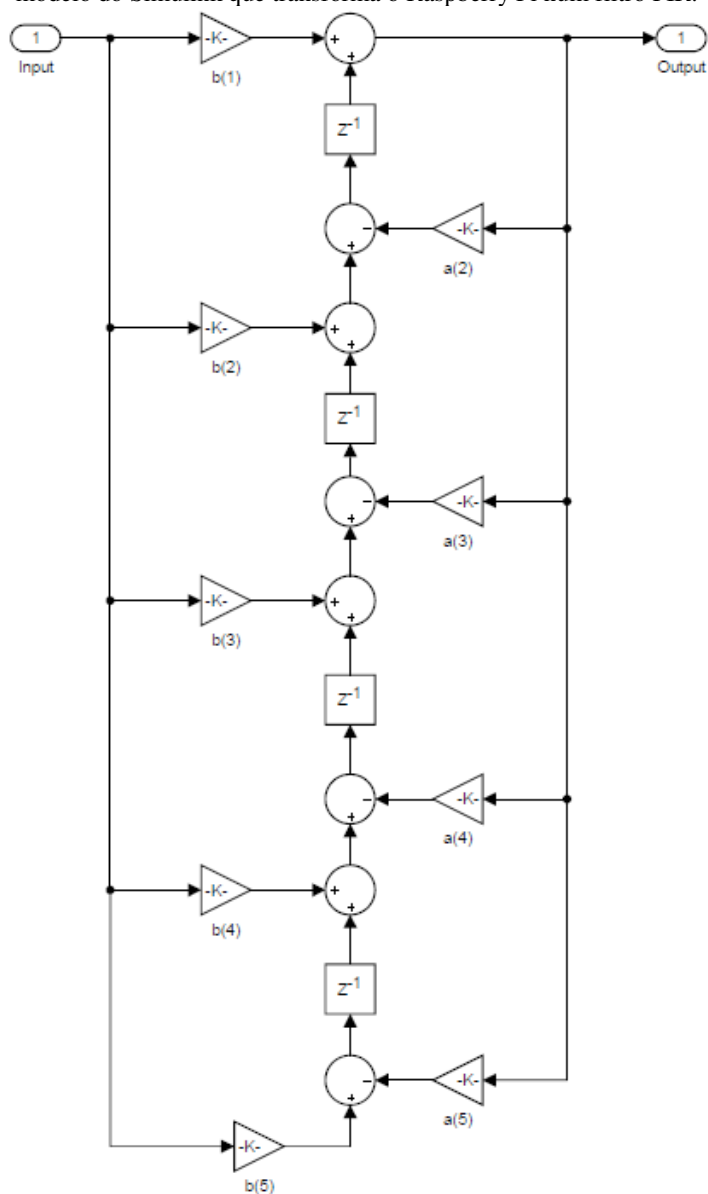


Fonte: O autor (2018).

Na Figura 55 é possível observar os componentes internos do subsistema “Filtro IIR” que pode ser observado na Figura 54. Esse diagrama de blocos implementa o filtro digital IIR passa-baixas, ordem 4, com frequência de corte de 10 kHz. Assim como no caso do filtro FIR apresentado anteriormente, a obtenção desse diagrama de blocos (incluindo os coeficientes do filtro) será explicada posteriormente.



Figura 55 – Componentes internos do subsistema “Filtro IIR”, utilizado no modelo do Simulink que transforma o Raspberry Pi num filtro FIR.



Fonte: O autor (2018).

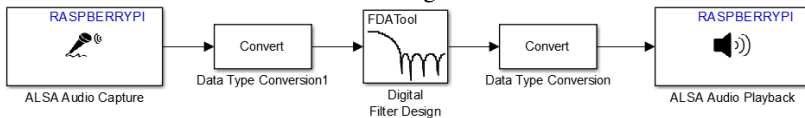
De forma similar ao exemplo anterior com o filtro FIR, após o início da execução do experimento, os estudantes podem aplicar sinais senoidais de diferentes frequências (provenientes do gerador de funções) na entrada da placa de som USB, e simultaneamente observar o sinal à saída da placa de som USB (com o uso do osciloscópio). Ao fazer uma varredura de frequências, os estudantes poderão constatar que o modelo do Simulink mostrado na Figura 54 de fato transforma o Raspberry Pi num filtro digital passa-baixas com frequência de corte em 10 kHz.

Nesse ponto do experimento, algumas diferenças interessantes entre os filtros digitais dos tipos FIR e IIR podem ser discutidas. Primeiramente, a principal diferença entre esses dois tipos de filtros consiste na ausência de realimentação no primeiro e a presença dela no segundo (justamente por causa dessa realimentação, a resposta ao impulso do filtro IIR é infinita). Além disso, pode ser notado que, para as mesmas especificações (filtro passa-baixas com frequência de corte de 10 kHz), o filtro IIR possui uma ordem menor do que o filtro FIR.

### 4.3.3 Raspberry Pi como qualquer tipo de filtro digital

O modelo do Simulink que pode ser observado na Figura 56, proposto por Pasolini, Bazzi e Mirabella (2016), é utilizado para transformar o Raspberry Pi em qualquer tipo de filtro digital.

Figura 56 – Modelo do Simulink utilizado para transformar o Raspberry Pi num filtro digital.



Fonte: O autor (2018).

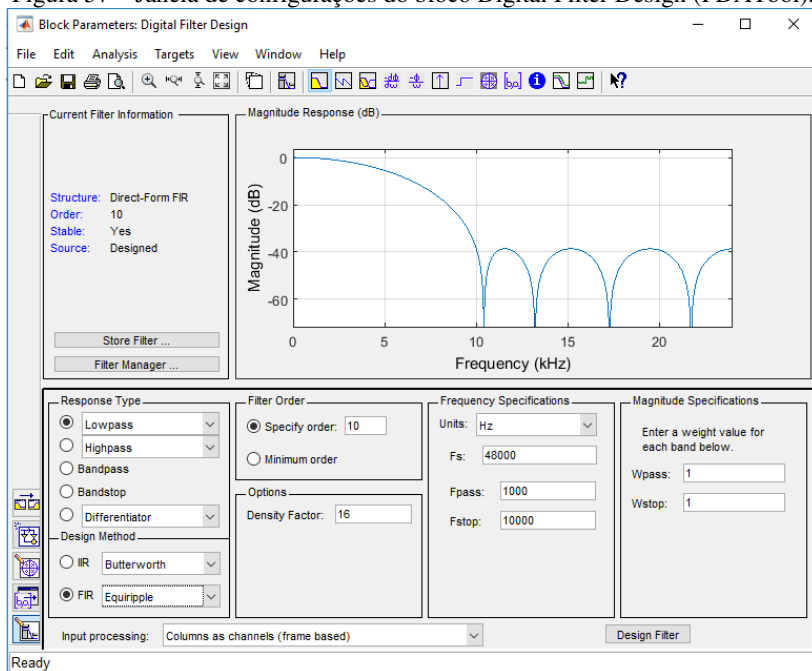
Os blocos *ALSA Audio Capture*, *Data Type Conversion* e *ALSA Audio Playback* que podem ser observados na Figura 56 já foram apresentados nos experimentos anteriores. Já o bloco *Digital Filter Design* (FDATool) é apresentado pela primeira vez neste experimento. Esse bloco é uma ferramenta extremamente poderosa dos softwares MATLAB e Simulink.

Na Figura 57 é possível observar a janela de configurações do bloco *Digital Filter Design*. A grande vantagem da utilização dessa ferramenta é a possibilidade de gerar uma grande quantidade de diferentes filtros digitais de forma extremamente fácil. É possível gerar

filtros com características do tipo passa baixas, passa altas, passa faixa, rejeita faixa, entre outras. Além disso, é possível definir se o filtro digital a ser gerado será do tipo FIR ou IIR, sendo que para cada um desses dois tipos é possível selecionar um entre diferentes métodos de projeto.

Com relação à ordem do filtro digital a ser gerado, o usuário pode especificar um determinado valor ou permitir que a ferramenta determine a mínima ordem para atender as especificações desejadas. As especificações de frequência do filtro, por sua vez, podem ser especificadas em valores de frequências normalizadas (frequências digitais em radianos) ou em Hertz. Para a especificação das frequências em Hertz, é necessário inserir o valor correto da taxa de amostragem sendo utilizada pelo sistema (no caso dos experimentos apresentados neste trabalho, a taxa de amostragem é de 48 kHz). Quando o usuário estabelece que a ferramenta deve determinar a ordem mínima do filtro para satisfazer as especificações, é necessário inserir os valores de atenuação nas bandas passantes e nas bandas de rejeição, sendo que esses valores podem ser inseridos em escala linear ou em dB.

Figura 57 – Janela de configurações do bloco Digital Filter Design (FDATool).

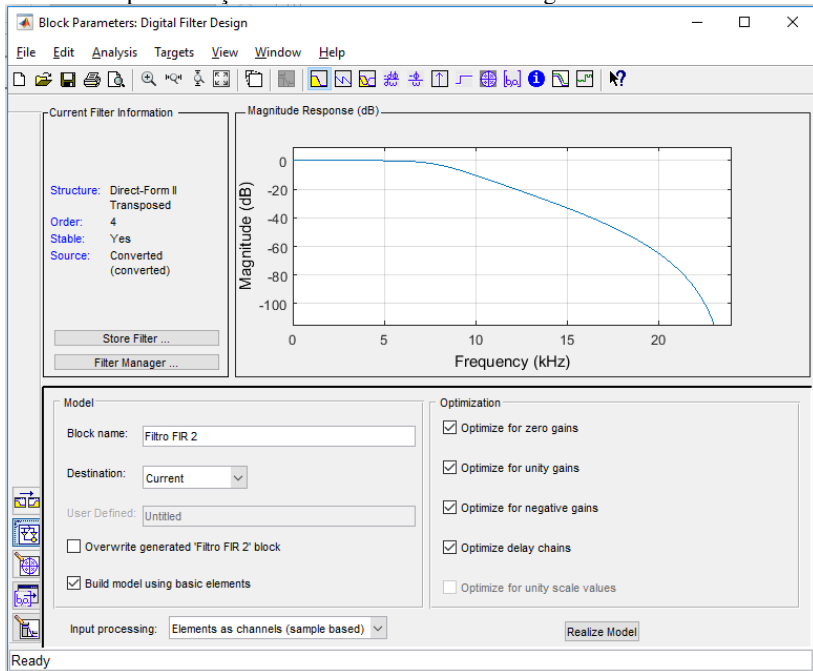


Fonte: O autor (2018).

Após todas as especificações necessárias terem sido inseridas pelo usuário, é preciso clicar em **Design Filter** para que a ferramenta FDATool gere o filtro digital. Se a geração do filtro for bem sucedida, é possível observar na interface da ferramenta os diagramas de magnitude e fase do filtro gerado. Além disso, a ferramenta também indica a ordem do filtro e se ele é estável.

A ferramenta FDATool foi utilizada para gerar os subsistemas “Filtro FIR” e “Filtro IIR” utilizados anteriormente. Essa ferramenta conta com uma opção de implementação automática de filtros digitais em diagrama de blocos do Simulink, que pode ser acessada ao clicar-se na aba *Realize Model* no canto esquerdo inferior da janela. Na Figura 58 é possível visualizar essa aba na ferramenta FDATool.

Figura 58 – Aba *Realize Model* na ferramenta FDATAol, acessada para a implementação automática de filtros em diagramas de blocos.



Fonte: O autor (2018).

Após a definição das características do filtro digital desejado, deve-se clicar no botão *Realize Model* (observado na Figura 58) para a geração automática de um subsistema do Simulink contendo o diagrama de blocos que implementa o filtro desejado. Mais ainda, antes de gerar o diagrama de blocos, na parte superior da ferramenta FDATAol, ao acessar **Edit** → **Convert Structure**, é possível determinar o tipo de estrutura do filtro (forma direta I, forma direta II, ou uma das formas transposta). Em **Edit** ainda há a opção de gerar o filtro utilizando apenas estruturas de segunda ordem.

#### 4.3.4 Sugestões de experimentos práticos

O experimento prático que transforma o Raspberry Pi num filtro digital é extremamente interessante para ser utilizado com estudantes cursando disciplinas introdutórias de Comunicações, visto que esses

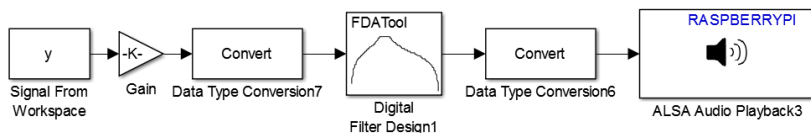
possuem os conhecimentos básicos de Sinais e Sistemas e pouco conhecimento sobre Processamento Digital de Sinais.

Os procedimentos a serem realizados no Laboratório são relativamente simples. Com o uso do gerador de funções, sinais senoidais de diferentes frequências podem ser gerados e aplicados na entrada da placa de som USB. Nesta etapa é importante lembrar-se de três detalhes importantes: o ganho do microfone deve ser ajustado previamente com o uso da ferramenta Alsamixer, a amplitude dos sinais de entrada deve ser suficientemente baixa para evitar a saturação do sinal de saída, e os sinais senoidais a serem aplicados não podem ter frequência acima de 24 kHz. O sinal na saída da placa de som USB, que corresponde ao sinal filtrado, deve ser observado com o uso do osciloscópio.

Neste experimento, os estudantes poderão observar na prática (isto é, com o uso de sinais reais) o comportamento de diferentes filtros digitais. Com isso, fixarão os principais conceitos relacionados ao projeto desses filtros. Entre os conceitos a serem estudados, estão a característica do filtro (passa faixa, passa altas, passa banda, rejeita faixa, etc.), as diferentes bandas de sua resposta em frequência (banda passante, banda de transição e banda de rejeição), o significado das atenuações nas bandas passante e de transição, o impacto da ordem do filtro na extensão da banda de transição e no atraso de processamento, a conversão entre frequências em Hertz em frequências normalizadas, entre muitos outros. Além disso, com o uso da ferramenta FDATool, os estudantes poderão conhecer como é a forma das características de magnitude e fase de diferentes filtros FIR e IIR, e posteriormente poderão comparar as características visualizadas na ferramenta com curvas obtidas experimentalmente. Mais ainda, será possível mostrar que um filtro digital IIR consegue atender determinadas especificações de frequências com uma ordem menor em comparação com um filtro FIR.

Outro experimento interessante que pode ser realizado em aulas de laboratório pode ser a filtragem digital de sinais de áudio (voz ou música). Para esse experimento, o modelo do Simulink mostrado na Figura 59 pode ser utilizado.

Figura 59 – Modelo do Simulink utilizado para realizar a filtragem de sinais de áudio.



Fonte: O autor (2018).

O modelo do Simulink mostrado na Figura 59 é muito semelhante aos modelos mostrados anteriormente nesta seção. A diferença do novo modelo é que neste as amostras do sinal a ser filtrado são provenientes do bloco *Signal From Workspace*, que foi utilizado no experimento para geração de um sinal de ECG (mais detalhes podem ser consultados na subseção 4.2.8). Esse bloco deve ser utilizado para importar amostras de um arquivo de áudio *.wav*. O parâmetro *Samples per frame* pode ser definido como 1024, e o período de amostra (*Sample time*) como 1/48000 segundos. Assumindo-se que a amplitude das amostras está normalizada, o bloco *Gain* tem a função de aplicar um ganho igual a  $2^{15} - 1$  ao sinal.

O bloco *Digital Filter Design* que pode ser observado na Figura 59 pode ser usado para implementar diferentes filtros digitais (passa-baixas, passa-faixa e passa-altas). Após o início da execução do experimento, os estudantes podem observar os efeitos da filtragem sobre os sinais de áudio na característica de seu espectro (utilizando a função FFT do osciloscópio), e também podem escutar os efeitos da filtragem com o uso de fones de ouvido conectados à saída da caixa de som USB.

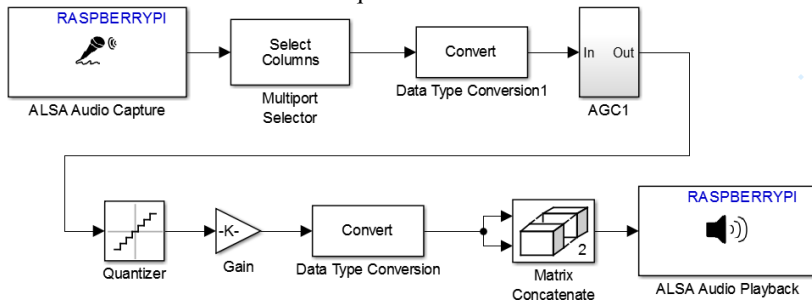
## 4.4 RASPBERRY PI COMO UM QUANTIZADOR

O experimento a ser proposto nesta subsecção tem o objetivo de permitir que os estudantes visualizem o efeito da quantização em sinais reais. Na primeira parte do experimento, o Raspberry Pi será utilizado para realizar a quantização de um sinal senoidal proveniente do gerador de funções. Na segunda parte do experimento, o Raspberry Pi será utilizado para reproduzir gravações de voz salvas em arquivos *.wav*, e nessa etapa os estudantes poderão perceber os efeitos da imitação de frequência nesses sinais. Por fim, na terceira parte do experimento, o Raspberry Pi será utilizado para quantizar as amostras de sinais de áudio provenientes de arquivos *.wav*. Nesse experimento, os efeitos da quantização podem ser observados visualmente com o uso do osciloscópio e também de forma audível ao conectarem-se fones de ouvidos na saída da placa de som USB.

### 4.4.1 Quantização de um sinal senoidal com o Raspberry Pi

O modelo do Simulink que pode ser observado na Figura 60 transforma o Raspberry Pi num quantizador. O sinal a ser quantizado nessa etapa é proveniente do gerador de funções, cuja saída está conectada à entrada analógica da placa de som USB.

Figura 60 – Modelo do Simulink que transforma o Raspberry Pi num quantizador.



Fonte: O autor (2018).

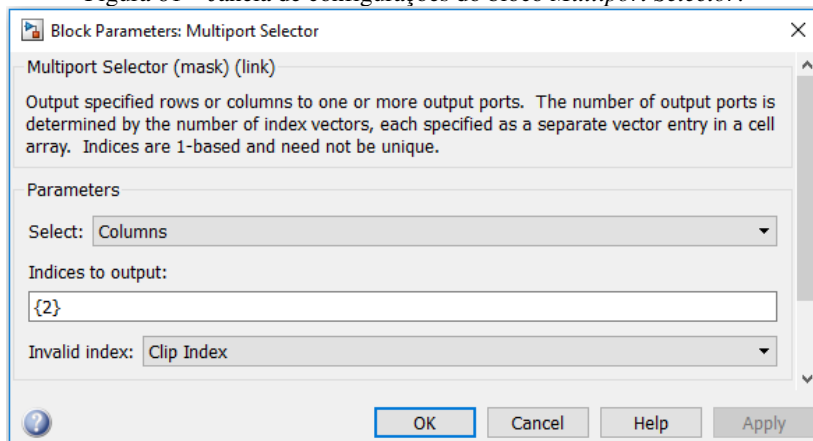
Alguns dos blocos do Simulink que podem ser observados na Figura 60 não foram apresentados anteriormente. Eles serão discutidos brevemente a seguir.

O bloco *Multiport Selector* tem a função de selecionar um dos canais do sinal proveniente do bloco *ALSA Audio Capture*. Em outras



palavras, esse bloco tem a função de converter um sinal de entrada do tipo estéreo em um sinal de saída do tipo mono. A janela de configurações desse bloco pode ser observada na Figura 61. O único parâmetro relevante desse bloco é *Índices to output*, que é definido como 2 com o objetivo de selecionar apenas o segundo canal do sinal de entrada (PASOLINI, BAZZI e MIRABELLA, 2016).

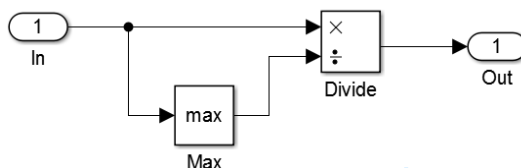
Figura 61 – Janela de configurações do bloco *Multiport Selector*.



Fonte: O autor (2018).

Os componentes internos do subsistema AGC podem ser observados na Figura 62. Esse subsistema tem a função de realizar a normalização do sinal aplicado a sua entrada, isto é, um controle de ganho automático (AGC, do inglês *Automatic Gain Control*). O AGC divide todas as amostras de cada quadro do sinal aplicado à sua entrada pelo máximo valor detectado no quadro, fazendo com que o sinal à sua saída possua amostras com valores entre  $-1$  e  $1$ . Essa normalização é necessária para o correto funcionamento do bloco *Quantizer*.

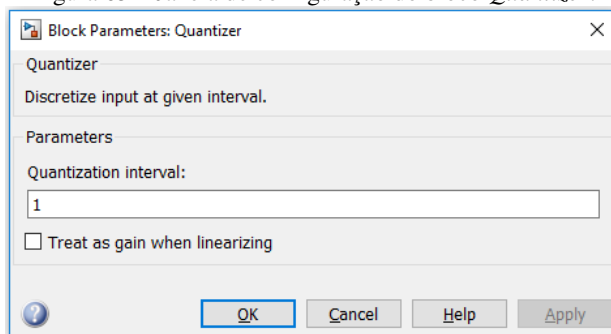
Figura 62 – Componentes internos do subsistema AGC que pode ser observado na Figura 60.



Fonte: O autor (2018).

O bloco *Quantizer* realiza a quantização de um sinal de entrada. Sua janela de configurações pode ser observada na Figura 63. O único parâmetro relevante desse bloco é o intervalo de quantização (*Quantization interval*). A quantização realizada por esse bloco é do tipo *mid-tread*, através da qual o valor de amplitude zero é representado por um dos níveis de quantização.

Figura 63 – Janela de configuração do bloco *Quantizer*.



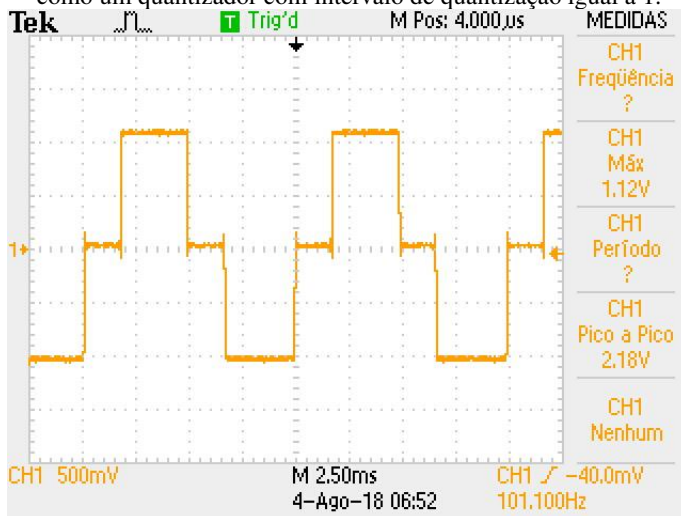
Fonte: O autor (2018).

Por fim, o bloco *Gain* que pode ser observado na Figura 60 tem o objetivo de dar um ganho igual a  $2^{15} - 1$  ao sinal de saída do bloco *Quantizer*.

O experimento que transforma o Raspberry Pi num quantizador pode ser realizado para diferentes valores de intervalos de quantização. A partir de agora serão apresentados alguns resultados observados em algumas realizações do experimento. Em todos os casos o sinal de entrada foi uma onda senoidal com frequência de 100 Hz. Além disso, para que o sinal de saída pudesse ser adequadamente observado no osciloscópio, foi utilizado o acoplamento CC.

Com o uso de um intervalo de quantização igual a 1, o sinal de entrada foi quantizado com o uso de 3 níveis de quantização, conforme pode ser observado na Figura 64.

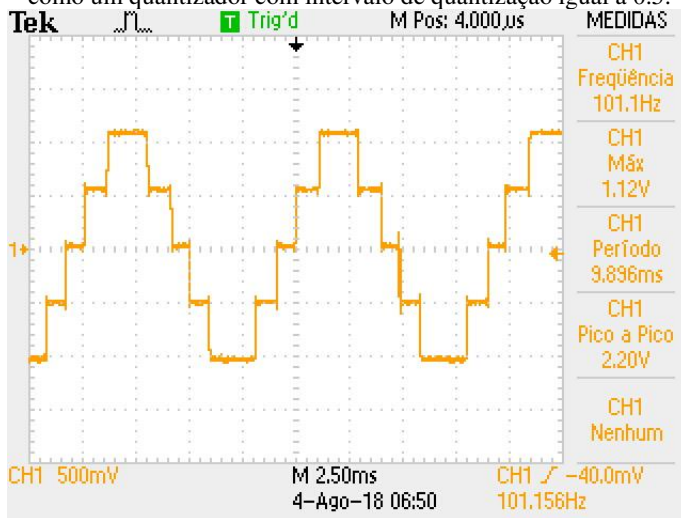
Figura 64 – Sinal de saída observado quando o Raspberry Pi foi configurado como um quantizador com intervalo de quantização igual a 1.



Fonte: O autor (2018).

Com o uso de um intervalo de quantização igual a 0,5, o sinal de entrada foi quantizado com o uso de 5 níveis de quantização, conforme pode ser observado na Figura 65.

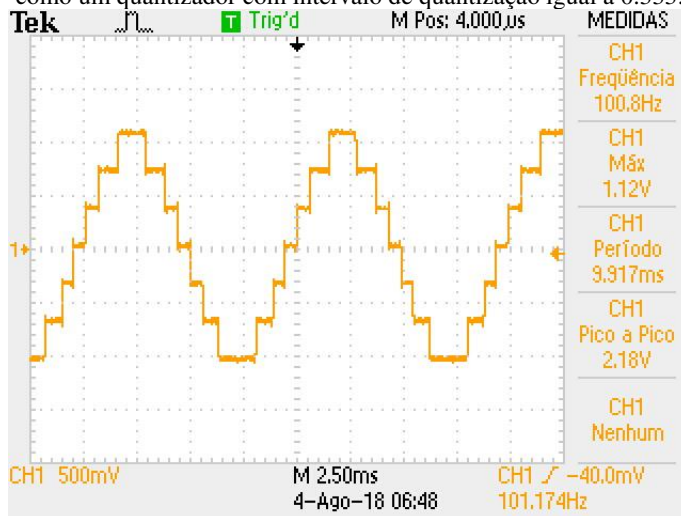
Figura 65 – Sinal de saída observado quando o Raspberry Pi foi configurado como um quantizador com intervalo de quantização igual a 0.5.



Fonte: O autor (2018).

Com o uso de um intervalo de quantização igual a 0.333, o sinal de entrada foi quantizado com o uso de 7 níveis de quantização, conforme pode ser observado na Figura 66.

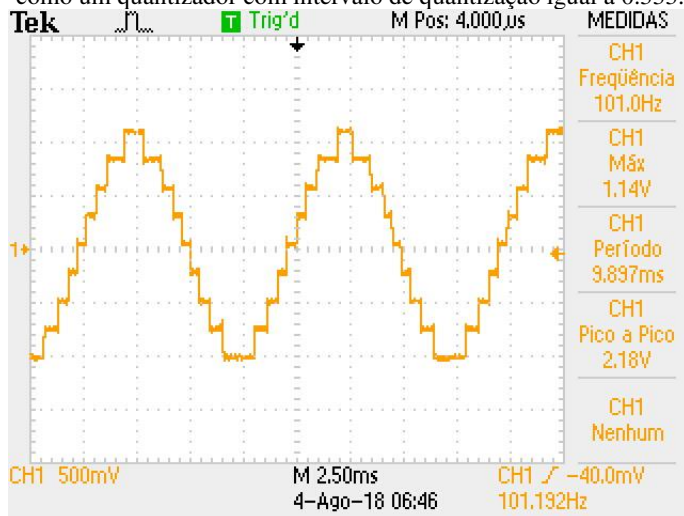
Figura 66 – Sinal de saída observado quando o Raspberry Pi foi configurado como um quantizador com intervalo de quantização igual a 0.333.



Fonte: O autor (2018).

Por fim, com o uso de um intervalo de quantização igual a 0.25, o sinal de entrada foi quantizado com o uso de 9 níveis de quantização, conforme pode ser observado na Figura 67.

Figura 67 – Sinal de saída observado quando o Raspberry Pi foi configurado como um quantizador com intervalo de quantização igual a 0.333.



Fonte: O autor (2018).

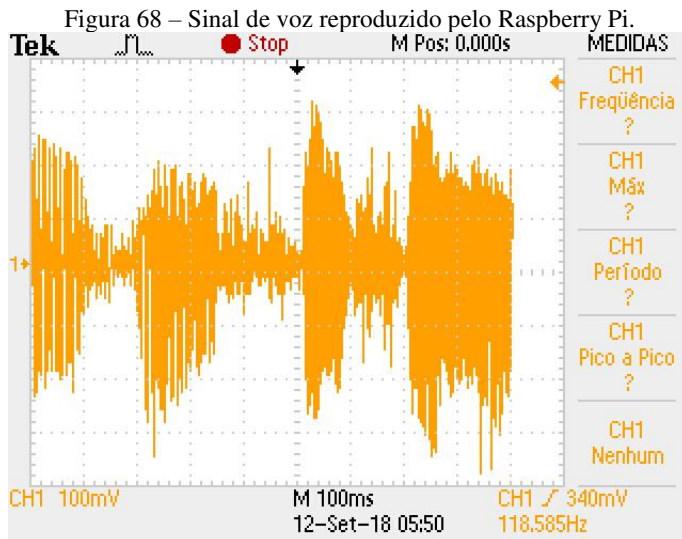
Com o modelo do Simulink sendo executado em modo externo no Raspberry Pi, o valor do intervalo de quantização adotado (parâmetro *Quantization interval* do bloco *Quantizer*) pode ser alterado durante a execução do experimento, e com isso as alterações na forma de onda do sinal de saída podem ser observadas em tempo real.

#### 4.4.2 Reprodução de sinais de áudio com o Raspberry Pi

A segunda etapa deste experimento tem o objetivo de reproduzir gravações de voz armazenadas em arquivos *.wav* na saída da placa de som USB. Dois arquivos de áudio serão utilizados: um contendo uma gravação de voz original (de alta qualidade), com largura de banda de 24 kHz, e outro contendo um sinal de voz cuja largura de banda foi limitada a 3.5 kHz (aproximadamente a largura de banda de canais telefônicos). Ambas os sinais são constituídos de amostras com amplitude normalizada (isto é, com valor máximo unitário).

O modelo do Simulink utilizado para a reprodução dos arquivos de áudio é o mesmo utilizado anteriormente na subseção 4.2.8, e pode ser observado na Figura 48. Da mesma forma, os procedimentos para a importação das amostras do sinal contidas no arquivo *.wav* são exatamente os mesmos.

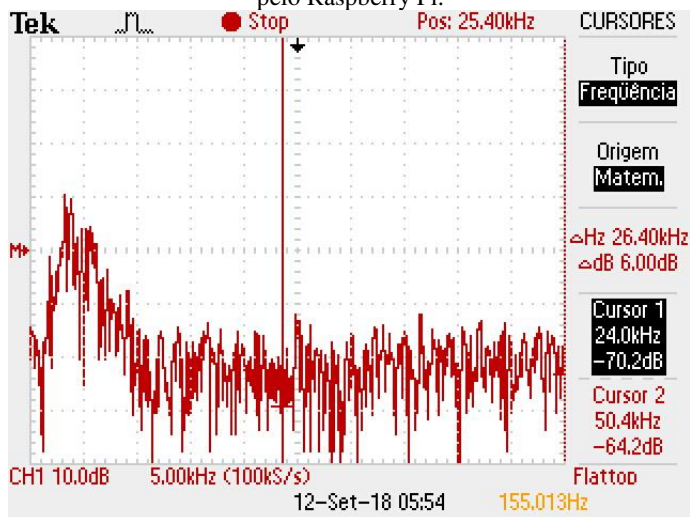
Primeiramente, o modelo do Simulink deve ser configurado para que o Raspberry Pi reproduza a gravação de voz original. Ao conectar-se a saída da placa de som USB na entrada do osciloscópio, é possível visualizar na tela do segundo o sinal mostrado na Figura 68.



Fonte: O autor (2018).

Com o uso da ferramenta matemática FFT do osciloscópio, é possível observar o espectro do sinal de voz original mostrado na Figura 68. Esse espectro, por sua vez, pode ser observado na Figura 69.

Figura 69 – Espectro do sinal de voz (sem limitação de frequência) reproduzido pelo Raspberry Pi.



Fonte: O autor (2018).

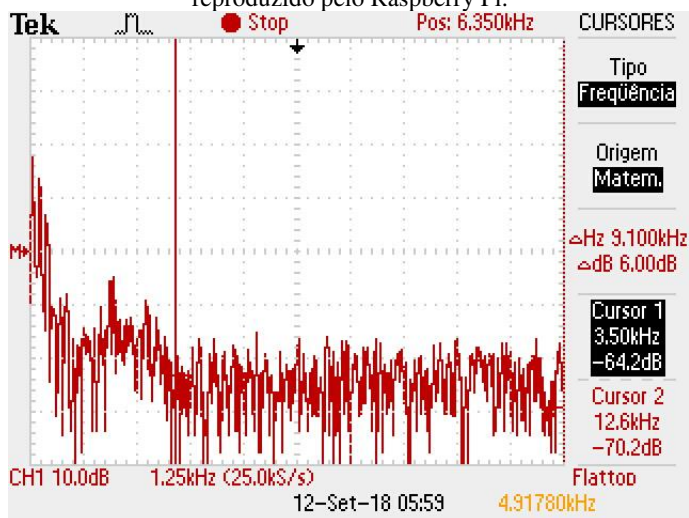
O espectro do sinal de voz observado no osciloscópio não é estático, isto é, sua forma altera-se com o decorrer do tempo. No entanto, eventualmente é possível observar que o sinal de voz original possui componentes em frequências próximas de 20 kHz.

Após a reprodução do sinal de voz original, o modelo do Simulink deve ser configurado para a reprodução do sinal de voz cuja banda de frequência está limitada a 3.5 kHz. Após o início da execução do experimento, é possível observar na tela do osciloscópio um sinal cuja forma é muito semelhante à forma do sinal de voz original, observado anteriormente na Figura 68.

Novamente utilizando a função matemática FFT do osciloscópio, é possível observar o espectro do sinal de voz limitado em frequência. Esse espectro está ilustrado na Figura 70.



Figura 70 – Espectro do sinal de voz (com largura de banda limitada a 3.5 kHz) reproduzido pelo Raspberry Pi.



Fonte: O autor (2018).

Assim como o espectro do sinal de voz original, o espectro do sinal de voz limitado em frequência a 3.5 kHz não é estático, ou seja, sua forma altera-se com o decorrer do tempo.

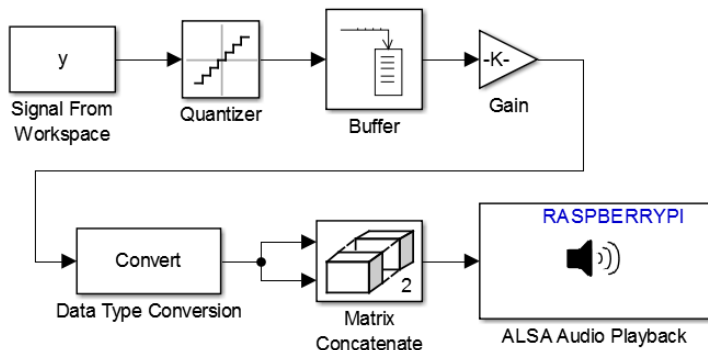
Além de os estudantes analisarem os sinais de voz com o uso do osciloscópio, eles podem utilizar os fones de ouvido conectados à saída analógica da placa de som USB para comparar a qualidade das duas gravações. É possível notar que o sinal original apresenta uma qualidade muito superior à do sinal limitado em frequência, sendo que a qualidade da segunda é muito semelhante à de sinais de voz em canais telefônicos. Todavia, também é possível concluir que, apesar da limitação de banda em 3.5 kHz, o sinal é inteligível. Em outras palavras, a maior parte da potência do sinal de voz concentra-se nas baixas frequências, de modo que a informação transmitida ainda é preservada.

#### 4.4.3 Quantização de sinais de áudio com o Raspberry Pi

O modelo do Simulink que pode ser observado na Figura 71 é utilizado para transformar o Raspberry Pi num quantizador de sinais de áudio. Esses sinais a serem quantizados são os mesmos arquivos utilizados na etapa anterior, isto é, provenientes de arquivos .wav.

Alternativamente, arquivos de música também salvos em formato `.wav` pode ser utilizados nessa etapa do experimento.

Figura 71 – Modelo do Simulink utilizado para transformar o Raspberry Pi num quantizador de sinais de áudio.



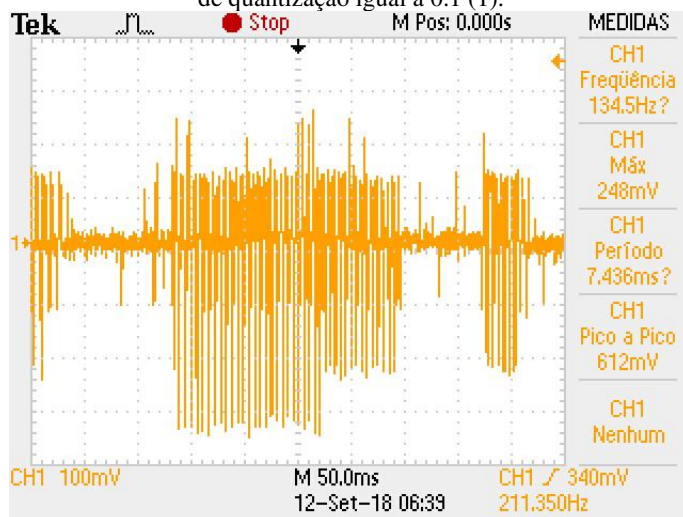
Fonte: O autor (2018).

O bloco *Signal from Workspace*, utilizado para importar as amostras armazenadas numa variável do *Workspace* do MATLAB, deve ser configurado com os seguintes parâmetros: *Sample time* (período de amostra) igual a  $1/48000$  s e *Samples per frame* igual a 1. Além disso, o bloco deve ser configurado para realizar repetição cíclica do sinal reproduzido.

O bloco *Quantizer* que pode ser observado na Figura 71 é exatamente igual ao utilizado anteriormente na quantização do sinal senoidal. A sequência de amostras à saída desse bloco é organizada em quadros de 1024 amostras pelo bloco *Buffer*. O bloco *Gain* que pode ser observado nesse modelo é utilizado para aplicar um ganho de valor  $2^{15} - 1$  ao sinal a ser reproduzido pela placa de som USB.

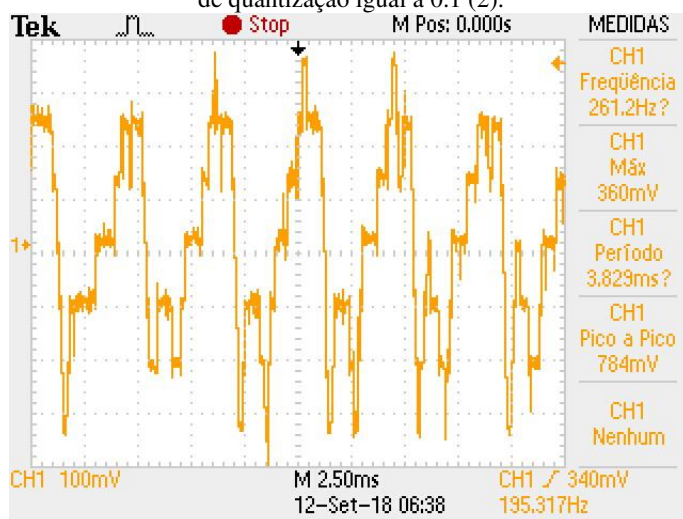
Após o início da execução do experimento, a saída da placa de som USB pode ser conectada à entrada do osciloscópio para que o sinal de voz quantizado possa ser observado no domínio do tempo. Na Figura 72 e na Figura 73, a forma de onda do sinal de voz quantizado com o uso de um intervalo de quantização igual a 0.1 pode ser observada.

Figura 72 – Sinal de voz quantizado pelo Raspberry Pi com o uso de intervalo de quantização igual a 0.1 (1).



Fonte: O autor (2018).

Figura 73 – Sinal de voz quantizado pelo Raspberry Pi com o uso de intervalo de quantização igual a 0.1 (2).



Fonte: O autor (2018).

A saída da placa de som USB também pode ser conectada a fones de ouvido para que os estudantes possam escutar os efeitos do ruído de quantização na inteligibilidade do sinal de voz. Da mesma maneira realizada anteriormente para a quantização do sinal senoidal, o intervalo de quantização (parâmetro *Quantization interval* do bloco *Quantizer*) pode ser alterado em tempo real durante a execução do experimento em modo externo. Dessa forma, é possível observar em tempo real a degradação da qualidade do sinal de áudio de forma proporcional ao intervalo de quantização utilizado.

## 4.5 MODULAÇÕES EM AMPLITUDE COM O RASPBERRY PI

### 4.5.1 Raspberry Pi como um modulador DSB-SC

Este experimento tem como objetivo gerar na saída analógica da placa de som USB um sinal modulado DSB-SC. Para fins didáticos, o primeiro sinal de mensagem a ser utilizado será o sinal senoidal  $m(t) = \cos(2\pi f_m t)$ . A Transformada de Fourier desse sinal é:

$$M(f) = \frac{1}{2} (\delta(f - f_m) + \delta(f + f_m)).$$

Assim, o espectro do sinal de mensagem será constituído simplesmente por um impulso na frequência  $-f_m$  e outro impulso na frequência  $f_m$ .

Para a modulação DSB-SC, o sinal modulado é dado por:

$$s(t) = m(t) \cos(2\pi f_c t).$$

Na expressão anterior,  $m(t)$  corresponde ao sinal de mensagem, e o termo  $\cos(2\pi f_c t)$  corresponde à portadora senoidal com frequência  $f_c$ .

Considerando o sinal de mensagem apresentado anteriormente, o sinal modulado é dado por:

$$s(t) = \cos(2\pi f_m t) \cos(2\pi f_c t).$$

A Transformada de Fourier do sinal modulado é dada por:

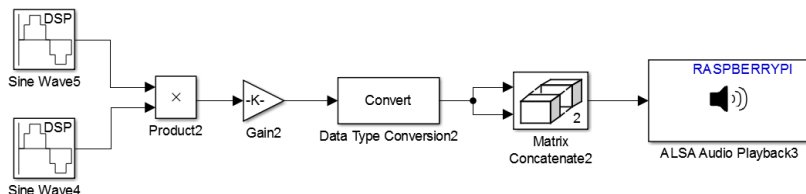
$$S(f) = \frac{1}{4} \left( \begin{array}{l} \delta(f - (f_c - f_m)) + \delta(f - (f_c + f_m)) \\ \delta(f + (f_c - f_m)) + \delta(f + (f_c + f_m)) \end{array} \right).$$

Assim, o espectro do sinal modulado será constituído impulsos nas frequências  $-(f_c - f_m)$ ,  $-(f_c + f_m)$ ,  $(f_c - f_m)$  e  $(f_c + f_m)$ . Em outras palavras, a modulação DSB-SC desloca o espectro em banda base

do sinal de mensagem para uma banda em altas frequências centralizada na frequência da portadora.

O modelo do Simulink que pode ser observado na Figura 74 gera saída da placa de som USB o sinal modulado DSB-SC apresentado anteriormente. Nesse modelo, o bloco *Sine Wave* superior foi configurado para gerar o sinal de mensagem, isto é, o sinal senoidal com frequência  $f_m$ . Já o bloco *Sine Wave* inferior foi configurado para gerar a portadora, isto é, um sinal senoidal com frequência  $f_c$ .

Figura 74 – Modelo do Simulink que gera um sinal modulado DSB-SC.

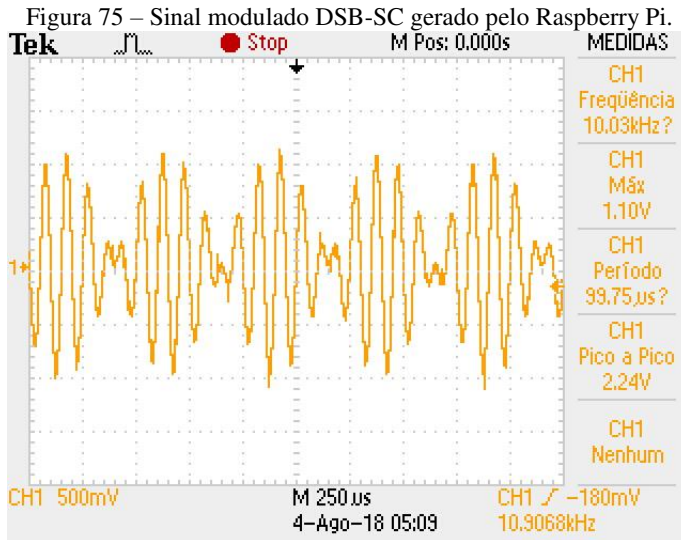


Fonte: O autor (2018).

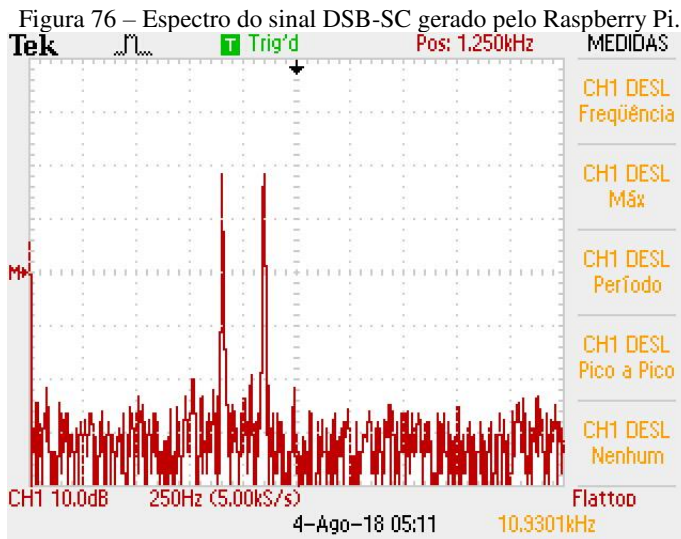
Para a execução do primeiro experimento, foram adotados os parâmetros  $f_m = 1$  kHz e  $f_c = 10$  kHz. Além disso, os dois blocos *Sine Wave* que pode ser observados na Figura 74 foram configurados para gerar sinais com amplitude unitária, amostras agrupadas em quadros de tamanho  $N = 1024$  amostras e taxa de amostragem de 48 kHz.

Os demais blocos que podem ser observados na Figura 74 já foram apresentados nos experimentos anteriores. Cabe lembrar que o valor do ganho aplicado pelo bloco *Gain* deve ser de  $2^{15} - 1$ .

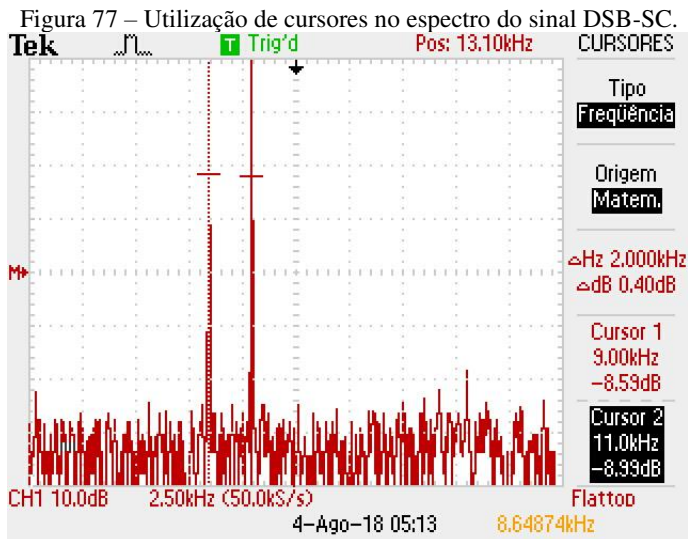
Após a execução do modelo do Simulink que gera um sinal modulado DSB-SC, foi possível observar na tela do osciloscópio o sinal mostrado na Figura 75.



Utilizando a função matemática FFT do osciloscópio, foi possível observar o espectro (unilateral) do sinal mostrado na Figura 75. Esse espectro pode ser observado na Figura 76.



Com a utilização de cursores, foi possível mensurar a magnitude e a frequência dos dois impulsos que podem ser observados no espectro da Figura 76. Esses valores podem ser observados na Figura 77.



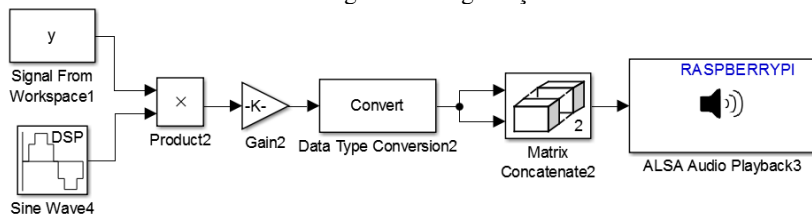
Fonte: O autor (2018).

Conforme era esperado, o espectro (unilateral) do sinal modulado DSB-SC gerado pelo Raspberry Pi consiste de duas componentes impulsivas, uma na frequência  $f_c - f_m = 9$  kHz e outra na frequência  $f_c + f_m = 11$  kHz.

Após a realização da modulação DSB-SC com o sinal de mensagem senoidal, o Raspberry Pi também pode ser utilizado para realizar o mesmo tipo de modulação tendo como sinal de mensagem um sinal de voz. O sinal a ser utilizado nessa etapa é a gravação de voz cuja largura de banda é limitada a 3.5 kHz.

O modelo do Simulink que pode ser observado na Figura 78 é utilizado para realizar a modulação DSB-SC do sinal de voz limitado em frequência. A única diferença desse modelo em comparação com o utilizado anteriormente na modulação com o sinal de mensagem senoidal (mostrado na Figura 74) consiste na substituição do bloco *Sine Wave* pelo bloco *Signal From Workspace*.

Figura 78 – Modelo do Simulink que gera um sinal modulado DSB-SC cujo sinal de mensagem é uma gravação de voz.



Fonte: O autor (2018).

As amostras do sinal de voz contidas num arquivo `.wav` devem ser salvas numa variável do *Workspace* do MATLAB seguindo os mesmos procedimentos apresentados na subseção 4.2.8. Essas amostras são importadas para o modelo do Simulink pelo bloco *Signal from Workspace*, que deve ser configurado com os seguintes parâmetros: *Sample time* (período de amostra) igual a  $1/48000$  s e *Samples per frame* igual a 1024 (com isso, as amostras do sinal são organizadas em quadros de 1024 amostras). Além disso, o bloco deve ser configurado para realizar a repetição cíclica do sinal.

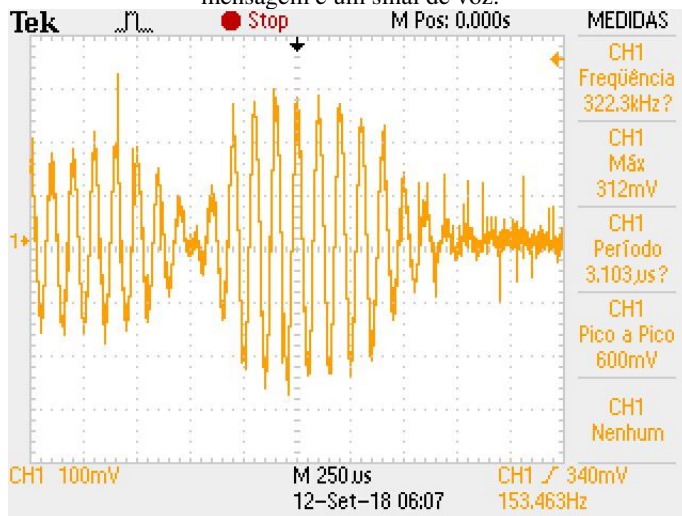
O bloco *Sine Wave* que pode ser observado na Figura 78 gera uma portadora senoidal de 10 kHz, com amostras a uma taxa de amostragem de 48 kHz e organizadas em quadros de 1024 amostras.

O ganho aplicado ao sinal modulado DSB-SC pelo bloco *Gain* deve ser configurado como  $2^{15} - 1$ .

Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 79.



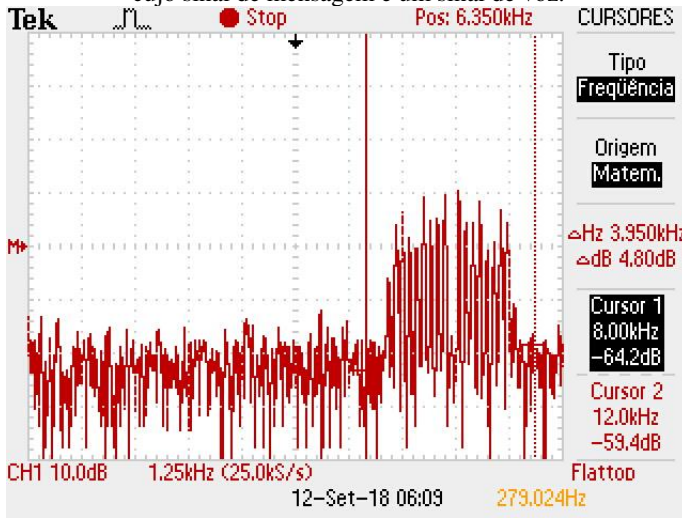
Figura 79 – Sinal modulado DSB-SC gerado pelo Raspberry Pi, cujo sinal de mensagem é um sinal de voz.



Fonte: O autor (2018).

Utilizando a função matemática FFT do osciloscópio, é possível observar o espectro do sinal DSB-SC mostrado na Figura 79. Esse espectro, por sua vez, pode ser observado na Figura 80.

Figura 80 – Espectro do sinal modulado DSB-SC gerado pelo Raspberry Pi, cujo sinal de mensagem é um sinal de voz.



Fonte: O autor (2018).

O espectro mostrado na Figura 80 corresponde ao espectro do sinal de voz (originalmente banda base) deslocado para banda passante. O espectro resultante está centralizado em 10 kHz (a frequência da portadora senoidal) e possui largura de banda igual a 7 kHz (ou seja, o dobro da largura de banda do sinal de voz em banda base). É interessante notar que, pelo fato de o sinal de voz em banda base não possuir componentes significativas em torno de 0 Hz, o sinal modulado não possui componentes significativas em torno de 10 kHz.

#### 4.5.2 Raspberry Pi como um modulador AM

O objetivo deste experimento é gerar na saída analógica da placa de som USB um sinal modulado AM. Assim como no experimento de geração de um sinal DSB-SC, o primeiro sinal de mensagem a ser utilizado será o sinal senoidal  $m(t) = \cos(2\pi f_m t)$ .

A expressão do sinal modulado AM é dada por:

$$s(t) = (a + m(t)) \cos(2\pi f_c t) = a \cos(2\pi f_c t) + m(t) \cos(2\pi f_c t).$$

Na expressão anterior  $a$  é uma constante (nível DC) adicionada ao sinal de mensagem e  $\cos(2\pi f_c t)$  corresponde à portadora senoidal.

Considerando o sinal de mensagem senoidal, a expressão do sinal modulado AM é dada por:

$$s(t) = a \cos(2\pi f_c t) + \cos(2\pi f_m t) \cos(2\pi f_c t).$$

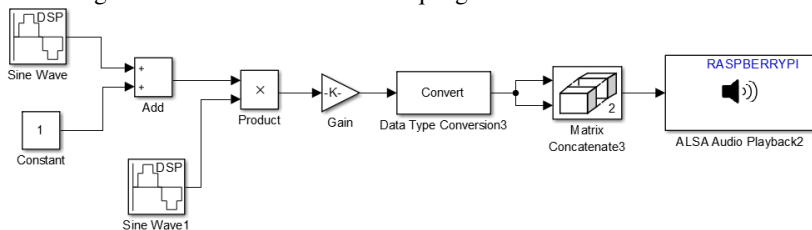
A Transformada de Fourier do sinal modulado é então dada por:

$$S(f) = \frac{a}{2}(\delta(f - f_c)) + \frac{a}{2}(\delta(f + f_c)) + \frac{1}{4}(\delta(f - (f_c - f_m))) + \frac{1}{4}(\delta(f + (f_c - f_m))) + \frac{1}{4}(\delta(f - (f_c + f_m))) + \frac{1}{4}(\delta(f + (f_c + f_m))).$$

Analisando a expressão anterior, podemos observar que o espectro do sinal modulado AM é muito semelhante ao espectro do sinal modulado DSB-SC. A única diferença consiste na adição componentes impulsivas nas frequências  $-f_c$  e  $f_c$ . Essas componentes correspondem à portadora senoidal cuja amplitude é modulada pelo sinal  $m(t)$ .

O modelo do Simulink que pode ser observado na Figura 81 gera na saída da placa de som USB o sinal modulado AM apresentado anteriormente. Nesse modelo, o bloco *Sine Wave* superior foi configurado para gerar o sinal de mensagem, isto é, o sinal senoidal com frequência  $f_m$ . A esse sinal de mensagem é adicionado um nível DC unitário. Já o bloco *Sine Wave* inferior foi configurado para gerar a portadora, isto é, um sinal senoidal com frequência  $f_c$ .

Figura 81 – Modelo do Simulink que gera um sinal modulado AM.



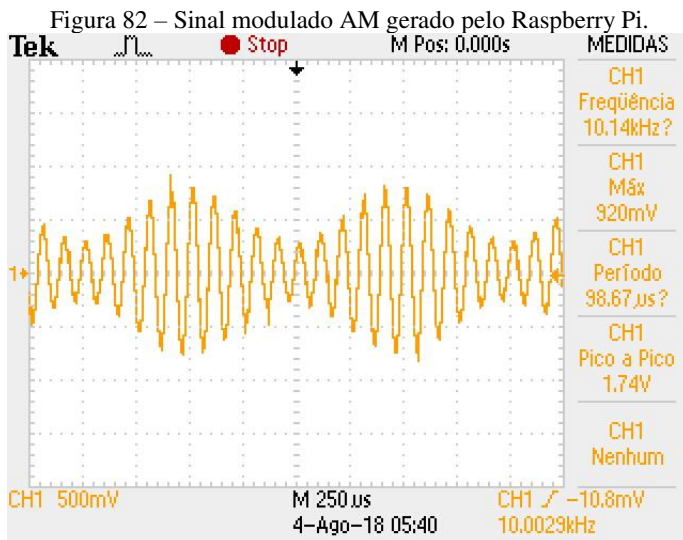
Fonte: O autor (2018).

Para a execução do experimento, o sinal de mensagem  $m(t)$  utilizado é uma senóide com amplitude 0.5 e frequência  $f_m = 1$  kHz. Já a portadora é uma senóide com amplitude unitária de frequência  $f_c = 10$  kHz. Assim como no experimento anterior, os dois blocos *Sine Wave*

que pode ser observados na Figura 81 foram configurados para gerar sinais com amostras agrupadas em quadros de tamanho  $N = 1024$  amostras e taxa de amostragem de 48 kHz.

Os demais blocos que podem ser observados na Figura 81 também foram utilizados no experimento de geração do sinal DSB-SC. Porém, para a geração adequada do sinal modulado AM, o valor de ganho a ser utilizado deve ser de  $2^{14} - 1$ . O valor de ganho igual a  $2^{15} - 1$  faz com que o sinal AM na saída da placa de som USB seja distorcido.

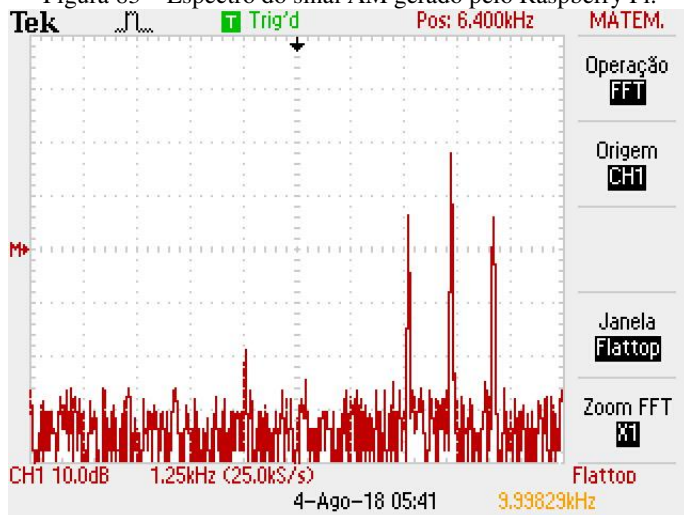
Após a execução do modelo do Simulink que gera um sinal modulado AM, foi possível observar na tela do osciloscópio o sinal mostrado na Figura 82.



Fonte: O autor (2018).

Utilizando a função matemática FFT do osciloscópio, foi possível observar o espectro (unilateral) do sinal mostrado na Figura 82. Esse espectro pode ser observado na Figura 83.

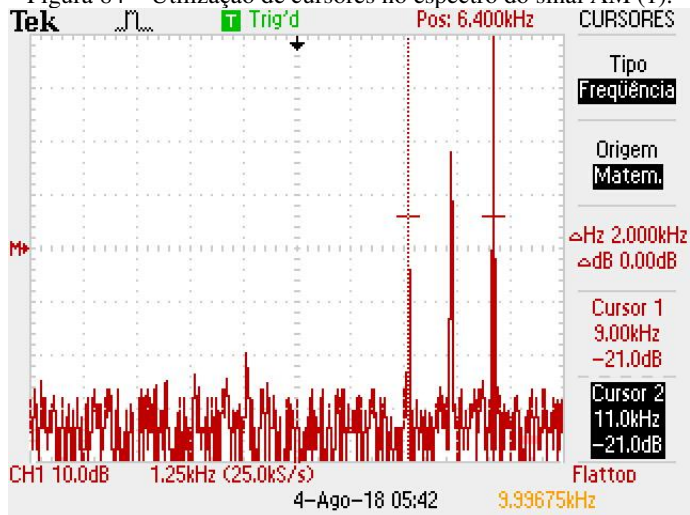
Figura 83 – Espectro do sinal AM gerado pelo Raspberry Pi.



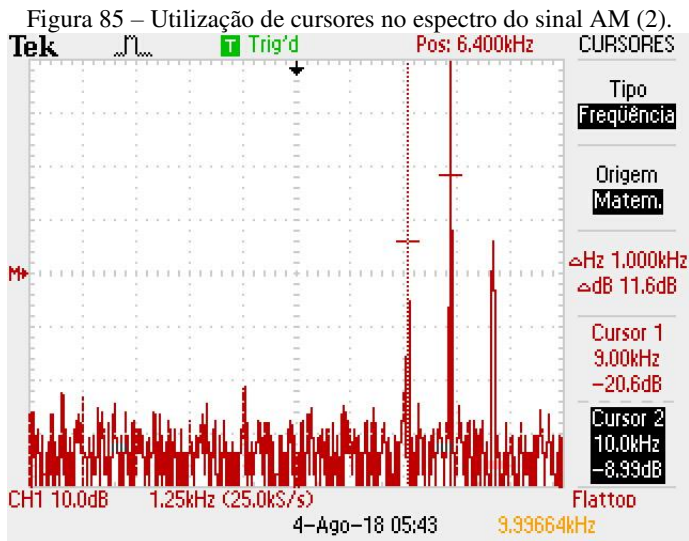
Fonte: O autor (2018).

Com a utilização de cursores, foi possível mensurar a magnitude e a frequência dos três impulsos que podem ser observados no espectro. Esses valores podem ser observados na Figura 84 e na Figura 85.

Figura 84 – Utilização de cursores no espectro do sinal AM (1).



Fonte: O autor (2018).

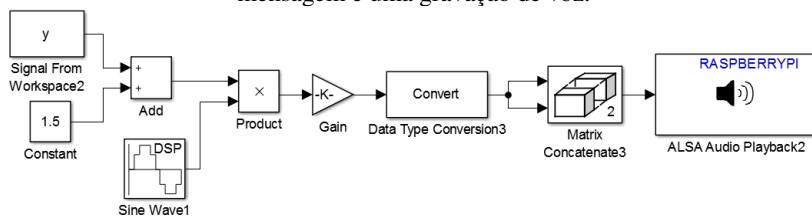


Conforme esperado, o espectro (unilateral) do sinal AM consiste em três componentes impulsivas. As componentes nas frequências  $f_c - f_m = 9$  kHz e  $f_c + f_m = 11$  kHz correspondem ao espectro do sinal  $m(t)$  em banda passante. Já a componente em  $f_c = 10$  kHz corresponde à portadora senoidal contida no sinal modulado AM.

Após a realização da modulação AM com o sinal de mensagem senoidal, o Raspberry Pi também pode ser utilizado para realizar o mesmo tipo de modulação tendo como sinal de mensagem um sinal de voz. Novamente, o sinal a ser utilizado nessa etapa é a gravação de voz cuja largura de banda é limitada a 3.5 kHz.

O modelo do Simulink que pode ser observado na Figura 86 é utilizado para transformar o Raspberry Pi num gerador de um sinal modulado AM cujo sinal de mensagem é a gravação de voz. Esse modelo é uma adaptação do modelo utilizado anteriormente para a modulação DSB-SC, que pode ser observado na Figura 78. A única diferença entre os dois modelos consiste no fato de que, no modulador AM, um nível DC (constante) é adicionado ao sinal de mensagem. Além disso, o valor do ganho aplicado ao sinal modulado pelo bloco *Gain* deve ser definido como  $2^{13} - 1$  com o objetivo de evitar a distorção do sinal à saída da placa de som USB.

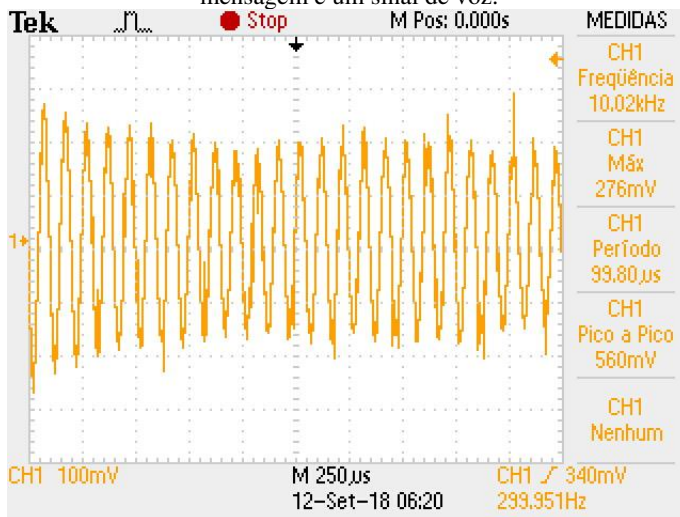
Figura 86 – Modelo do Simulink que gera um sinal modulado AM cujo sinal de mensagem é uma gravação de voz.



Fonte: O autor (2018).

Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 87.

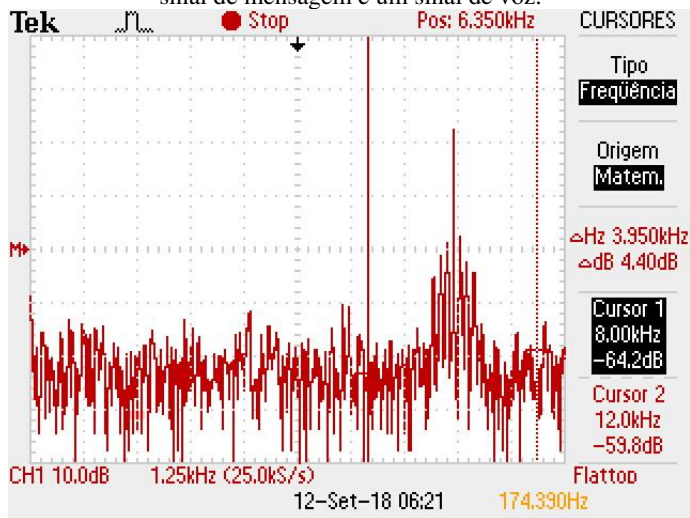
Figura 87 – Sinal modulado AM gerado pelo Raspberry Pi, cujo sinal de mensagem é um sinal de voz.



Fonte: O autor (2018).

Utilizando a função matemática FFT do osciloscópio, é possível observar o espectro do sinal AM mostrado na Figura 87. Esse espectro, por sua vez, pode ser observado na Figura 88.

Figura 88 – Espectro do sinal modulado AM gerado pelo Raspberry Pi, cujo sinal de mensagem é um sinal de voz.



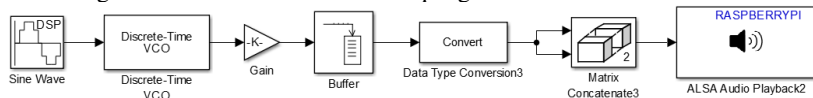
Fonte: O autor (2018).

O espectro do sinal AM modulado não é estático, ou seja, sua forma altera-se com o decorrer do tempo. Porém, na Figura 88 é possível observar que, diferentemente do sinal DSB-SC e conforme o esperado, o sinal AM possui uma componente impulsiva na frequência de 10 kHz (frequência da portadora senoidal). Por outro lado, assim como o espectro do sinal modulado DSB-SC, o sinal modulado AM está centrado em torno de 10 kHz e possui largura de banda de 7 kHz (que é o dobro da largura de banda do sinal de voz em banda base).

#### 4.6 MODULAÇÃO EM FREQUÊNCIA COM O RASPBERRY PI

O objetivo deste experimento é gerar um sinal modulado em frequência na saída da placa de som USB. O modelo do Simulink que pode ser observado na Figura 89 transforma o Raspberry Pi num modulador FM.

Figura 89 – Modelo do Simulink que gera um sinal modulado FM.



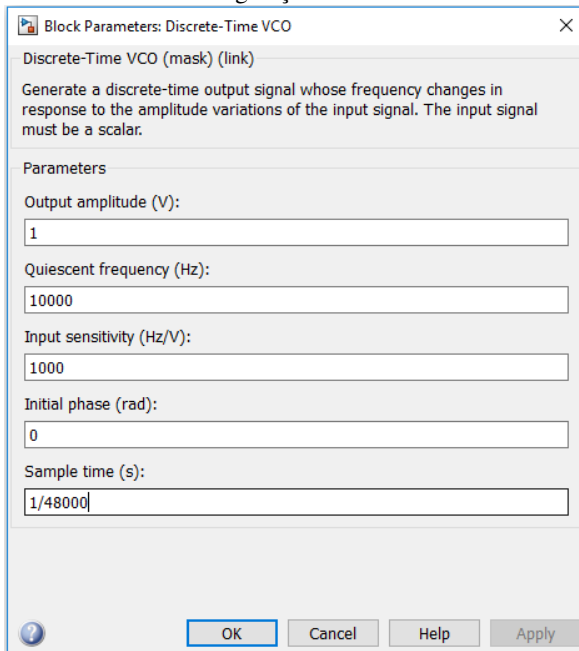
Fonte: O autor (2018).



Para fins didáticos, o primeiro sinal de mensagem a ser utilizado neste experimento é o sinal senoidal. Esse sinal é gerado pelo bloco *Sine Wave* que pode ser observado na Figura 89. Para a execução do primeiro experimento, esse bloco foi configurado para gerar uma senóide discreta com amplitude unitária, frequência de 500 Hz e taxa de amostragem de 48 kHz. Além disso, para a correta utilização do bloco *VCO* (que será explicado a seguir), o parâmetro *Samples per frame* foi definido como 1.

A janela de configurações do bloco *Discrete-Time VCO* pode ser observada na Figura 90. Esse bloco corresponde a um oscilador controlado por tensão que opera em tempo discreto. Sua função é gerar um sinal de saída senoidal cuja frequência varia em resposta a variações de amplitude do sinal de entrada.

Figura 90 – Janela de configurações do bloco *Discrete-Time VCO*.

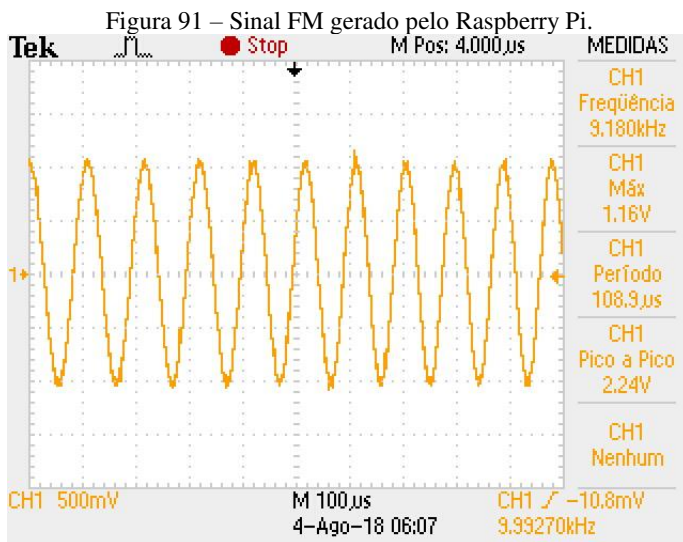


Fonte: O autor (2018).

O bloco *Discrete-Time VCO*, inicialmente, foi configurado com os seguintes parâmetros: amplitude do sinal de saída unitária, frequência quiescente ( $f_c$ ) de 10 kHz, sensibilidade de frequência ( $k_f$ ) de 1 kHz/V e taxa de amostragem de 48 kHz.

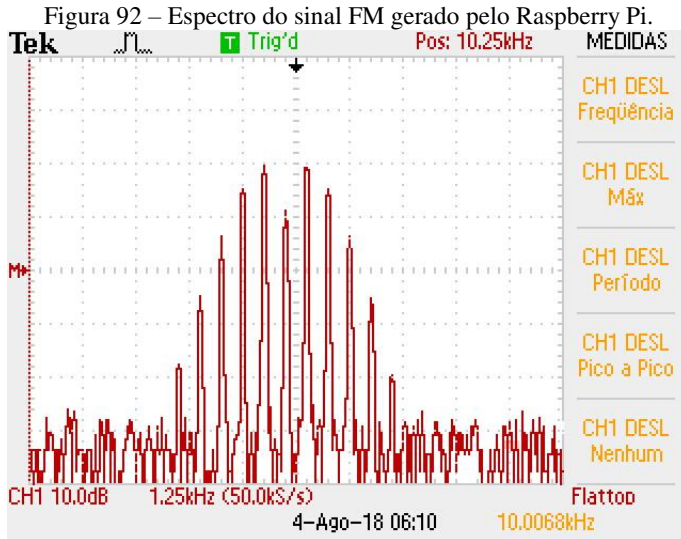
O valor de pico do sinal de mensagem senoidal é unitário. Dessa forma, o desvio de frequência do sinal FM será de  $\Delta f = k_f m_p = 1 \text{ kHz}$ . Portanto, o valor de frequência máximo do sinal modulado será  $f_{max} = f_c + \Delta f = 11 \text{ kHz}$ , enquanto que o valor mínimo de frequência do sinal modulado será  $f_{min} = f_c - \Delta f = 9 \text{ kHz}$ .

Após a execução do modelo do Simulink mostrado na Figura 89, foi possível observar na tela do osciloscópio um sinal senoidal com amplitude constante e frequência variante no tempo. Na Figura 91 é possível observar uma captura de tela do osciloscópio com o experimento em execução, porém nessa imagem estática não é possível perceber a variação na frequência da senóide.



Fonte: O autor (2018).

Utilizando a função matemática FFT do osciloscópio, foi possível observar o espectro (unilateral) do sinal FM gerado pelo Raspberry Pi. Esse espectro pode ser observado na Figura 92.



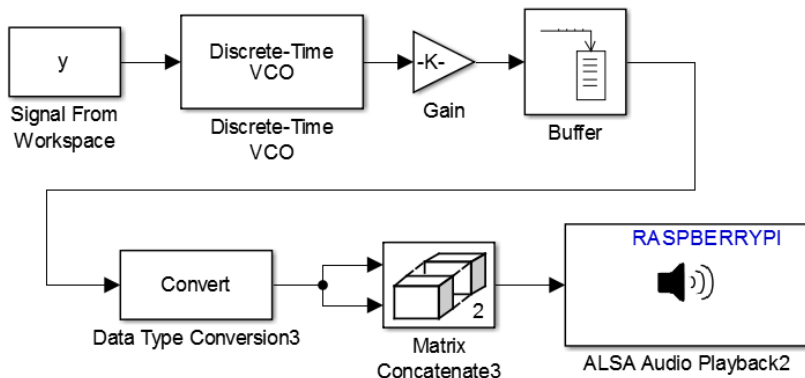
Fonte: O autor (2018).

Com a utilização de cursores, é possível estimar a largura de banda do espectro do sinal FM gerado pelo Raspberry Pi. Além disso, em diferentes execuções do experimento, é interessante realizar alterações na amplitude do sinal de mensagem, na largura de banda do sinal de mensagem (nesse caso, na frequência  $f_m$ ) e na sensibilidade de frequência do VCO e observar os efeitos no espectro do sinal FM.

Após a geração de um sinal FM utilizando o sinal de mensagem senoidal, é possível gerar também um sinal FM utilizando como sinal de mensagem o sinal de voz com largura de banda limitada a 3.5 kHz (o mesmo sinal utilizado nos experimentos de modulação em amplitude).

O modelo do Simulink que pode ser observado na Figura 93 é utilizado para transformar o Raspberry Pi num gerador de um sinal modulado FM cujo sinal de mensagem é o sinal de voz. A única diferença desse modelo em comparação com o modelo mostrado na Figura 89 consiste na substituição do bloco *Sine Wave* pelo bloco *Signal From Workspace*.

Figura 93 – Modelo do Simulink que transforma o Raspberry Pi num modulador FM cujo sinal de mensagem é um sinal de voz.

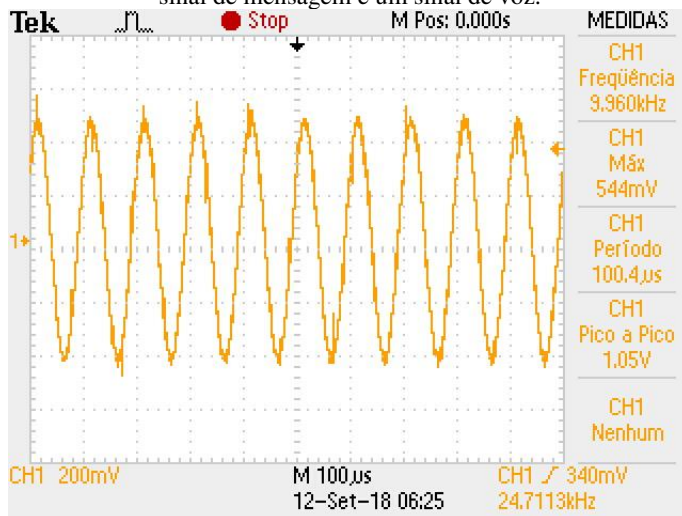


Fonte: O autor (2018).

As amostras do sinal de voz contidas num arquivo `.wav` devem ser salvas numa variável do *Workspace* do MATLAB seguindo os mesmos procedimentos apresentados na subseção 4.2.8. Essas amostras são importadas para o modelo do Simulink pelo bloco *Signal from Workspace*, que deve ser configurado com os seguintes parâmetros: *Sample time* (período de amostra) igual a  $1/48000$  s e *Samples per frame* igual a 1. Além disso, o bloco deve ser configurado para realizar a repetição cíclica do sinal.

Após o início da execução do experimento, é possível visualizar na tela do osciloscópio o sinal mostrado na Figura 94. Assim como o sinal FM gerado com o sinal de mensagem senoidal, o sinal FM cujo sinal de mensagem é o sinal de voz é um sinal senoidal de amplitude constante e cuja frequência varia no decorrer do tempo. As variações de frequência não podem ser observadas na Figura 94 visto que trata-se de uma ilustração estática.

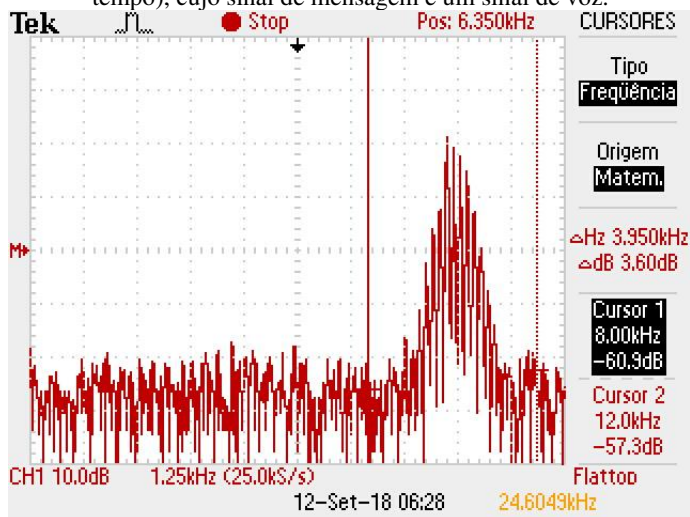
Figura 94 – Sinal FM gerado pelo Raspberry Pi (no domínio do tempo), cujo sinal de mensagem é um sinal de voz.



Fonte: O autor (2018).

Com o uso da ferramenta matemática FFT do osciloscópio, é possível observar o espectro do sinal FM mostrado na Figura 94. Esse espectro, por sua vez, pode ser observado na Figura 95.

Figura 95 – Espectro do sinal FM gerado pelo Raspberry Pi (no domínio do tempo), cujo sinal de mensagem é um sinal de voz.



Fonte: O autor (2018).

O espectro do sinal FM é centralizado na frequência de 10 kHz, que é a frequência quiescente do VCO. Além disso, sua forma não é estática, ou seja, altera-se com o decorrer do tempo. Ao observar-se o espectro durante a execução da simulação, é possível observar que o sinal FM possui largura de banda maior do que 7 kHz, que é a largura de banda dos sinais modulados DSB-SC e AM tendo o mesmo sinal de mensagem.

Como sugestão de experimento a ser realizado em aulas práticas, os estudantes podem realizar a modulação FM de um sinal de voz utilizando diferentes valores de  $k_f$ . Para cada valor, os estudantes devem mensurar a largura de banda do sinal modulado com o uso de cursores no espectro e comparar o valor mensurado com o valor teórico aproximado dado pela Regra de Carson.

## 4.7 MODULAÇÕES EM BANDA BASE COM O RASPBERRY PI

Neste experimento, o Raspberry Pi será utilizado para realizar diferentes modulações digitais em banda base do tipo PAM (*Pulse Amplitude Modulation*). Os modelos do Simulink utilizados neste experimento são adaptações dos modelos propostos no trabalho de Pasolini, Bazzi e Mirabella (2016).

Uma modulação M-PAM genérica associa a cada símbolo  $a_i$  do alfabeto M-ário uma forma de onda em banda base  $g(t)$ , gerando o sinal que é representado pela seguinte equação:

$$s(t) = \sum_{i=-\infty}^{\infty} a_i g(t - iT).$$

Na expressão anterior,  $g(t)$  é o pulso de transmissão e  $T$  é o intervalo de tempo entre um símbolo e o seguinte (isto é, o período de cada símbolo).

Além disso, cabe lembrar que cada símbolo  $a_i$  do alfabeto M-ário representa um número de bits dado por  $b_{símbolo} = \log_2 M$ . Assim, na modulação 2-PAM, os bits  $\{0,1\}$  a serem transmitidos são mapeados, por exemplo, nos símbolos  $a_i \in \{-1, +1\}$ . Já na modulação 4-PAM, as duplas de bits  $\{"00", "01", "10", "11"\}$  são mapeadas, por exemplo, nos símbolos  $a_i \in \{-1, -1/3, +1/3, +1\}$ .

Primeiramente, as modulações em banda base serão implementadas com o uso de pulsos quadrados. Em seguida, as modulações em banda base serão implementadas com o uso de pulsos do tipo cosseno levantado.

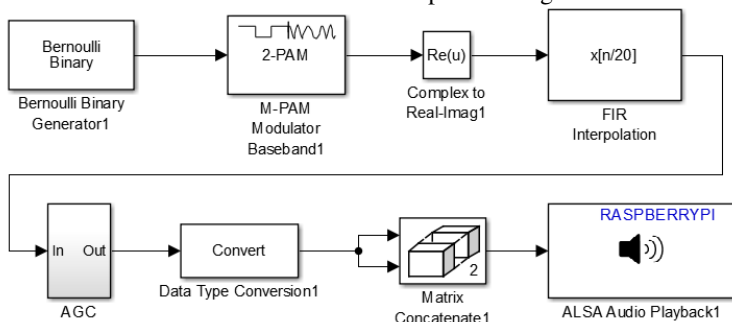
### 4.7.1 Modulação 2-PAM com pulso de transmissão retangular NRZ

O primeiro pulso de transmissão a ser utilizado neste experimento é o pulso retangular sem retorno a zero (NRZ, do inglês *non-return-to-zero*). A expressão (no domínio do tempo) desse pulso é dada por:

$$g(t) = \text{rect}\left(\frac{t - T/2}{T}\right) = \begin{cases} 1, & 0 \leq t \leq T \\ 0, & \text{caso contrário} \end{cases}$$

O modelo do Simulink que pode ser observado na Figura 96 transforma o Raspberry Pi num modulador em banda base 2-PAM que utiliza pulsos quadrados.

Figura 96 – Modelo do Simulink que transforma o Raspberry Pi num modulador em banda base 2-PAM com pulso retangular.

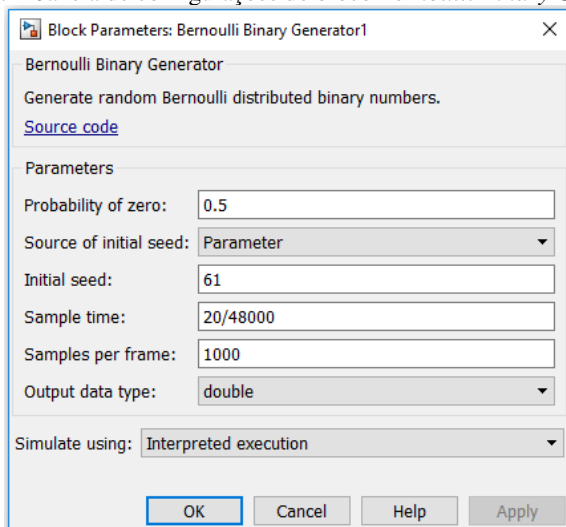


Fonte: O autor (2018).

Alguns dos blocos do Simulink que podem ser observados na Figura 96 não foram utilizados nos experimentos anteriores. A seguir, serão apresentadas algumas explicações sobre seu funcionamento.

O bloco *Bernoulli Binary Generator* tem a função de gerar uma sequência aleatória de bits com iguais probabilidades de ocorrência de 0's e 1's. Na Figura 97 é possível observar a janela de configurações desse bloco.

Figura 97 – Janela de configurações do bloco *Bernoulli Binary Generator*.



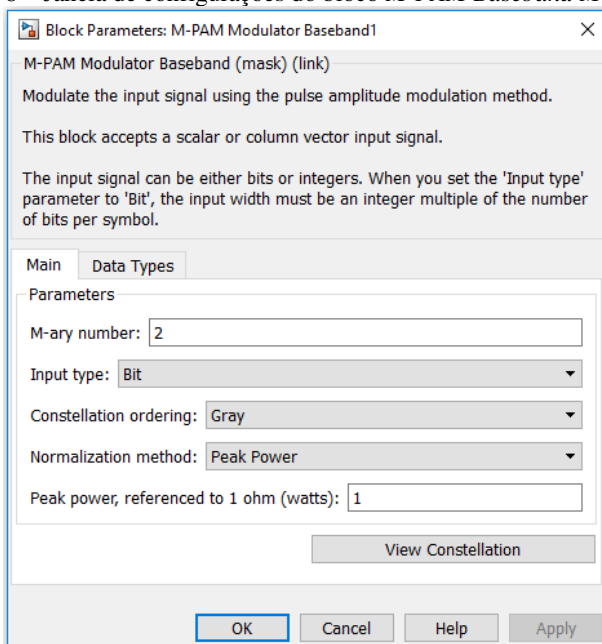
Fonte: O autor (2018).



O parâmetro *Initial seed* recebe um valor que é utilizado como semente de um gerador de números randômicos. O parâmetro *Sample time*, que corresponde ao período dos bits, foi definido como sendo 20/48000 s, o que corresponde a uma taxa de bits  $R_b = 2400$  bits/s. Esses bits possuem o formato *double*, e são agrupados em quadros de 1000 amostras.

O bloco *M-PAM Modulador Baseband*, observado na Figura 96, recebe como entrada uma sequência de bits e gera como saída uma sequência de símbolos de um alfabeto M-ário. Na Figura 98 é possível observar a janela de configurações desse bloco.

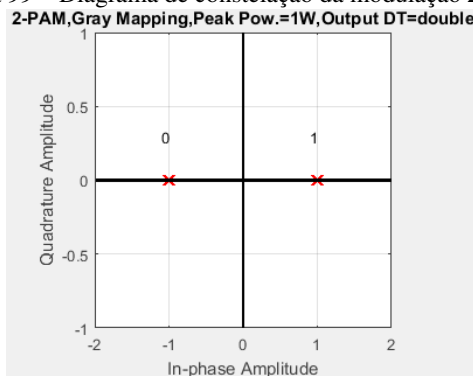
Figura 98 – Janela de configurações do bloco *M-PAM Baseband Modulator*.



Fonte: O autor (2018).

Utilizando os parâmetros mostrados na Figura 98, o diagrama de constelação resultante (o diagrama que mostra o mapeamento entre bits e símbolos correspondentes) pode ser observado na Figura 99.

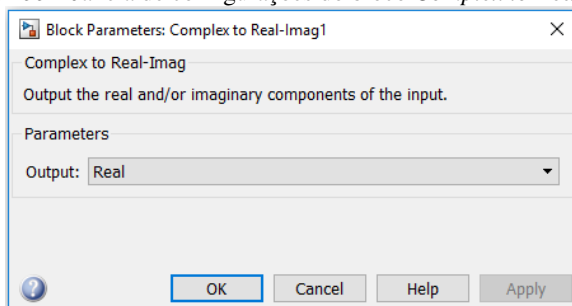
Figura 99 – Diagrama de constelação da modulação 2-PAM.



Fonte: O autor (2018).

O bloco *Complex to Real-Imag* que pode ser observado na Figura 96 tem a função de extrair a parte real do sinal de saída do bloco *M-PAM Modulator Baseband*, que é uma sequência de números complexos (pontos da constelação mostrada na Figura 99). A janela de configurações do bloco *Complex to Real-Imag* pode ser observada na Figura 100.

Figura 100 – Janela de configurações do bloco *Complex to Real-Imag*.



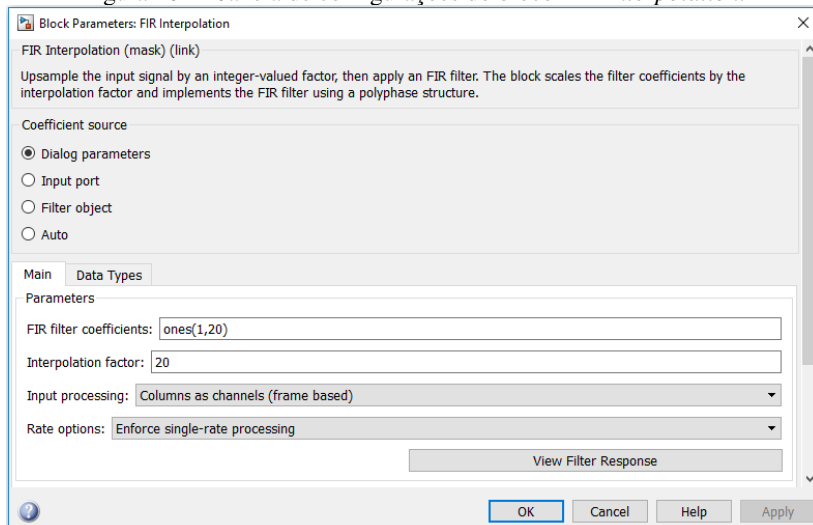
Fonte: O autor (2018).

Dessa forma, o sinal à saída do bloco *Complex to Real-Imag* é uma sequência de símbolos (números reais)  $\{-1, +1\}$ . Cada símbolo  $-1$  corresponde a um bit 0, e cada símbolo  $+1$  corresponde a um bit 1. Como a taxa de bits utilizada é  $R_b = 2400$  bits/s, a taxa de símbolos será então  $R_s = 2400$  símbolos/s.

O bloco *FIR Interpolation* que pode ser observado na Figura 96 implementa o filtro de transmissão cuja resposta ao impulso é o pulso de

transmissão  $g(t)$ . A janela de configurações desse bloco pode ser observada na Figura 101.

Figura 101 – Janela de configurações do bloco *FIR Interpolation*.



Fonte: O autor (2018).

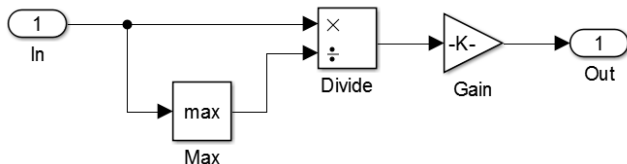
Para a execução deste experimento, cada pulso  $g(t)$  será constituído por 20 amostras. Dessa forma, o parâmetro *Interpolation factor* do bloco *FIR Interpolation*, que corresponde ao número de amostras por símbolo, deve ser definido como 20. Para que o pulso de transmissão seja do tipo NRZ, os coeficientes do filtro são definidos com o uso da expressão do MATLAB “ones(1,20)”. Além disso, o parâmetro *Rate options* deve ser definido como *Enforce single-rate processing*, pois dessa forma o bloco adapta o tamanho dos quadros de saída para que as taxas de amostragem da entrada e da saída sejam as mesmas, independentemente da sobreamostragem (*upsampling*) realizada pelo bloco.

O sinal de entrada do bloco *FIR Interpolation* é uma sequência de símbolos a uma taxa de 2400 símbolos/s. Como para cada um desses símbolos é atribuído um pulso de transmissão constituído por 20 amostras, o sinal à saída do bloco possui taxa de 48000 amostras/s.

Por fim, na Figura 102 podemos observar os componentes internos do subsistema AGC. A função desse sistema é realizar o controle de ganho automático do sinal à saída do bloco *FIR*

*Interpolation.* Esse controle de ganho é feito através de uma normalização: para cada quadro à entrada, o valor de todas as amostras é dividido pelo valor máximo encontrado nas amostras do quadro, e em seguida todas as amostras são multiplicadas por uma constante igual a  $2^{15} - 1$ .

Figura 102 – Componentes internos do subsistema AGC.

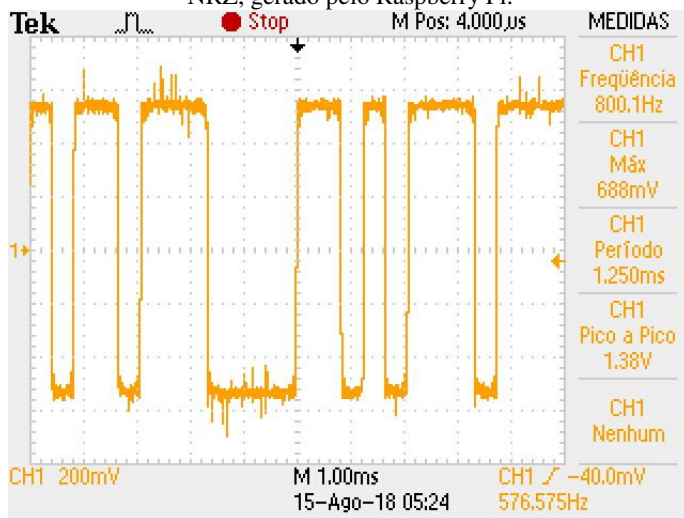


Fonte: O autor (2018).

Explicações sobre os demais blocos que podem ser observados na Figura 96 já foram apresentadas nas seções anteriores.

Após a execução do experimento, é possível observar na tela do osciloscópio o sinal da Figura 103.

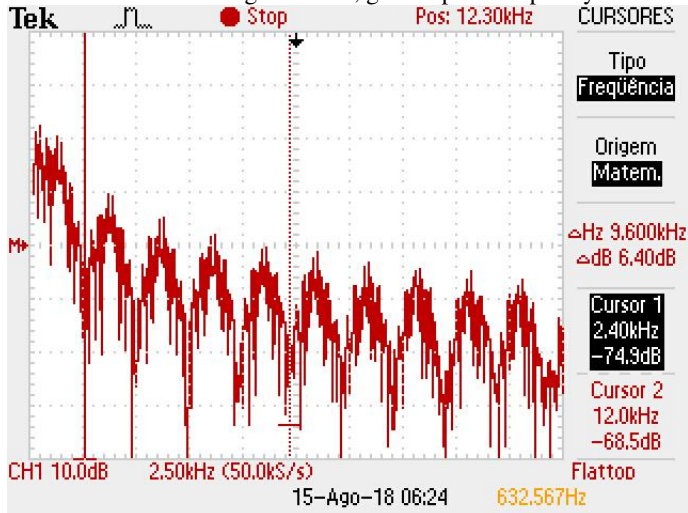
Figura 103 – Sinal 2-PAM em banda base, com pulso de transmissão retangular NRZ, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, foi possível observar o espectro (magnitude) do sinal mostrado na Figura 103. Esse espectro, por sua vez, pode ser observado na Figura 104.

Figura 104 – Espectro do sinal 2-PAM em banda base, com pulso de transmissão retangular NRZ, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Como o pulso de transmissão é uma função  $\text{rect}$  no tempo, era esperado que seu espectro fosse uma função do tipo sinc na frequência. Os cursores que podem ser observados na Figura 104 indicam que os pontos de nulo do espectro de magnitude ocorrem em frequências múltiplas de  $1/T_s = 2.4 \text{ kHz}$ , que é justamente o valor da taxa de símbolos.

#### 4.7.2 Modulação 2-PAM com pulso de transmissão retangular RZ

O segundo pulso de transmissão a ser utilizado no experimento é o pulso retangular com retorno a zero (RZ, do inglês *return-to-zero*). A expressão (no domínio do tempo) desse pulso é dada por:

$$g(t) = \text{rect}\left(\frac{t - T/4}{T/2}\right) = \begin{cases} 1, & 0 \leq t \leq T/2 \\ 0, & \text{caso contrário} \end{cases}$$

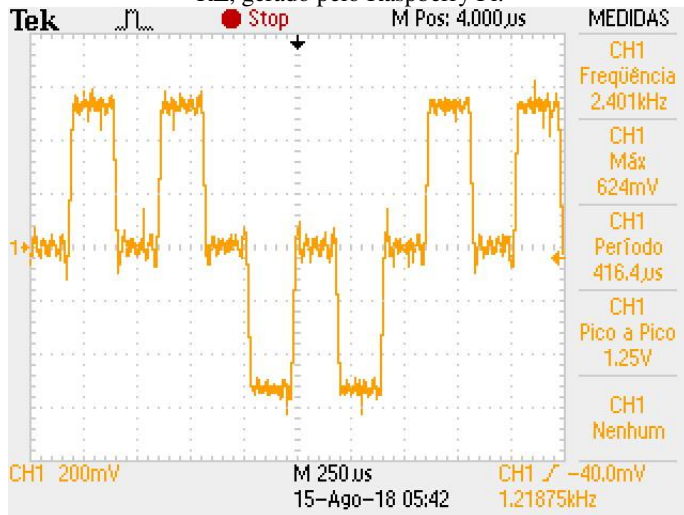
O modelo do Simulink utilizado para transformar o Raspberry Pi num modular em banda base 2-PAM que utiliza pulsos de transmissão

retangulares RZ é exatamente igual ao mostrado anteriormente na Figura 96. Os seguintes parâmetros são mantidos: taxa de bits  $R_b = 2400$  bits/s, taxa de símbolos  $R_s = 2400$  símbolos/s e fator de sobreamostragem (*upsampling factor*) igual a 20.

A única diferença entre os dois modelos do Simulink consiste na configuração dos parâmetros do bloco *FIR Interpolation*. Para que o pulso de transmissão seja RZ, e não NRZ, os coeficientes do filtro devem ser definidos com o uso da expressão do MATLAB “[ones(1,10) zeros(1,10)]”.

Após o início da execução do modelo do Simulink, é possível observar na tela do osciloscópio o sinal mostrado na Figura 105.

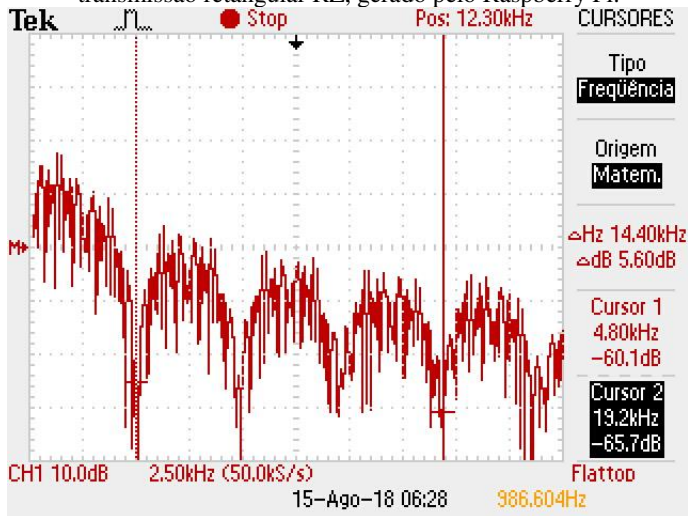
Figura 105 – Sinal 2-PAM em banda base, com pulso de transmissão retangular RZ, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível observar o espectro (magnitude) do sinal mostrado na Figura 105. Esse espectro, por sua vez, pode ser observado na Figura 106.

Figura 106 – Espectro do sinal 2-PAM em banda base, com pulso de transmissão retangular RZ, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

O pulso retangular RZ, assim como o NRZ, também é uma função rect no tempo. Logo, era esperado que seu espectro fosse uma função sinc na frequência. Os cursores que podem ser observados na Figura 106 indicam que os pontos de nulo do espectro ocorrem em frequências múltiplas de  $2R_s = 4800$  kHz. Esse resultado já era esperado e previsto pela dualidade tempo-frequência. O pulso retangular RZ, no domínio do tempo, possui metade da duração do pulso NRZ. Consequentemente, seu espectro ocupa o dobro de banda de frequência.

Após a execução dos experimentos com os pulsos de transmissão retangulares NRZ e RZ, é possível discutir suas vantagens e desvantagens. Quando o pulso NRZ está sendo utilizado, uma longa sequência de 1's ou 0's seguidos transmitidos pode fazer com que o receptor perca a informação sobre a temporização. Esse problema não ocorre quando o pulso RZ está sendo utilizado, pois para cada bit transmitido há uma transição (degrau) no sinal modulado. Essa transição pode ser utilizada pelo receptor para a recuperação da informação de temporização. Porém, o preço cobrado pelo pulso RZ pela informação de temporização contida no sinal transmitido é uma largura de banda que é o dobro da largura de banda do pulso NRZ.

### 4.7.3 Modulação 2-PAM com pulso de transmissão Manchester

O terceiro pulso de transmissão a ser utilizado neste experimento é o pulso retangular do tipo Manchester. Sua expressão (no domínio do tempo) é dada por:

$$g(t) = \text{rect}\left(\frac{t - T/4}{T/2}\right) - \text{rect}\left(\frac{t - 3T/4}{T/2}\right) = \begin{cases} 1, & 0 \leq t < T/2, \\ -1, & T/2 \leq t < 3T/2. \\ 0, & \text{caso contrário} \end{cases}$$

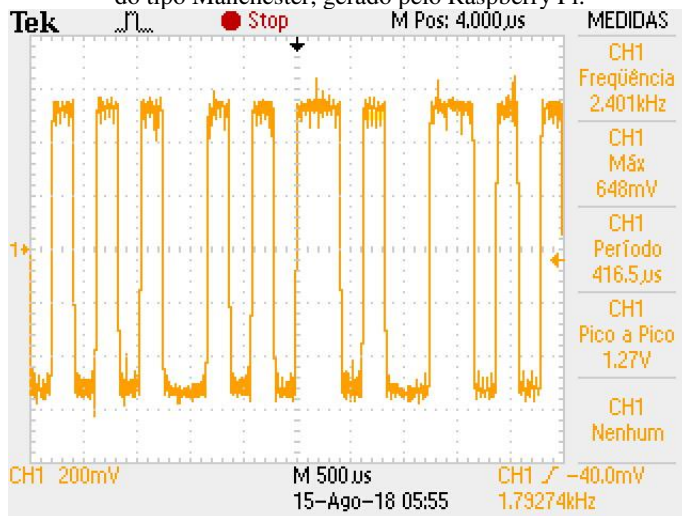
O modelo do Simulink utilizado para transformar o Raspberry Pi num modular em banda base 2-PAM que utiliza pulsos de transmissão do tipo Manchester é exatamente igual ao utilizado para o caso dos pulsos NRZ e RZ, e pode ser observado na Figura 96. Os seguintes parâmetros ainda são mantidos: taxa de bits  $R_b = 2400$  bits/s, taxa de símbolos  $R_s = 2400$  símbolos/s e fator de sobreamostragem (*upsampling factor*) igual a 20.

Novamente, a única diferença consiste na configuração dos parâmetros do bloco *FIR Interpolation*. Para que o pulso de transmissão seja o retangular do tipo Manchester, os coeficientes do filtro devem ser definidos com o uso da expressão do MATLAB “[ones(1,10) – ones(1,10)]”.

Após o início da execução do modelo do Simulink, é possível observar na tela do osciloscópio o sinal mostrado na Figura 107.



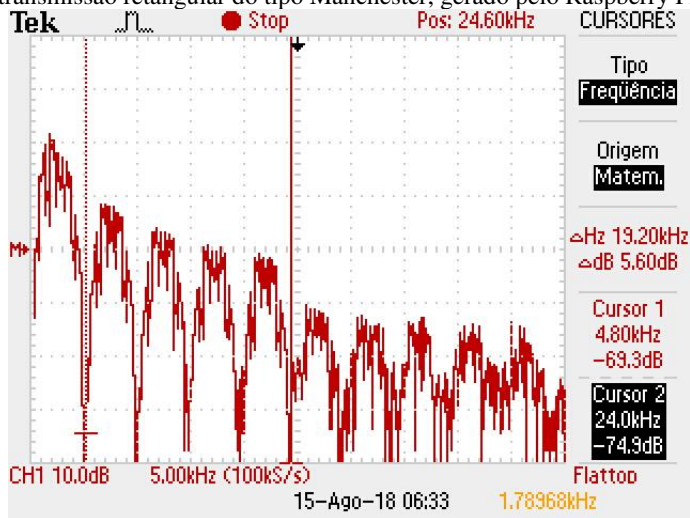
Figura 107 – Sinal 2-PAM em banda base, com pulso de transmissão retangular do tipo Manchester, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível observar o espectro (magnitude) do sinal mostrado na Figura 107. Esse espectro, por sua vez, pode ser observado na Figura 108.

Figura 108 – Espectro do sinal 2-PAM em banda base, com pulso de transmissão retangular do tipo Manchester, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

O pulso do tipo Manchester, do domínio do tempo, possui duração  $T$  e é formado pela concatenação de dois pulsos rect com duração  $T/2$ . Logo, era esperado que o espectro do sinal fosse uma função do tipo sinc na frequência com pontos de nulo em frequências múltiplas de  $2/T$ . Cursores na Figura 108 indicam os pontos de nulo do espectro. No caso desse experimento, como a taxa de símbolos é  $R_s = 2400$  símbolos/s, os pontos de nulo ocorrem em frequências múltiplas de  $2/T_s = 2R_s = 4800$  kHz.

Analisando a Figura 107 e a Figura 108, é possível concluir que, assim como o pulso retangular RZ, o uso do pulso do tipo Manchester faz com que o sinal transmitido contenha a informação sobre temporização ao custo de uma largura de banda que é o dobro da largura de banda quando o pulso NRZ é utilizado.

#### 4.7.4 Modulação 4-PAM com pulso de transmissão retangular

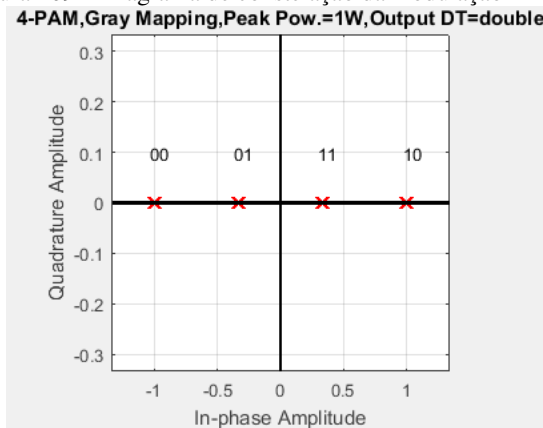
Está é a última etapa do experimento que utiliza pulsos de transmissão retangulares. O objetivo agora é transformar o Raspberry Pi num modulador num modulador 4-PAM que utilize o pulso de transmissão retangular NRZ. Para isso, o modelo do Simulink mostrado

anteriormente na Figura 96 é mais uma vez reaproveitado. A configuração de parâmetros de alguns blocos deve ser alterada.

Primeiramente, o parâmetro *Sample time* do bloco *Bernoulli Binary Generator* deve ser alterado para 10/48000. Com isso, o novo valor da taxa de bits passa a ser  $R_b = 4800$  bits/s.

Em seguida, o parâmetro *M-ary number* do bloco *M-PAM Modulator Baseband* deve ser alterado para 4. Com isso, a cada sequência de dois bits apresentada à entrada do bloco será atribuído um símbolo de um alfabeto quaternário. Mantendo-se as demais configurações utilizadas anteriormente para a modulação 2-PAM, o diagrama de constelação resultante obtido pode ser observado na Figura 109. Essa constelação pode ser observada após clicar-se no botão *View Constellation* localizado dentro da janela de configurações do bloco *M-PAM Modulator Baseband*.

Figura 109 – Diagrama de constelação da modulação 4-PAM.



Fonte: O autor (2018).

O bloco *Complex to Real-Imag* presente no modelo do Simulink extrai a parte real dos pontos da constelação mostrada na Figura 109. Dessa forma, à sequência de bits “00” é atribuído o símbolo “-1”, à sequência de bits “01” é atribuído o símbolo “-1/3”, à sequência de bits “11” é atribuído o símbolo “1/3” e à sequência de bits “10” é atribuído o símbolo “1”.

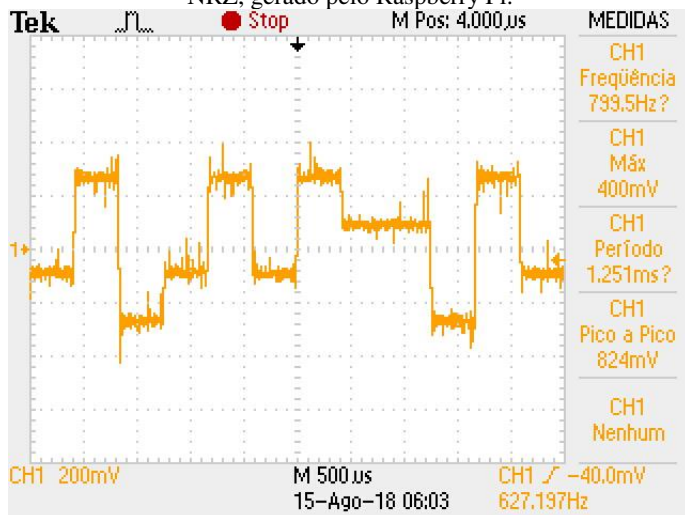
Cada símbolo gerado necessita de 2 bits, sendo que a taxa de bits sendo utilizada é  $R_b = 4800$  bits/s. Consequentemente, a taxa de símbolos é  $R_s = R_b/2 = 2400$  símbolos/s.

O fator de sobreamostragem sendo utilizado, assim como nas etapas anteriores, continua sendo igual a 20. Logo, a cada símbolo é atribuído um pulso de transmissão constituído de 20 amostras. O pulso de transmissão sendo utilizado nesta etapa é o retangular NRZ, e por isso os coeficientes do bloco *FIR Interpolation* são definidos com o uso da expressão do MATLAB “ones(1,20)”. Sendo assim, o sinal à saída do bloco possui taxa de amostragem de 48 kHz.

Por fim, a última alteração necessária no modelo do Simulink é o ajuste do ganho aplicado ao sinal. Esse ganho é definido pelo bloco *Gain* que pode ser encontrado dentro do subsistema *AGC*. Para evitar a distorção do sinal à saída da placa de som USB, o valor de ganho a ser adotado para o modulador 4-PAM deve ser de  $2^{14} - 1$ .

Após o início da execução do modelo do Simulink, é possível observar na tela do osciloscópio o sinal mostrado na Figura 110.

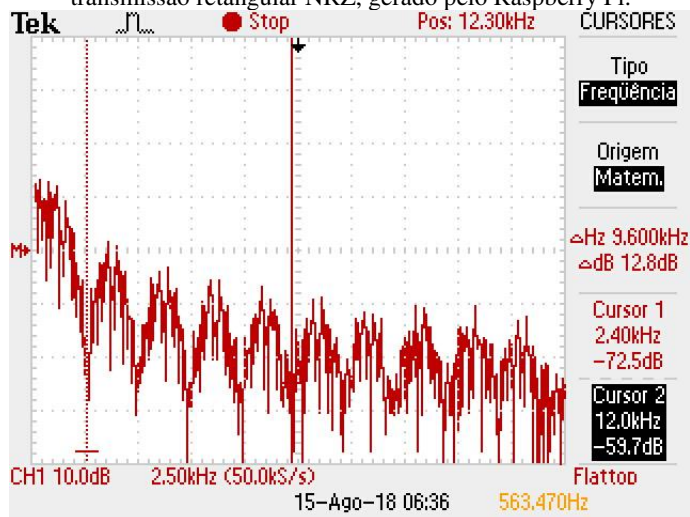
Figura 110 – Sinal 4-PAM em banda base, com pulso de transmissão retangular NRZ, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível observar o espectro (magnitude) do sinal mostrado na Figura 110. Esse espectro, por sua vez, pode ser observado na Figura 111.

Figura 111 – Espectro do sinal 4-PAM em banda base, com pulso de transmissão retangular NRZ, gerado pelo Raspberry Pi.



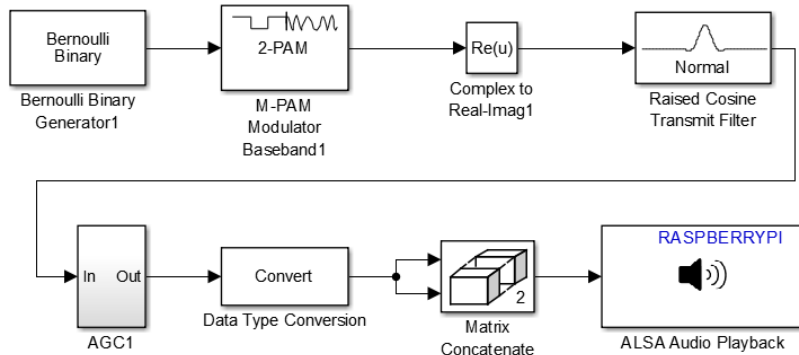
Fonte: O autor (2018).

Na Figura 111 é possível observar que o espectro do sinal 4-PAM com pulsos NRZ apresenta a mesma forma do espectro do sinal 2-PAM com pulsos NRZ, observado anteriormente na Figura 104. Isso se deve ao fato de que o formato do espectro depende somente do tipo de pulso de transmissão utilizado.

#### 4.7.5 Modulação 2-PAM com pulso Cosseno Levantado

O modelo do Simulink que pode ser observado na Figura 112 transforma o Raspberry Pi num modulador 2-PAM que utiliza pulsos de transmissão do tipo Cosseno Levantado.

Figura 112 – Modelo do Simulink que transforma o Raspberry Pi num modulador 2-PAM em banda base com pulso cosseno levantado.

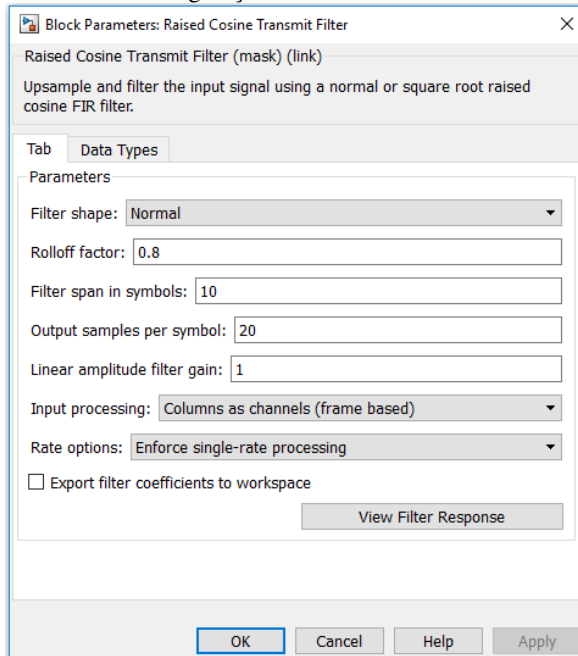


Fonte: O autor (2018).

A única diferença entre o modelo do Simulink mostrado na Figura 112 em comparação com os modelos do Simulink utilizados nos experimentos de modulação 2-PAM com pulsos de transmissão retangulares consiste na substituição do bloco *FIR Interpolation* pelo bloco *Raised Cosine Transmit Filter*. As configurações dos demais blocos são mantidas.

A janela de configurações do bloco *Raised Cosine Transmit Filter* pode ser observada na Figura 113. O parâmetro *Filter shape* deve ser definido como *Normal*, pois assim o bloco implementa um filtro de transmissão cuja resposta ao impulso é um pulso com espectro do tipo cosseno levantado. Se esse parâmetro fosse definido como *Square root*, o espectro do pulso seria do tipo raiz de cosseno levantado.

Figura 113 – Janela de configurações do bloco *Raised Cosine Transmit Filter*.



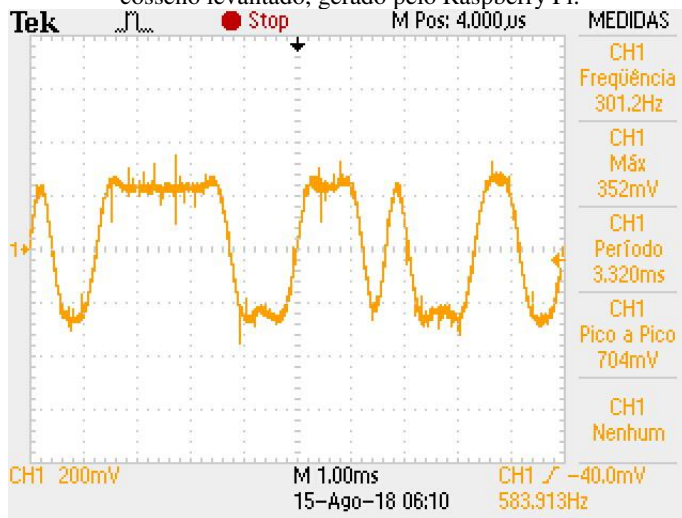
Fonte: O autor (2018).

Na Figura 113 podemos observar a configuração de parâmetros do bloco que é utilizada para o experimento. O fator de *roll-off* a ser utilizado é de 0.8. A duração de cada pulso de transmissão é de 10 símbolos. O fator de sobreamostragem (*upsampling factor*) é igual a 20. Por fim, o ganho do filtro é unitário. Assim como na utilização do bloco *FIR Interpolation*, o parâmetro *Rate options* deve ser definido como *Enforce single-rate processing*.

O ganho aplicado ao sinal 2-PAM, definido pelo bloco *Gain* dentro do subsistema *AGC* deve ser definido como  $2^{14} - 1$  para evitar a distorção do sinal à saída da placa de som USB.

Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 114.

Figura 114 – Sinal 2-PAM em banda base, com pulso de transmissão do tipo cosseno levantado, gerado pelo Raspberry Pi.

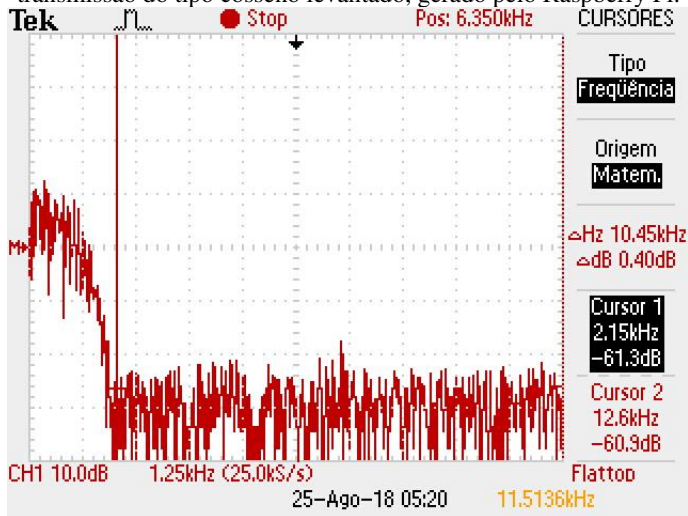


Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível observar o espectro (magnitude) do sinal mostrado na Figura 114. Esse espectro, por sua vez, pode ser observado na Figura 115. Nessa Figura é possível observar claramente que, diferentemente dos espectros dos pulsos retangulares, o espectro do pulso cosseno levantado é limitado em frequência. Além disso, um cursor foi adicionado na Figura para possibilitar a estimação da largura de banda do sinal.



Figura 115 – Espectro do sinal 2-PAM em banda base, com pulso de transmissão do tipo cosseno levantado, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

A largura de banda do pulso cosseno levantado é dada por:

$$B = (1 + \alpha) \frac{1}{2T_s} = (1 + \alpha) \frac{R_s}{2}.$$

No experimento apresentado nesta subseção, temos que  $R_s = 2400$  símbolos/s e  $\alpha = 0.8$ . Logo, a largura de banda do sinal gerado pelo Raspberry Pi é de 2.16 kHz. Esse valor está consistente com a estimação do espectro que pode ser observada na Figura 115.

Como sugestão de experimento prático, os estudantes podem realizar a transmissão digital em banda base utilizando pulsos de transmissão com diferentes valores de fator de *roll-off* (por exemplo, 0, 0.2, 0.4, 0.6, 0.8 e 1). Para cada um desses valores, os estudantes podem calcular a largura de banda teórica e compará-la com o valor obtido na prática. O valor prático pode ser estimado com o uso de um cursor no espectro do sinal obtido com a ferramenta FFT do osciloscópio.

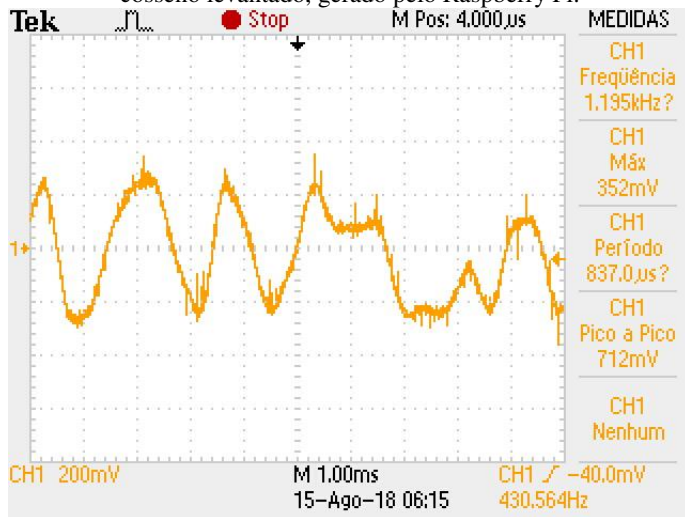
#### 4.7.6 Modulação 4-PAM com pulso Cosseno Levantado

Esta é a última etapa do experimento que transforma o Raspberry Pi num modulador digital em banda base. Agora o Raspberry Pi será configurado para gerar um sinal em banda base 4-PAM com pulso de

transmissão cosseno levantado. O modelo do Simulink necessário é exatamente igual ao mostrado anteriormente na Figura 112. Uma única alteração é necessária: o parâmetro *M-ary number* do bloco *M-PAM Modulator Baseband* deve ser definido como 4.

Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 116.

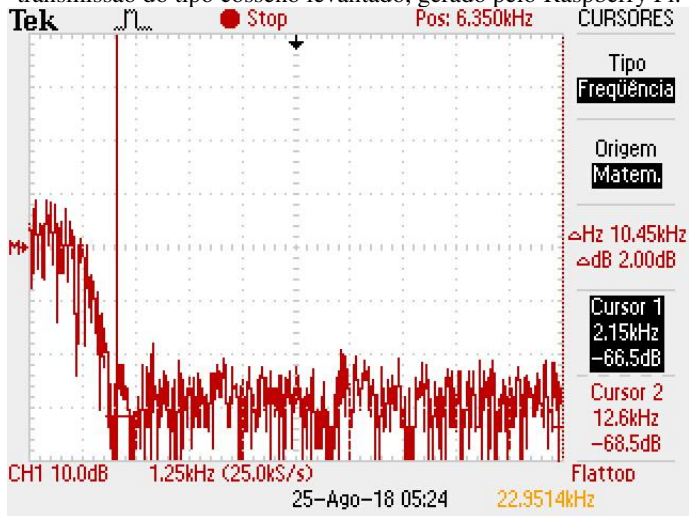
Figura 116 – Sinal 4-PAM em banda base, com pulso de transmissão do tipo cosseno levantado, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível observar o espectro (magnitude) do sinal mostrado na Figura 116. Esse espectro, por sua vez, pode ser observado na Figura 117. Nessa Figura é possível observar que, assim como o espectro do sinal 2-PAM mostrado na Figura 115, o espectro do sinal 4-PAM com pulso de transmissão cosseno levantado é limitado em frequência. Um cursor foi adicionado a esse espectro de modo a possibilitar a estimação da largura de banda do sinal gerado.

Figura 117 – Espectro do sinal 4-PAM em banda base, com pulso de transmissão do tipo cosseno levantado, gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Conforme pode ser observado na Figura 117, a largura de banda do sinal 4-PAM com pulso de transmissão cosseno levantado é aproximadamente 2.16 kHz, valor igual ao da largura de banda do sinal 2-PAM apresentado anteriormente. Esse resultado já era esperado visto que ambos os sinais utilizam o mesmo pulso de transmissão.

#### 4.8 MODULAÇÃO ASK COM O RASPBERRY PI

Este experimento e os seguintes tratarão de modulações digitais em banda passante. Neste experimento, o Raspberry Pi será utilizado para realizar modulações por chaveamento de amplitude, isto é, modulações ASK (do inglês *Amplitude Shift Keying*).

De acordo com Lathi e Ding (2015), numa modulação em amplitude, a amplitude de uma portadora é variada em proporção ao sinal de mensagem (isto é, em banda base). Para a modulação digital, esse sinal de mensagem é um trem de pulsos dado por:

$$m(t) = \sum_{i=-\infty}^{\infty} a_i g(t - iT).$$

Na expressão acima,  $a_i$  corresponde a sequência de símbolos de um alfabeto M-ário,  $g(t)$  é o pulso de transmissão e  $T$  é o período do símbolo. O sinal modulado ASK é dado então pela seguinte expressão:

$$s(t) = m(t) \cos(2\pi f_c t).$$

Assim, na modulação ASK, a informação é transmitida através de alterações na amplitude da portadora senoidal, sendo que a amplitude da portadora só pode assumir um de  $M$  possíveis valores discretos.

Neste experimento, o Raspberry Pi será utilizado para gerar sinais modulados para três diferentes tipos de chaveamento de amplitude: OOK, 2-ASK e 4-ASK.

#### 4.8.1 Geração de um sinal OOK com o Raspberry Pi

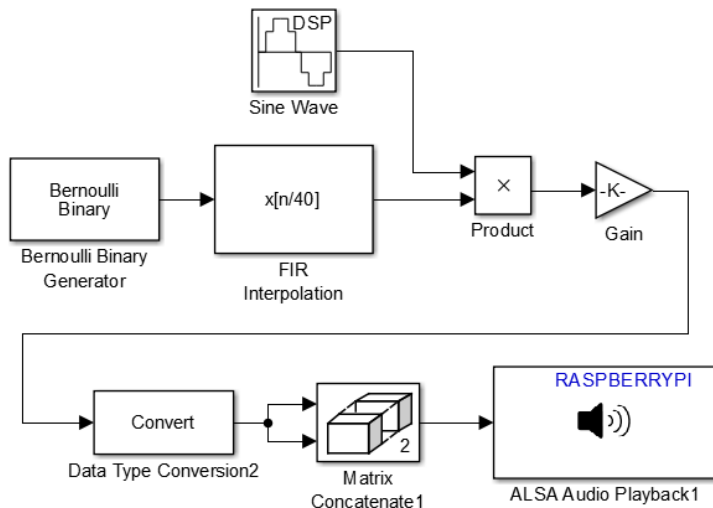
O primeiro tipo de chaveamento de amplitude a ser utilizado é o chaveamento On-Off (OOK, do inglês *On-Off Keying*). Nesse tipo de modulação, o símbolo “0” é atribuído ao bit de informação 0 e o símbolo “1” é atribuído ao bit de informação 1. Dessa forma (considerando o pulso de transmissão retangular com duração  $T_s$ ), o sinal transmitido será dado por:

$$s(t) = \begin{cases} 0, & \text{para o bit de informação 0;} \\ \cos(2\pi f_c t), & \text{para o bit de informação 1.} \end{cases}$$

Assim, o sinal modulado será composto por pulsos de RF com duração  $T_s$  intercalados por períodos de “silêncio” (ausência de sinal transmitido).

O modelo do Simulink que pode ser observado na Figura 118 é utilizado para transformar o Raspberry Pi num modulador OOK. Todos os blocos que podem ser observados nessa Figura já foram apresentados em experimentos anteriores, então apenas serão apontados os principais parâmetros de configuração para a execução do experimento.

Figura 118 – Modelo do Simulink que transforma o Raspberry Pi num modulador OOK.



Fonte: O autor (2018).

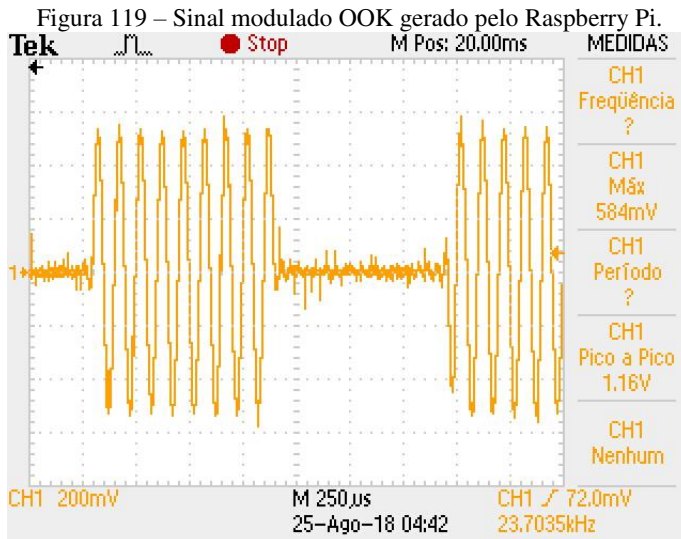
O bloco *Bernoulli Binary Generator* foi configurado para gerar uma sequência aleatória de bits a uma taxa  $R_b = 1200$  bits/s. O formato do sinal de saída é *double*, e suas amostras estão agrupadas em quadros de 1000 amostras.

O bloco *FIR Interpolation* é utilizado para implementar um filtro de transmissão cuja resposta ao impulso é o pulso de transmissão retangular NRZ. Neste experimento, o fator de sobreamostragem utilizado (*Interpolation factor*) é igual a 40, ou seja, para cada bit de informação é associado um pulso de transmissão de 40 amostras. Dessa forma, o sinal à saída desse bloco possui uma taxa de amostras de 48000 amostras/s. Portanto, os coeficientes do filtro são determinados com o uso do comando da expressão do MATLAB “ones(1,40)”.

O sinal à saída do filtro de transmissão é uma sequência de pulsos retangulares constituídos de 40 amostras, sendo que a amplitude desses pulsos pode ser somente igual a 0 ou 1. O bloco *Product* realiza a multiplicação dessa sequência de pulsos (o sinal de mensagem  $m(t)$ ) pela portadora senoidal  $\cos(2\pi f_c t)$  que é gerada pelo bloco *Sine Wave*.

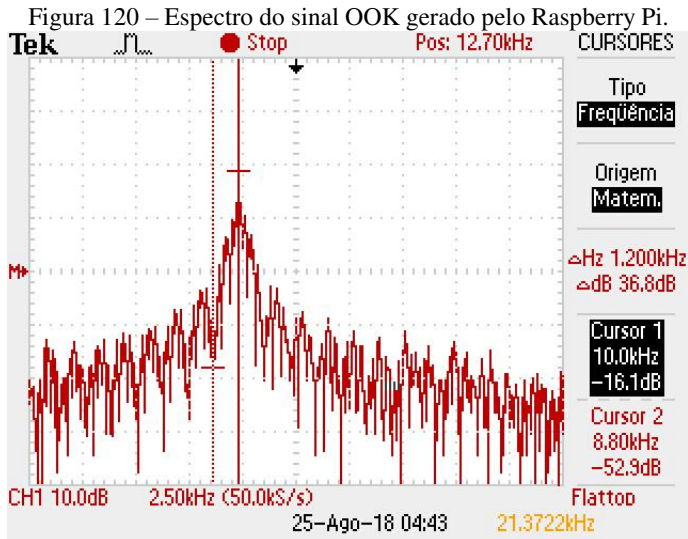
O bloco *Sine Wave* foi configurado para gerar um sinal senoidal com amplitude unitária e frequência  $f_c = 10$  kHz. Por fim, o bloco *Gain* é utilizado para aplicar um ganho igual a  $2^{15} - 1$  ao sinal modulado.

Após o início da execução do experimento, o sinal mostrado na Figura 119 é observado na tela do osciloscópio. Esse sinal está de acordo com o que era esperado.



Fonte: O autor (2018).

Utilizando a função FFT do osciloscópio, foi possível observar o espectro do sinal mostrado na Figura 119. Esse espectro, por sua vez, pode ser observado na Figura 120. Dois cursores podem ser observados nessa Figura. Um deles indica a frequência central do espectro, que é de 10 kHz (a frequência da portadora senoidal). O outro cursor indica um ponto de nulo do espectro na frequência de 8800 Hz.



Fonte: O autor (2018).

Conforme mencionado nos experimento de transmissão digital em banda base, o pulso retangular com duração  $T$ , que é uma função  $\text{rect}$  no tempo, possui um espectro em banda base que é uma função  $\text{sinc}$  na frequência, cujos pontos de nulos ocorrem em frequências múltiplas de  $1/T$ . O espectro que pode ser observado na Figura 120 corresponde ao espectro do pulso retangular deslocado para banda passante.

A taxa de símbolos utilizada nesse experimento é igual à taxa de bits, isto é,  $T_s = 1/1200$  s. Logo, o espectro do sinal OOK possui pontos de nulo em frequências múltiplas de  $1/T_s = 1200$  Hz. Assumindo que a largura de banda do sinal em banda passante é dada pela banda de frequências entre os dois pontos de nulo em torno da frequência central  $f_c = 10$  kHz, temos então que  $B \approx 2/T_s = 2400$  Hz.

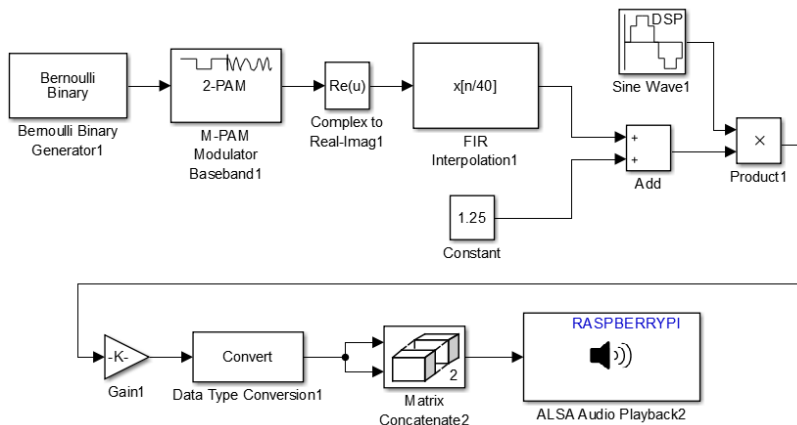
#### 4.8.2 Geração de um sinal 2-ASK com o Raspberry Pi

Para a modulação 2-ASK, o símbolo “ $A_0$ ” é atribuído ao bit de informação 0 e o símbolo “ $A_1$ ” é atribuído ao bit de informação 1. Assim, a amplitude da portadora senoidal assume um de dois valores possíveis ( $A_0$  ou  $A_1$ ). A expressão do sinal modulado é então dada por:

$$s(t) = \begin{cases} A_0 \cos(2\pi f_c t), & \text{para o bit de informação 0;} \\ A_1 \cos(2\pi f_c t), & \text{para o bit de informação 1.} \end{cases}$$

Na Figura 121 é possível observar o modelo do Simulink que transforma o Raspberry Pi num modulador 2-ASK que utiliza o pulso de transmissão retangular. Todos os blocos que podem ser observados nessa Figura já foram apresentados em experimentos anteriores. Portanto, a seguir serão apontados os principais parâmetros de configuração dos blocos para execução do experimento, bem como algumas explicações sobre o funcionamento do modelo.

Figura 121 – Modelo do Simulink utilizado para transformar o Raspberry Pi num modulador 2-ASK.



Fonte: O autor (2018).

O bloco *Bernoulli Binary Generator* é utilizado para gerar uma sequência aleatória de bits a uma taxa  $R_b$ , que neste experimento é igual a 1200 bits/s. O sinal de saída desse bloco é definido como *Boolean* e agrupado em quadros de 1000 amostras.

O bloco *M-PAM Modulator Baseband* tem a função atribuir símbolos de um alfabeto M-ário a sequência de bits de informação. Neste experimento, o parâmetro *M-ary number* é definido como 2, e o parâmetro *Minimum distance* é definido como 1.5. Dessa forma, o símbolo “-0.75” é atribuído ao bit de informação 0 e o símbolo “0.75” é atribuído ao bit de informação 1.

O sinal à saída do bloco *M-PAM Modulator Baseband* corresponde a uma sequência de símbolos de uma constelação, e por isso é um sinal complexo. Sendo assim, o bloco *Complex to Real-Imag* tem a função de extrair a parte real desse sinal, que é a sequência de símbolos “-0.75” e “0.75”.



O bloco *FIR Interpolation* que pode ser observado na Figura 121 é exatamente igual ao bloco de mesmo nome utilizado anteriormente para a modulação OOK. O sinal à saída desse bloco é uma sequência de pulsos retangulares NRZ, sendo que cada pulso é constituído de 40 amostras e possui duração de  $1/1200$  s. Além disso, para os bits de informação 0 os pulsos possuem amplitude igual a  $-0.75$ , e para os bits de informação 1 os pulsos possuem amplitude igual a  $0.75$ .

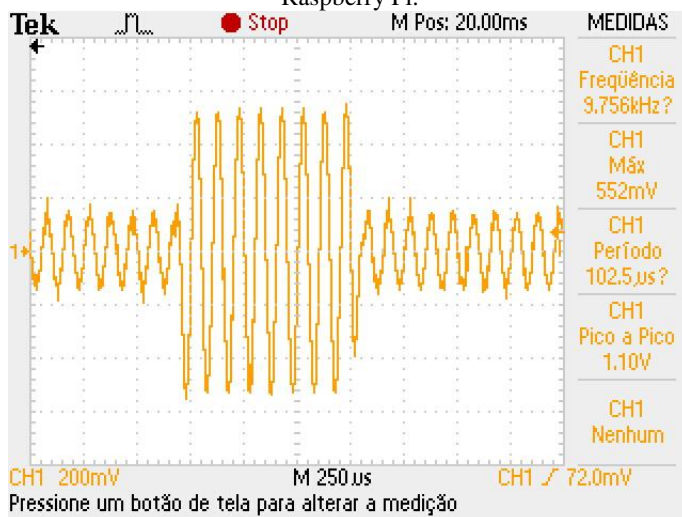
Os blocos *Constant* e *Add* que podem ser observados na Figura 121 tem a função de somar uma constante igual a  $1.25$  aos pulsos de transmissão retangulares. Dessa forma, para os bits de informação 0 os pulsos passam a possuir amplitude igual a  $0.5$ , e para os bits de informação 1 os pulsos passam a possuir amplitude igual a  $2$ .

A sequência resultante de pulsos é utilizada para modular a amplitude de uma portadora senoidal gerada pelo bloco *Sine Wave*. Esse bloco é exatamente igual ao utilizado para a modulação OOK, e assim gera um sinal senoidal com amplitude unitária e frequência de  $10$  kHz.

Para evitar a distorção do sinal à saída da placa de som USB, o bloco *Gain* deve ser configurado para aplicar um ganho igual a  $2^{14} - 1$  ao sinal modulado.

Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 122.

Figura 122 – Sinal 2-ASK com pulso de transmissão retangular gerado pelo Raspberry Pi.



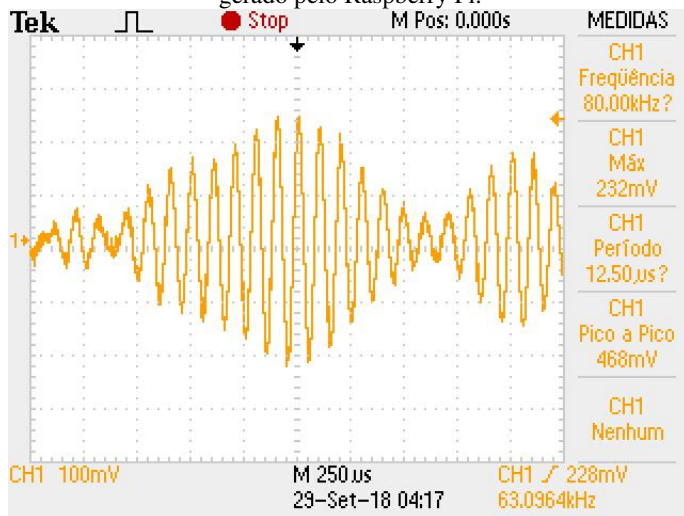
Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível observar o espectro do sinal 2-ASK gerado pelo Raspberry Pi. Esse espectro é exatamente igual ao espectro do sinal OOK mostrado na Figura 120. Isso era esperado visto que nos dois casos o pulso de transmissão  $g(t)$  é exatamente o mesmo.

O modelo do Simulink utilizado para gerar o sinal 2-ASK pode ser adaptado para utilizar o pulso de transmissão do tipo cosseno levantado. Para isso, basta substituir o bloco **FIR Interpolation** mostrado na Figura 121 por um bloco **Raised Cosine Transmit Filter**, o mesmo que foi utilizado nos experimentos de transmissão digital em banda base. Para a obtenção do sinal que será mostrado a seguir, os parâmetros utilizados para o filtro de transmissão foram: fator de *roll-off* igual a 0,8, *Filter span in symbols* igual a 10 e *Output samples per symbol* igual a 40.

Após o início da execução do experimento utilizando o pulso de transmissão do tipo cosseno levantado, é possível observar na tela do osciloscópio o sinal mostrado na Figura 123.

Figura 123 – Sinal 2-ASK com pulso de transmissão do tipo cosseno levantado gerado pelo Raspberry Pi.

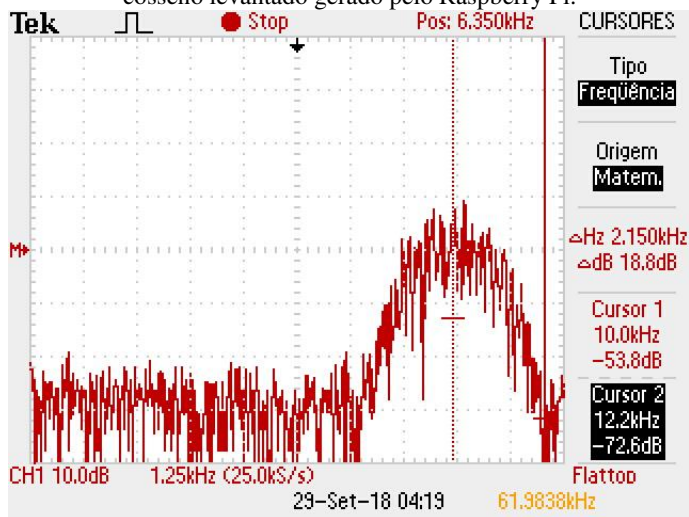


Fonte: O autor (2018).

Com o uso da ferramenta FFT do osciloscópio, é possível observar o espectro do sinal mostrado na Figura 123. Esse espectro, por

sua vez, pode ser observado na Figura 124. Dois cursores foram adicionados a esse espectro de modo a possibilitar a estimação da largura de banda do sinal.

Figura 124 – Espectro do sinal 2-ASK com pulso de transmissão do tipo cosseno levantado gerado pelo Raspberry Pi.



Fonte: O autor (2018).

O espectro mostrado na Figura 124 corresponde a uma translação do espectro do sinal digital 2-PAM em banda base, obtido no experimento anterior e que pode ser observado na Figura 115. O espectro do sinal em banda base foi deslocado para banda passante, tendo agora como frequência central o valor de frequência da portadora (10 kHz). Além disso, a largura de banda do sinal em banda passante é o dobro da largura de banda do sinal em banda base: 4.32 kHz.

#### 4.8.3 Geração de um sinal 4-ASK com o Raspberry Pi

Para a modulação 4-ASK, símbolos de um alfabeto 4-ário são atribuídos as duplas de bits de informação. Dessa forma, a amplitude portadora senoidal assume um de quatro valores possíveis. O sinal modulado (utilizando o pulso de transmissão retangular) é então uma sequência de pulsos de RF de duração  $T$  dada pela seguinte expressão:

$$s(t) = \begin{cases} A_1 \cos(2\pi f_c t), & \text{para a dupla de bits 00;} \\ A_2 \cos(2\pi f_c t), & \text{para a dupla de bits 01;} \\ A_3 \cos(2\pi f_c t), & \text{para a dupla de bits 10;} \\ A_4 \cos(2\pi f_c t), & \text{para a dupla de bits 11.} \end{cases}$$

O modelo do Simulink utilizado para transformar o Raspberry Pi num modulador 4-ASK (com uso do pulso de transmissão retangular) é exatamente igual ao modelo utilizado para a modulação 2-ASK, que pode ser observado na Figura 121. As únicas diferenças entre os dois modelos consistem apenas na configuração de alguns blocos. Portanto, a seguir serão apresentados apenas os diferentes parâmetros de configuração para execução do experimento, bem como algumas explicações sobre o funcionamento do modelo.

No bloco *M-PAM Modulator Baseband*, o parâmetro *M-ary number* deve ser definido como 4, o parâmetro *Constellation ordering* deve ser definido como *Gray*, e o parâmetro *Minimum distance* deve ser definido como 1. Dessa forma, o bloco atribui o símbolo “-1.5” à dupla de bits 00, o símbolo “-0.5” à dupla de bits 01, o símbolo “0.5” à dupla de bits 11 e o símbolo “1.5” à dupla de bits 10.

A taxa de bits (definida no bloco *Bernoulli Binary Generator*) é igual a 1200 bits/s. Como cada símbolo a atribuído a cada dois bits de informação, a taxa de símbolos é  $T_s = 600$  símbolos/s.

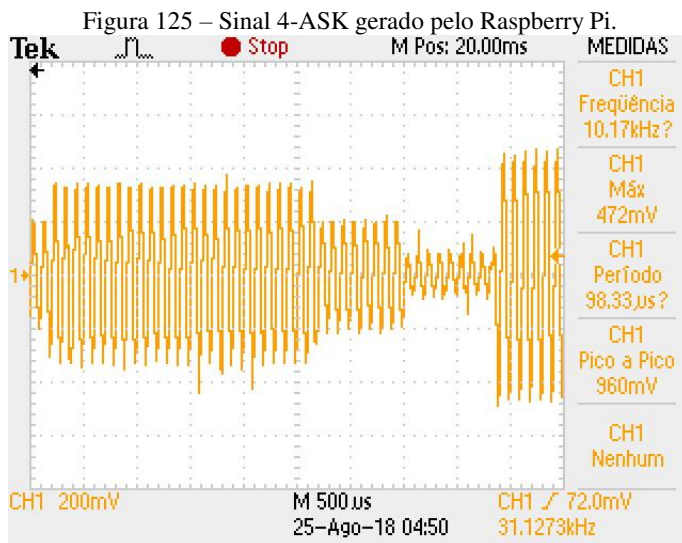
O sinal à saída do bloco *FIR Interpolation* é uma sequência de pulsos retangulares NRZ, sendo que cada pulso é constituído de 40 amostras. Além disso, a amplitude dos pulsos está associada à sequência de duplas de bits de informação da seguinte forma: “-1.5” à dupla de bits 00, o símbolo “-0.5” à dupla de bits 01, o símbolo “0.5” à dupla de bits 11 e o símbolo “1.5” à dupla de bits 10.

Na sequência, o valor da constante a ser adicionada à amplitude dos pulsos deve ser modificado para 2. Dessa forma, a amplitude dos pulsos passa a se associar à sequência de duplas de bits de informação da seguinte forma: “0.5” à dupla de bits 00, o símbolo “1.5” à dupla de bits 01, o símbolo “2.5” à dupla de bits 11 e o símbolo “3.5” à dupla de bits 10. Os pulsos resultantes são utilizados para modular a amplitude da portadora senoidal.

Por fim, o valor do ganho aplicado ao sinal modulado pelo bloco *Gain* deve ser alterado para  $2^{13} - 1$ , a fim de evitar a distorção do sinal à saída da placa de som USB.

Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 125. Nessa Figura é

possível observar que a amplitude da portadora senoidal assume sempre um de quatro valores possíveis.



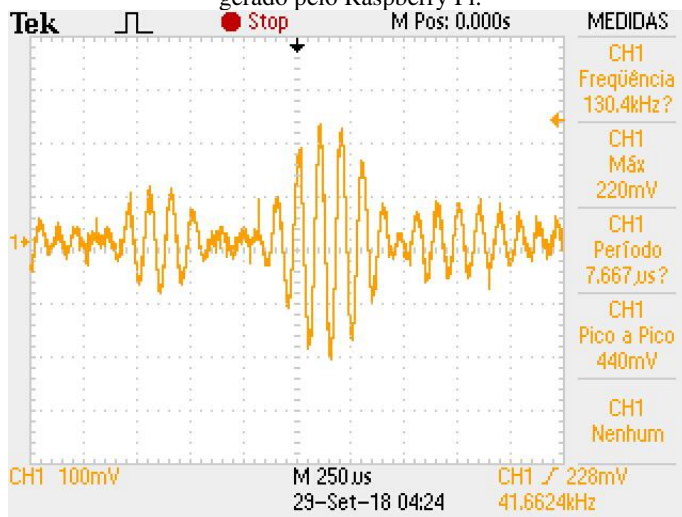
Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível observar o espectro do sinal 4-ASK gerado pelo Raspberry Pi. Esse espectro é exatamente igual aos espectros dos sinais OOK e 2-ASK gerados anteriormente, e pode ser observado na Figura 120. Isso era esperado visto que o pulso de transmissão  $g(t)$  é exatamente o mesmo nos três casos.

O modelo do Simulink utilizado para gerar o sinal 4-ASK pode ser alterado para utilizar pulsos de transmissão do tipo cosseno levantado. Para isso, a única alteração necessária é a mesma implementada anteriormente para a modulação 2-ASK: a substituição do bloco **FIR Interpolation** pelo bloco **Raised Cosine Transmit Filter**. O sinal a ser apresentado a seguir foi obtido utilizando os mesmos parâmetros do filtro de transmissão utilizado anteriormente.

Após a alteração ter sido realizado e o experimento ter sido iniciado, é possível observar na tela do osciloscópio o sinal mostrado na Figura 126.

Figura 126 – Sinal 4-ASK com pulso de transmissão do tipo cosseno levantado gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da ferramenta FFT do osciloscópio, é possível observar o espectro do sinal mostrado na Figura 126. Esse espectro, por sua vez, é exatamente igual ao espectro do sinal 2-ASK com pulso de transmissão do tipo cosseno levantado, mostrado anteriormente na Figura 124.

#### 4.9 MODULAÇÕES BPSK, BFSK e 4-FSK COM O RASPBERRY PI

O objetivo deste experimento é apresentar aos estudantes mais três modulações utilizadas para transmissão digital em banda passante: as modulações BPSK, BFSK e 4-FSK. Assim como no experimento anterior, neste experimento será utilizado apenas o pulso de transmissão retangular NRZ, a fim de facilitar a observação dos resultados.

##### 4.9.1 Geração de um sinal BPSK com o Raspberry Pi

De acordo com Lathi e Ding (2015), no esquema de chaveamento por deslocamento de fase (PSK, do inglês *Phase Shift Keying*), a transmissão de informação é realizada através do chaveamento da fase de uma portadora senoidal.

No caso da modulação BPSK (*Binary Shift Keying*), o símbolo  $-1$  é atribuído ao bit de informação 0, e o símbolo 1 é atribuído ao bit de informação 1. Assim, o sinal de mensagem é dado por:

$$m(t) = \sum_{i=-\infty}^{\infty} a_i g(t - iT).$$

Na expressão acima,  $a_i$  corresponde à sequência de símbolos e  $g(t)$  é o pulso de transmissão com duração  $T$ .

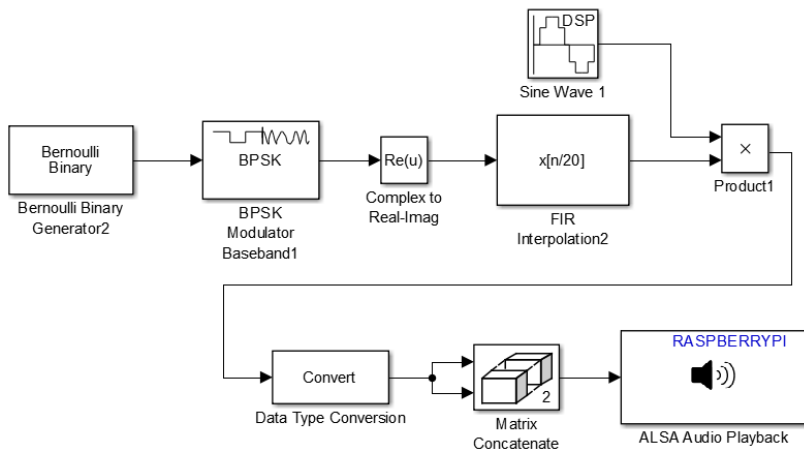
O sinal BPSK modulado possui uma expressão semelhante à do sinal ASK e é dado por:

$$s(t) = m(t) \cos(2\pi f_c t).$$

Considerando um pulso de transmissão retangular NRZ, temos então que o bit de informação 1 será transmitido pelo pulso de RF  $\cos(2\pi f_c t)$  e o bit de informação 0 será transmitido pelo pulso de RF  $-\cos(2\pi f_c t) = \cos(2\pi f_c t + \pi)$ . Portanto, os dois pulsos estão defasados de  $\pi$  radianos (LATHI e DING, 2015).

O modelo do Simulink mostrado na Figura 127 é utilizado para transformar o Raspberry Pi num gerador de um sinal modulado BPSK com pulso de transmissão retangular. Com exceção do bloco *BPSK Modulator Baseband*, todos os demais blocos do Simulink que podem ser observados nesta Figura já foram apresentados nos experimentos anteriores. Portanto, além de uma explicação sobre o novo bloco utilizado, serão apresentadas apenas breves descrições sobre o funcionamento do modelo e os principais parâmetros de configuração.

Figura 127 – Modelo do Simulink utilizado para transformar o Raspberry Pi num modulador BPSK.

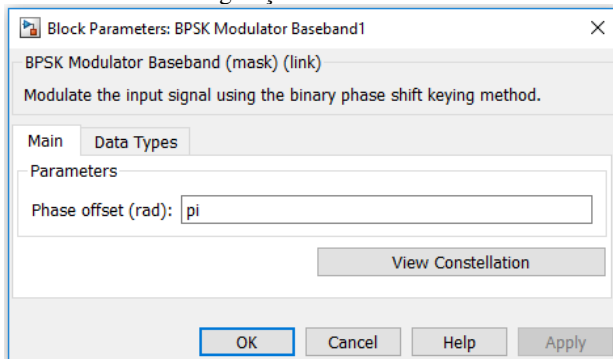


Fonte: O autor (2018).

O bloco *Bernoulli Binary Generator* é utilizado para gerar uma sequência aleatória de bits a uma taxa  $R_b = 2400$  bits/s. Neste experimento, o sinal de saída é configurado para ser do tipo *double* e estar agrupado em quadros de 1000 amostras.

O bloco *BPSK Modulator Baseband* tem a função de atribuir símbolos à sequência de bits. A janela de configurações desse bloco pode ser observada na Figura 128.

Figura 128 – Janela de configurações do bloco *BPSK Modulator Baseband*.

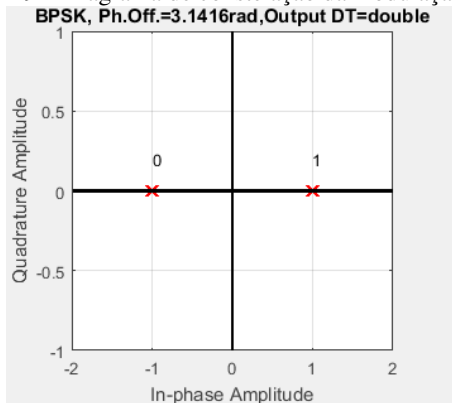


Fonte: O autor (2018).



Conforme pode ser observado na Figura 128, o único parâmetro do bloco *BPSK Modulator Baseband* a ser configurado é o *offset* da constelação de símbolos gerados (parâmetro *Phase offset (rad)*). Ao adotar-se um *offset* de  $\pi$  radianos, o bloco atribui o símbolo “-1” ao bit 0 e o símbolo “+1” ao bit 1. O diagrama de constelação resultante é obtido ao clicar-se no botão *View constellation*, e pode ser observado na Figura 129. Como a modulação é binária, a taxa de símbolos é igual à taxa de bits, ou seja,  $R_s = 2400$  símbolos/s.

Figura 129 – Diagrama de constelação da modulação BPSK.



Fonte: O autor (2018).

Se um *offset* de fase de 0 radianos fosse utilizado no lugar do *offset* de  $\pi$  radianos, o bloco atribuiria o símbolo “-1” ao bit 1 e o símbolo “+1” ao bit 0.

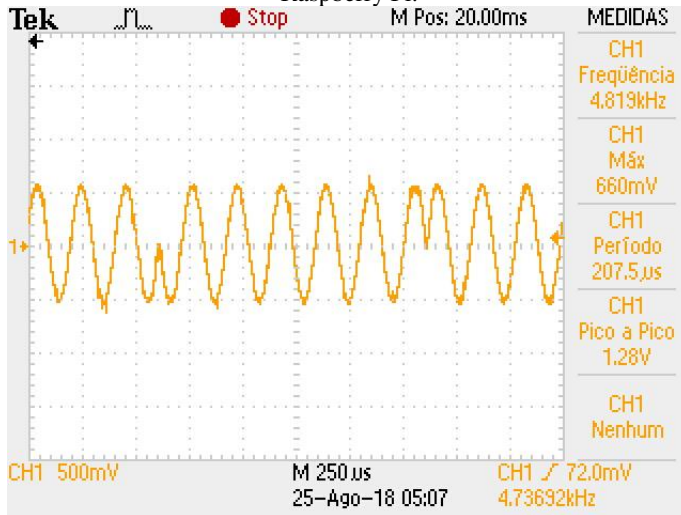
O sinal de saída do bloco *BPSK Modulator Baseband* é uma sequência de pontos de uma constelação, e por isso é do tipo complexo. Por esse motivo, é necessário a utilização do bloco *Complex to Real-Imag*, que extrai a parte real desse sinal.

O bloco *FIR Interpolation* implementa um filtro de transmissão cuja resposta ao impulso é um pulso retangular NRZ. Neste experimento, o fator de sobreamostragem utilizado (*Interpolation factor*) é igual a 20, isto é, para cada símbolo é atribuído um pulso de transmissão constituído por 20 amostras. Logo, os coeficientes do filtro são determinados com o uso da expressão do MATLAB “ones(1,20)”. A duração de cada pulso é igual à duração de cada símbolo, isto é,  $T = T_s = 1/2400$  s.

A sequência de pulsos é utilizada para modular a amplitude de uma portadora senoidal. Essa portadora é gerada pelo bloco *Sine Wave* que pode ser observado na Figura 127. Neste experimento, esse bloco está configurado para gerar um sinal senoidal com amplitude igual a  $2^{15} - 1$  e frequência  $f_c = 4800$  Hz.

Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 130. Nessa Figura é possível observar algumas inversões de fase características do sinal modulado BPSK.

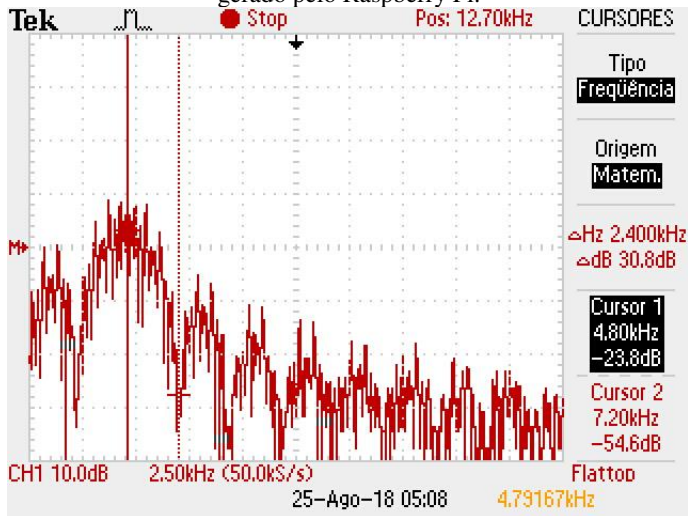
Figura 130 – Sinal BPSK com pulso de transmissão retangular gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, foi possível visualizar o espectro do sinal BPSK gerado pelo Raspberry Pi. Esse espectro pode ser observado na Figura 131. Dois cursores foram observados nessa Figura. Um deles indica a frequência central do espectro, que é a frequência da portadora senoidal  $f_c = 4.8$  kHz. O segundo cursor, localizado na frequência de 7.2 kHz indica o ponto de nulo imediatamente acima da frequência central.

Figura 131 – Espectro do sinal BPSK com pulso de transmissão retangular gerado pelo Raspberry Pi.



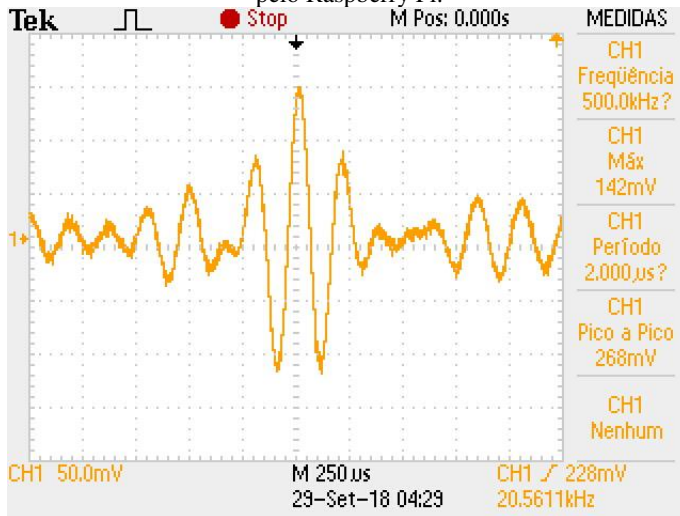
Fonte: O autor (2018).

O espectro observado na Figura 131 corresponde ao espectro em banda base do pulso retangular NRZ deslocado para banda passante. Como o pulso de transmissão é uma função rect no tempo, o espectro é uma função sinc na frequência cujos pontos de nulo ocorrem em frequências múltiplas de  $1/T_s = 2400$  Hz. Assumindo que a largura de banda do sinal transmitido é aproximadamente a banda de frequência entre o primeiro nulo imediatamente antes da frequência central  $f_c$  e o primeiro nulo imediatamente após  $f_c$ , temos que  $B \approx 4800$  Hz.

O modelo do Simulink mostrado na Figura 127 pode ser alterado para fazer com que o sinal BPSK gerado utilize o pulso de transmissão do tipo cosseno levantado. Para isso, basta substituir o bloco **FIR Interpolation** pelo bloco **Raised Cosine Transmit Filter**. A janela de configurações desse bloco foi mostrada anteriormente na Figura 113. Para a obtenção do sinal que será mostrado a seguir, os parâmetros utilizados para o filtro de transmissão foram: fator de *roll-off* igual a 0,8, *Filter span in symbols* igual a 10 e *Output samples per symbol* igual a 20.

Após a troca do pulso de transmissão e o início da execução do experimento, é possível observar a tela do osciloscópio o sinal mostrado na Figura 132.

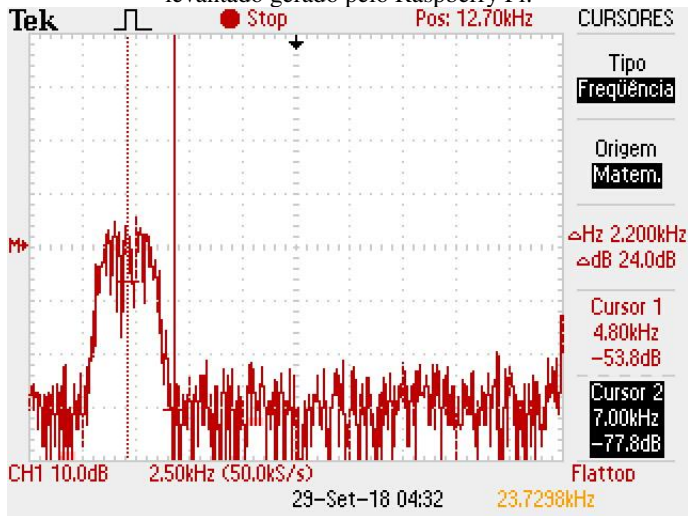
Figura 132 – Sinal BPSK com pulso de transmissão cosseno levantado gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da ferramenta FFT do osciloscópio, é possível observar o espectro do sinal mostrado na Figura 132. Esse espectro, por sua vez, pode ser observado na Figura 133. Dois cursores foram adicionados a esse espectro com o objetivo de permitir a estimação da largura de banda do sinal.

Figura 133 – Espectro do sinal BPSK com pulso de transmissão cosseno levantado gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Assim como no caso dos sinais 2-ASK e 4-ASK, o espectro do sinal BPSK corresponde ao espectro do sinal 2-PAM em banda base transladado para banda passante. A frequência central do espectro em banda passante é 4.8 kHz, que é o valor de frequência da portadora. Além disso, a largura de banda em banda passante é o dobro da largura de banda em banda base: 4.32 kHz.

#### 4.9.2 Geração de um sinal BFSK com o Raspberry Pi

Conforme afirmam Lathi e Ding (2015), quando os dados (digitais) são transmitidos por meio de variações na frequência de uma portadora, temos o caso de chaveamento por deslocamento de frequência (FSK, do inglês *Frequency Shift Keying*). No esquema de modulação BPSK, um bit de informação 0 é transmitido por um pulso de RF com frequência  $f_0$ , e um bit de informação 1 é transmitido por um pulso de RF com frequência  $f_1$ . Assim, a informação sobre o dado transmitido reside na frequência da portadora.

Lathi e Ding (2015) também afirmam que um sinal BPSK pode ser visto como a soma de dois sinais ASK entrelaçados (mais especificamente, dois sinais OOK). Assumindo que o símbolo “0” é

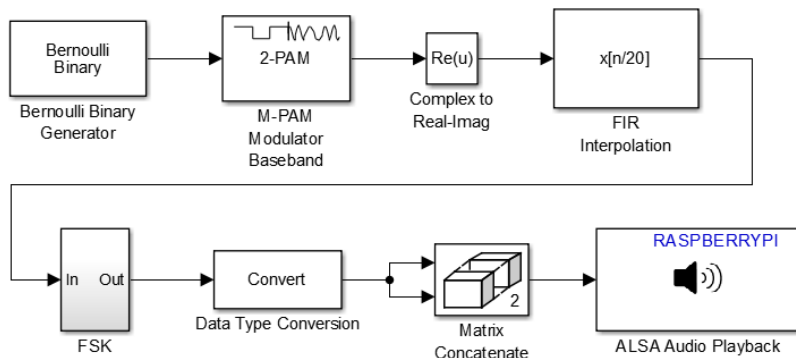
atribuído ao bit 0, e o símbolo “+1” é atribuído ao bit 1, o sinal modulado BFSK pode ser expresso como:

$$s(t) = \sum_{i=-\infty}^{\infty} (1 - a_i)p(t - iT) \cos(2\pi f_0 t) + \sum_{i=-\infty}^{\infty} a_i p(t - iT) \cos(2\pi f_1 t).$$

Na expressão acima,  $a_i$  é a sequência de símbolos transmitidos e  $p(t)$  é o pulso de transmissão com duração  $T$ . O sinal FSK pode ser interpretado então como a superposição de dois sinais OOK com diferentes frequências de portadora.

O modelo do Simulink que pode ser observado na Figura 134 é utilizado para transformar o Raspberry Pi num modulador BFSK. Esse modelo é uma adaptação do proposto no trabalho de Pasolini, Bazzi e Mirabella (2016).

Figura 134 – Modelo do Simulink utilizado para transformar o Raspberry Pi num modulador BFSK.



Fonte: O autor (2018).

Todos os blocos do Simulink que podem ser observados na Figura 134 já foram apresentados nos experimentos anteriores. A principal diferença desse modelo em comparação com os anteriores é a presença do subsistema nomeado como “FSK”, cujos componentes internos e cujo funcionamento serão apresentados adiante.

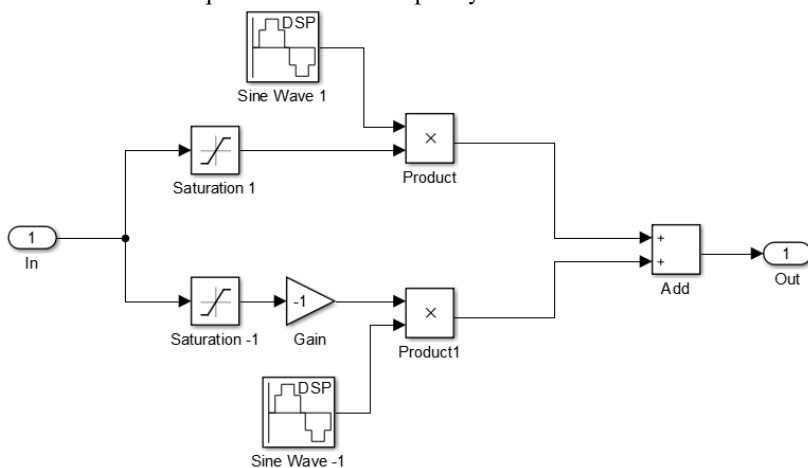
O bloco *Bernoulli Binary Generator* gera uma sequência aleatória de bits a uma taxa  $R_b = 2400$  bits/s. Em seguida, o bloco *M-PAM Modulator Baseband* (configurado com o parâmetro *M-ary number* igual a 2) atribui o símbolo “-1” aos bits 0 e o símbolo “+1” aos bits 1. Como o sinal à saída desse bloco é do tipo complexo (uma sequência de

pontos de uma constelação), o bloco *Complex to Real-Imag* extrai a parte real do sinal.

O bloco *FIR Interpolation* implementa um filtro de transmissão cuja resposta ao impulso é o pulso retangular NRZ. O fator de sobreamostragem (*Interpolation factor*) utilizado neste experimento é igual a 20, isto é, para cada símbolo transmitido é atribuído um pulso de transmissão constituído por 20 amostras. Portanto, os coeficientes do filtro devem ser definidos com o uso da expressão do MATLAB “ones(1,20)”.

Na Figura 135 é possível observar os componentes internos do subsistema FSK. Esse subsistema é o responsável por gerar uma portadora cuja frequência carrega a informação sobre os bits transmitidos.

Figura 135 – Componentes internos do subsistema “FSK”, utilizado no modelo do Simulink que transforma o Raspberry Pi num modulador BFSK.

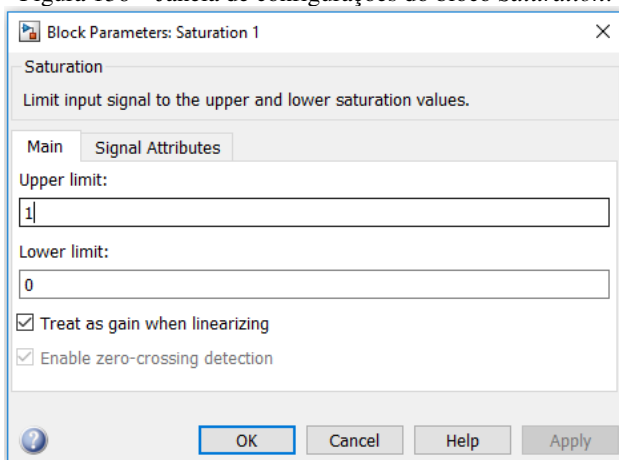


Fonte: O autor (2018).

O sinal à entrada do subsistema “FSK” é uma sequência de pulsos retangulares NRZ. Cada pulso possui a duração de um símbolo, que é igual ao inverso da taxa de bits ( $T_s = 1/R_b = 1/2400$  s). Além disso, a amplitude de cada pulso está relacionada ao bit transmitido: “-1” para o bit 0 e “+1” para o bit 1. Essa sequência de pulsos é utilizada como sinal de entrada para os dois blocos *Saturation* que podem ser observados na Figura 135.

A janela de configurações do bloco *Saturation* pode ser observada na Figura 136. A função desse bloco é limitar o sinal de entrada ao intervalo definido pelos níveis de saturação superior e inferior. Esses níveis de saturação são definidos pelos parâmetros *Upper limit* e *Lower limit*. Em outras palavras, quando a amplitude do sinal de entrada está dentro do intervalo definido pelos limites superior e inferior, o sinal de saída do bloco é igual ao sinal de entrada. Já quando a amplitude do sinal de entrada possui qualquer valor que está fora do intervalo especificado pelos dois limites, o sinal de saída é zero.

Figura 136 – Janela de configurações do bloco *Saturation*.



Fonte: O autor (2018).

O bloco *Saturation* localizado na parte superior da Figura 135 possui o parâmetro *Upper limit* igual a “1” e o parâmetro *Lower limit* igual a “0”. Já o bloco *Saturation* localizado na parte inferior da Figura 135 possui o parâmetro *Upper limit* igual a “0” e o parâmetro *Lower limit* igual a “-1”.

Dessa forma, quando o sinal de entrada do subsistema “FSK” é um pulso com amplitude “+1” (correspondendo a um bit de informação 1), o sinal de saída do bloco *Saturation* superior será igual a 1 e o sinal de saída do bloco *Saturation* inferior será igual a 0. Já quando o sinal de entrada do subsistema “FSK” é um pulso com amplitude “-1” (correspondendo a um bit de informação 0), o sinal de saída do bloco *Saturation* superior será igual a 0 e o sinal de saída do bloco *Saturation*

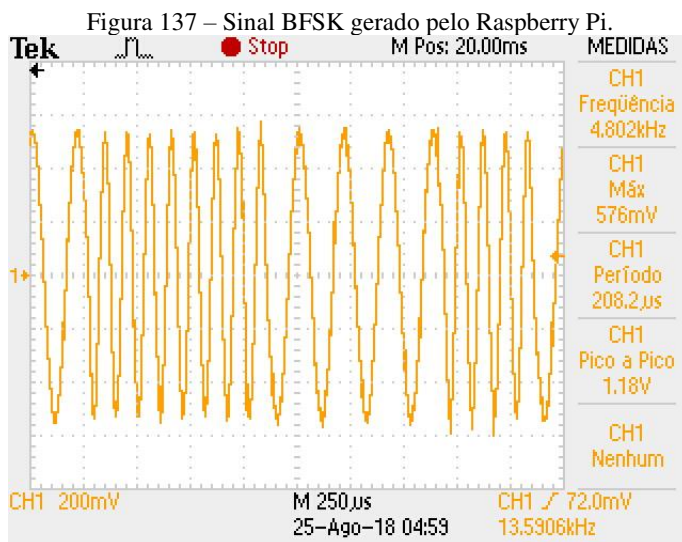


inferior será igual a  $-1$  (em seguida, com o uso de um bloco *Gain*, esse valor é multiplicado por  $-1$ ).

O sinal de saída do bloco *Saturation* superior é utilizado para controlar a amplitude de uma portadora senoidal com frequência  $f_1 = 4800$  Hz, que é gerada pelo bloco *Sine Wave* localizado na parte superior da Figura 135. Esse controle da amplitude é realizado com o uso do bloco *Product*, de forma similar à modulação OOK. Similarmente, o sinal de saída do bloco *Saturation* inferior é utilizado para controlar a amplitude de uma portadora senoidal com frequência  $f_0 = 9600$  Hz, que é gerada pelo bloco *Sine Wave* localizado na parte superior da Figura 135. É importante lembrar que a amplitude das duas senóides deve ser definida como  $2^{15} - 1$ .

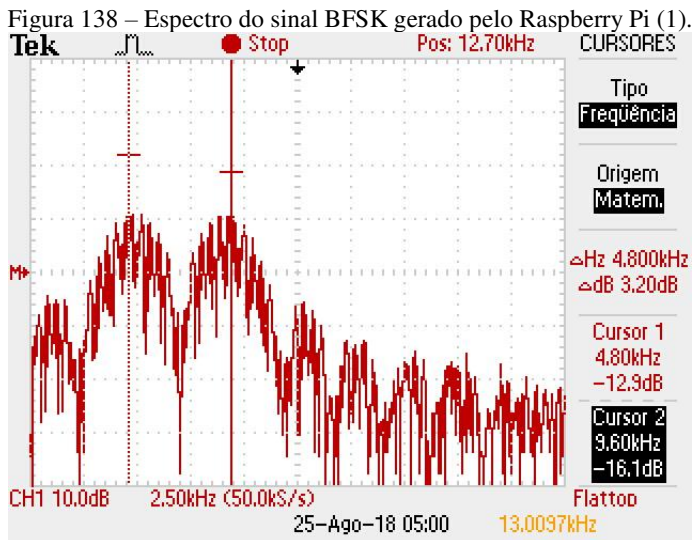
O sinal de saída dos dois blocos *Product* observados na Figura 135 são somados pelo bloco *Add*. O sinal resultante é uma senóide de amplitude constante cuja frequência assume o valor  $f_1 = 4800$  Hz, caso o bit transmitido seja 1, ou  $f_0 = 9600$  Hz, caso o bit transmitido seja 0.

Após o início da execução do experimento, é possível visualizar na tela do osciloscópio o sinal mostrado na Figura 137. Nessa Figura, é possível observar que a frequência da senóide é chaveada de acordo com o bit transmitido.



Fonte: O autor (2018).

Com o uso da ferramenta FFT do osciloscópio, é possível visualizar o espectro do sinal BFSK gerado pelo Raspberry Pi. Esse espectro pode ser visualizado na Figura 138 e na Figura 139. Na Figura 138, dois cursores foram adicionados para indicar as duas frequências centrais do espectro, que correspondem aos dois valores de frequência que a portadora senoidal pode assumir ( $f_0 = 9600$  Hz e  $f_1 = 4800$  Hz).

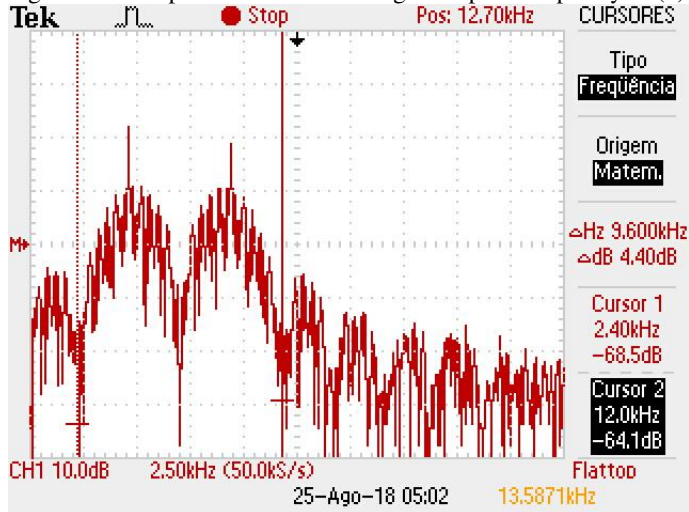


Fonte: O autor (2018).

Analisando o espectro do sinal BFSK mostrado na Figura 138, podemos concluir que ele é a soma de dois espectros ASK: um deles centralizado na frequência  $f_1 = 4800$  Hz, e o outro centralizado na frequência  $f_0 = 9600$  Hz. Como o pulso de transmissão utilizado é o retangular NRZ, que é uma função rect no tempo, é esperado que o espectro do sinal apresente a forma de uma função sinc na frequência. Além disso, como a taxa de símbolos é  $R_s = R_b = 2400$  símbolos/s, é esperado que o espectro possua pontos de nulo em frequências múltiplas de  $1/T_s = 2400$  Hz.

Na Figura 139, cursores foram adicionados em dois pontos de nulo do espectro: no primeiro ponto de nulo imediatamente antes de 4800 Hz e no primeiro ponto de nulo imediatamente após 9600 Hz. Esses cursores são utilizados para a determinação da largura de banda do sinal BFSK.

Figura 139 – Espectro do sinal BFSK gerado pelo Raspberry Pi (2).



Fonte: O autor (2018).

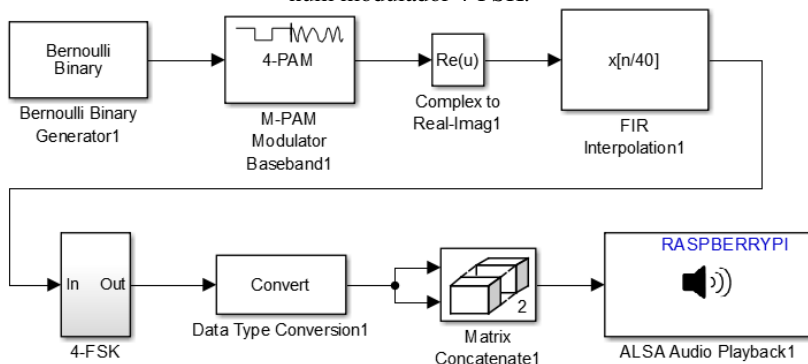
Considerando que a largura de banda do sinal BFSK é aproximadamente a diferença entre as frequências dos dois pontos de nulo indicados na Figura 139, então  $B \approx 9.6$  kHz. Logo, é possível concluir que a largura de banda do sinal BFSK é o dobro da largura de banda do sinal BPSK gerado anteriormente.

#### 4.9.3 Geração de um sinal 4-FSK com o Raspberry Pi

Na modulação 4-FSK, cada dupla de bits de informação a serem transmitidos é atribuída a um pulso de RF de duração  $T_s$  e cuja frequência pode assumir um de quatro valores possíveis.

O modelo do Simulink que pode ser observado na transformadora Raspberry Pi num modulador 4-FSK. Esse modelo é muito semelhante ao utilizado anteriormente para a modulação BFSK, e por isso serão apresentadas apenas as principais diferenças entre cada um deles.

Figura 140 – Modelo do Simulink utilizado para transformar o Raspberry Pi num modulador 4-FSK.



Fonte: O autor (2018).

O bloco *Bernoulli Binary Generator* mostrado na Figura 140 é ser configurado para gerar uma sequência aleatória de bits a uma taxa de 2400 bits/s.

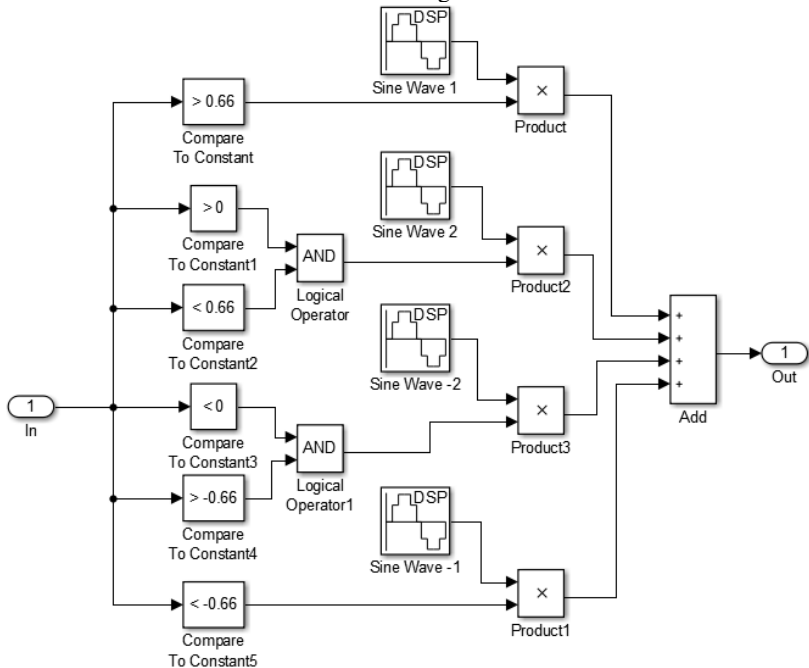
O bloco *M-PAM Modulator Baseband* é configurado com o parâmetro *M-ary number* igual a 4, mapeamento Gray e potência de referência igual a 1 W. Sendo assim, ele atribui um símbolo para cada dupla de bits utilizando o seguinte mapeamento: símbolo “-1” para os bits “00”, símbolo “-1/3” para os bits “01”, símbolo “+1/3” para os bits “11” e símbolo “+1” para os bits “10”. Como cada símbolo transmitido está relacionado a dois bits, a taxa de símbolos é  $T_s = 1200$  símbolos/s.

O bloco *Complex to Real-Imag* tem a função de extrair a parte real dos símbolos à saída do bloco *M-PAM Modulator Baseband*, que são do tipo complexo.

O bloco *FIR Interpolation* implementa um filtro de transmissão do tipo retangular NRZ. O fator de sobreamostragem desse filtro é igual a 40, isto é, a cada símbolo transmitido são atribuídas 40 amostras de sinal. Dessa forma, a taxa de amostragem do sinal à saída desse bloco é de 48 kHz.

O subsistema *4-FSK* que pode ser observado na Figura 140 é o responsável por implementar o chaveamento de frequência característico da modulação 4-FSK. Os componentes internos desse subsistema podem ser observados na Figura 141.

Figura 141 – Componentes internos do subsistema 4-FSK que pode ser observado na Figura 140.



Fonte: O autor (2018).

O subsistema 4-FSK realiza o chaveamento de amplitude de quatro diferentes blocos *Sine Wave* de acordo com o valor de amplitude do sinal de entrada (que é o sinal à saída do bloco *FIR Interpolation*). A amplitude cada bloco *Sine Wave* pode assumir somente os valores 0 ou 1, isto é, cada bloco *Sine Wave* pode estar “desativado” ou “ativado”. Além disso, os quatro blocos são configurados com amplitude igual a  $2^{14} - 1$  e taxa de amostragem de 48 kHz.

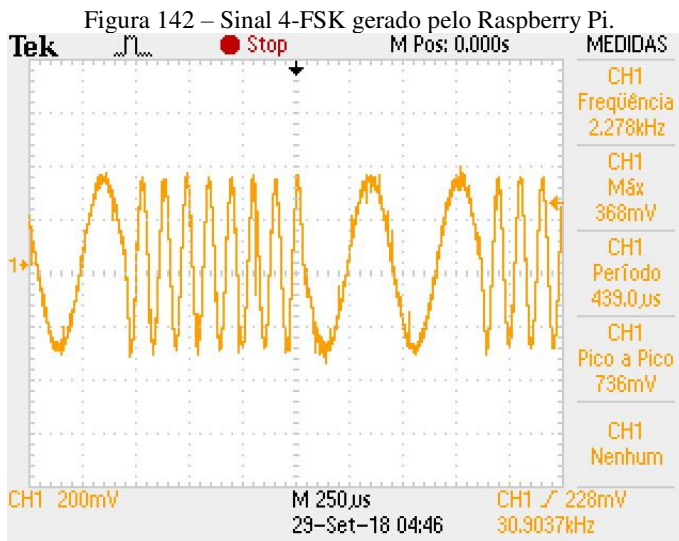
Se a amplitude do sinal de entrada for maior do que 0.66, o primeiro bloco *Sine Wave* de cima para baixo (com frequência de 2.4 kHz) será ativado e os demais desativados. Se a amplitude do sinal de entrada for menor do que 0.66 e maior do que 0, o segundo bloco *Sine Wave* de cima para baixo (com frequência de 2.4 kHz) será ativado e os demais desativados. Se a amplitude do sinal de entrada for menor do que 0 e maior do que  $-0.66$ , o terceiro bloco *Sine Wave* de cima para baixo (com frequência de 7.2 kHz) será ativado e os demais desativados. Por fim, se a amplitude do sinal de entrada for menor do que  $-0.66$ , o quarto

bloco *Sine Wave* de cima para baixo (com frequência de 9.6 kHz) será ativado e os demais desativados. O mapeamento de bits em símbolos e pulsos transmitidos está sumarizado no Quadro 1.

Quadro 1 – Mapeamento de bits em símbolos e pulsos transmitidos para a modulação 4-FSK.

| Bits transmitidos | Símbolo | Frequência do pulso transmitido |
|-------------------|---------|---------------------------------|
| 00                | -1      | 2.4 kHz                         |
| 01                | -1/3    | 4.8 kHz                         |
| 11                | +1/3    | 7.2 kHz                         |
| 10                | +1      | 9.6 kHz                         |

Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 142.

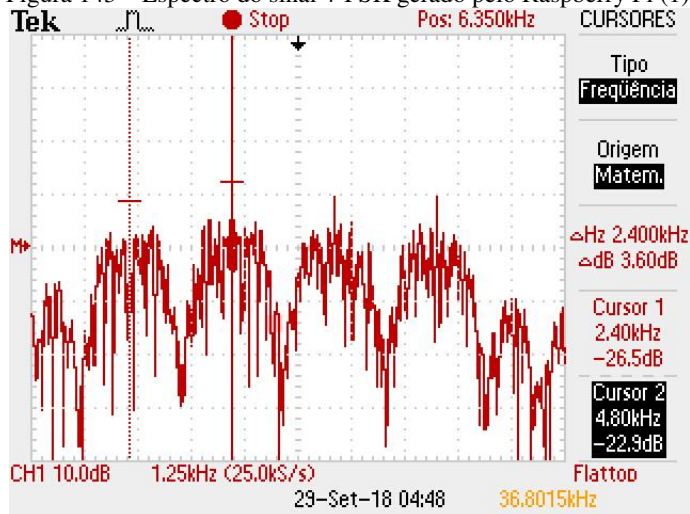


Fonte: O autor (2018).

Utilizando a ferramenta FFT do osciloscópio, é possível observar o espectro do sinal mostrado na Figura 142. Esse espectro, por sua vez, pode ser observado na Figura 143 e na Figura 144. Cursores foram

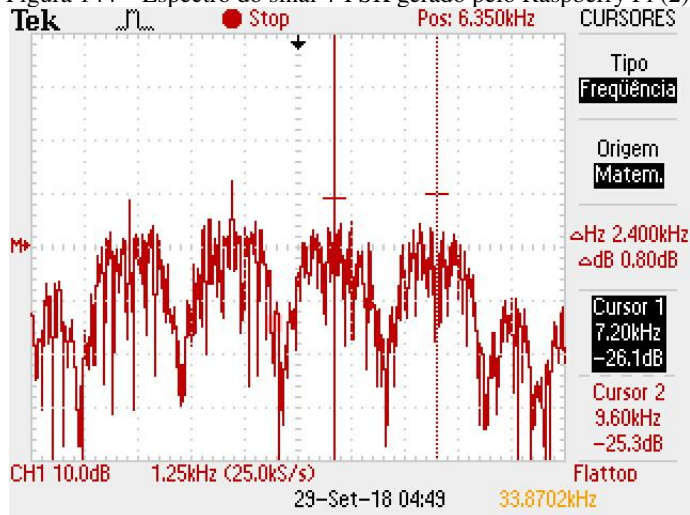
adicionados a essas Figuras com o objetivo de indicarem as frequências centrais dos lóbulos principais.

Figura 143 – Espectro do sinal 4-FSK gerado pelo Raspberry Pi (1).



Fonte: O autor (2018).

Figura 144 – Espectro do sinal 4-FSK gerado pelo Raspberry Pi (2).



Fonte: O autor (2018).

A mesma discussão anterior sobre o espectro do sinal 2-FSK se aplica ao espectro do sinal 4-FSK. Esse novo espectro é a superposição de quatro espectros ASK, centralizados nas frequências de 2.4 kHz, 4.8 kHz, 7.2 kHz e 9.6 kHz. Como o pulso de transmissão utilizado é o retangular NRZ, que é uma função rect no tempo, é esperado que o espectro do sinal apresente a forma de uma função sinc na frequência. Além disso, como a taxa de símbolos é  $R_s = 1200$  símbolos/s, era esperado que o espectro possua pontos de nulo em frequências múltiplas de  $1/T_s = 1200$  Hz.

Por fim, considerando que a largura de banda do sinal 4-FSK gerado é a diferença de frequência entre o ponto de nulo imediatamente antes do lóbulo principal centralizado em 2.4 kHz e o ponto de nulo imediatamente após o lóbulo principal centralizado em 9.6 kHz, temos que  $B \simeq 9.6$  kHz. Logo, a largura de banda do sinal 4-FSK é exatamente igual a do sinal BFSK gerado anteriormente.

## 4.10 MODULAÇÕES PSK E QAM COM O RASPBERRY PI

### 4.10.1 Modulações QPSK e 8-PSK com o Raspberry Pi

O último experimento proposto neste trabalho consiste em transformar o Raspberry Pi num gerador de sinais modulados QPSK e 8-PSK. De acordo com Silva (2017), “a modulação por chaveamento de fase (ou PSK, do inglês *Phase Shift Keying*) consiste em alterar a fase de uma portadora de acordo com o sinal de mensagem digital”. Ainda de acordo com o autor, o sinal modulado é dado por:

$$s(t) = \sum_{k=-\infty}^{\infty} Ag(t - kT) \cos(2\pi f_c t + \varphi[k]).$$

Na expressão acima,  $g(t)$  é o pulso de transmissão com duração  $T$  e  $\varphi[k] \in \{\varphi_0, \varphi_1, \dots, \varphi_{M-1}\}$  é a  $k$ -ésima fase transmitida. Silva (2017) ainda afirma que as  $M$  possíveis fases são normalmente regularmente espaçadas ao longo de um círculo, ou seja:

$$\varphi_i = \varphi_0 + i \frac{2\pi}{M}, i = 1, \dots, M - 1.$$

As duas escolhas mais comuns são  $\varphi_0 = 0$ , para  $M = 2$  (modulação BPSK), e  $\varphi_0 = \pi/M$ , para  $M \geq 4$ .

Utilizando as devidas identidades trigonométricas, o sinal modulado PSK também pode ser expresso como:



$$s(t) = s_I(t) \cos(2\pi f_c t) - s_Q(t) \sin(2\pi f_c t).$$

Na expressão acima:

$$s_I(t) = \sum_{k=-\infty}^{\infty} a_I[k]g(t - kT),$$

$$s_Q(t) = \sum_{k=-\infty}^{\infty} a_Q[k]g(t - kT).$$

Os termos  $\cos(2\pi f_c t)$  e  $\sin(2\pi f_c t)$  correspondem a duas portadoras em quadratura de fase utilizadas para transmissão de dados através de modulação por chaveamento de amplitude. Os termos  $s_I(t)$  e  $s_Q(t)$  são as componentes em fase e em quadratura do sinal modulado, respectivamente. Além disso:

$$a_I[k] = A \cos(\varphi[k])$$

$$a_Q[k] = A \sin(\varphi[k]).$$

Os símbolos a serem transmitidos são complexos, e podem ser expressos de forma compacta como:

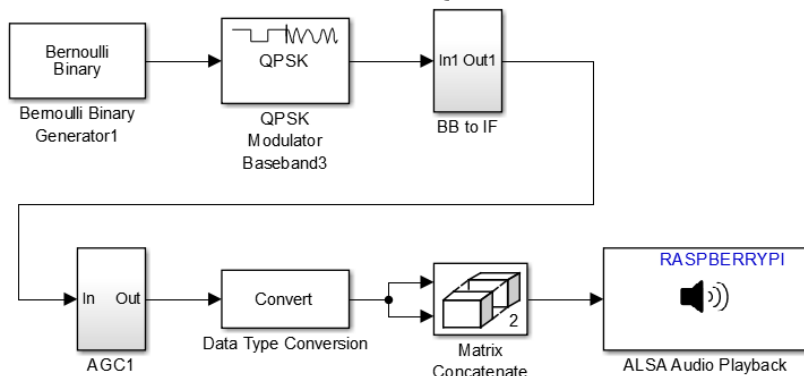
$$a[k] = a_I[k] + ja_Q[k] = Ae^{j\varphi[k]}.$$

Neste experimento, além do pulso de transmissão retangular NRZ utilizados nos experimentos anteriores de transmissão digital em banda passante, também será utilizado o pulso de transmissão do tipo cosseno levantado.

#### 4.10.1.1 Geração de um sinal QPSK com o Raspberry Pi

O modelo do Simulink que pode ser observado na Figura 145 é utilizado para transformar o Raspberry Pi no gerador de um sinal modulado QPSK. Esse modelo é uma adaptação do modelo proposto no trabalho de Pasolini, Bazzi e Mirabella (2016). Alguns dos blocos do Simulink que podem ser observados nessa Figura já foram utilizados nos experimentos anteriores, bem como o subsistema “AGC” (os componentes internos desse subsistema podem ser observados na Figura 102). Os novos componentes (que ainda não foram utilizados neste trabalho) são o bloco do Simulink *QPSK Modulador Baseband* e o subsistema denominado “BB to IF”. Adiante serão apresentadas breves explicações sobre o funcionamento do modelo e os novos componentes.

Figura 145 – Modelo do Simulink que transforma o Raspberry Pi num modulador QPSK.

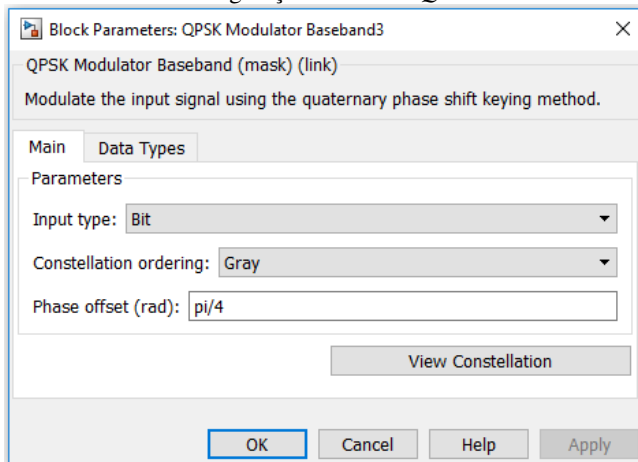


Fonte: O autor (2018).

O bloco *Bernoulli Binary Generator* gera uma sequência aleatória de bits a uma taxa  $R_b = 4800$  bits/s. As amostras do sinal de saída são do tipo *double* e estão agrupadas em quadros de 2000 amostras.

O bloco *QPSK Modulator Baseband* recebe como entrada a sequência de bits e fornece como saída uma sequência de pontos de uma constelação QPSK (a cada dupla de bits é atribuído um símbolo da constelação). Na Figura 146 é possível observar a janela de configurações desse bloco, onde é possível visualizar os parâmetros a serem definidos.

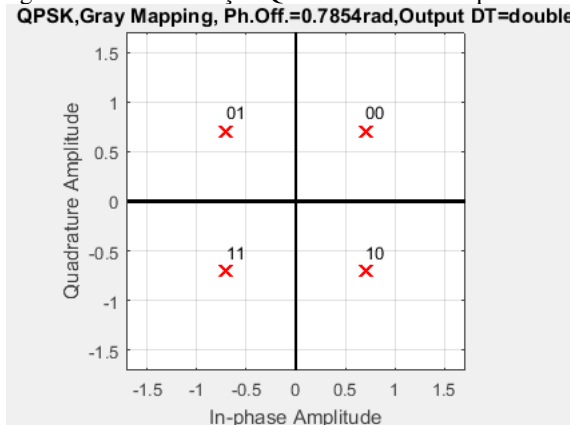
Figura 146 – Janela de configurações do bloco *QPSK Modulator Baseband*.



Fonte: O autor (2018).

O parâmetro *Constellation ordering* deve ser definido como *Gray*, e o *offset* de fase da constelação a ser adotado deve ser de  $\pi/4$  radianos. A constelação QPSK resultante pode ser obtida ao clicar-se no botão *View Constellation*. Essa constelação pode ser observada na Figura 147.

Figura 147 – Constelação QPSK adotada no experimento.



Fonte: O autor (2018).

Conforme pode ser observado na Figura 147, a bloco *QPSK Modulator Baseband* atribui a cada dupla de bits um símbolo complexo. O mapeamento de bits em símbolos está organizado no Quadro 2.

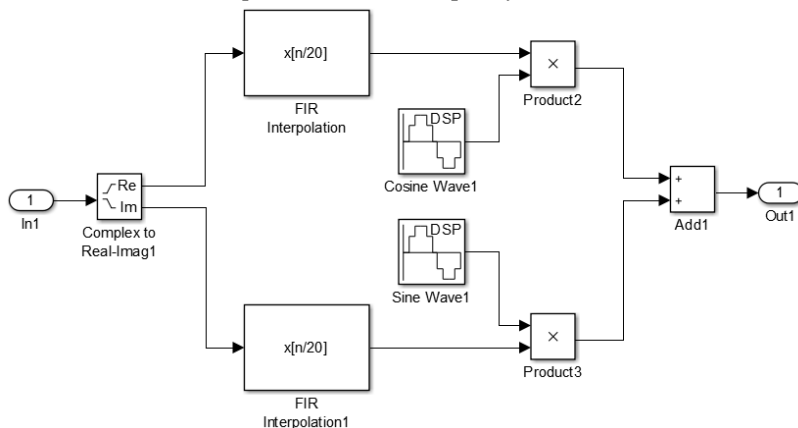
Quadro 2 – Mapeamento de bits em símbolos para a constelação QPSK.

| Bits | Símbolos  |
|------|---|
| 00   | $1e^{j\pi/4} = \frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}}$   |
| 01   | $1e^{j3\pi/4} = -\frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}}$ |
| 11   | $1e^{j5\pi/4} = -\frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}}$ |
| 10   | $1e^{j7\pi/4} = \frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}}$  |

É possível observar que os símbolos da constelação QPSK estão uniformemente espaçados sobre o círculo de raio unitário.

Os componentes internos do subsistema “BB to IF”, presente no modelo do Simulink mostrado anteriormente na Figura 145, podem ser observados na Figura 148.

Figura 148 – Componentes internos do subsistema “BB to IF”, presente no modelo do Simulink que transforma o Raspberry Pi num modulador QPSK.



Fonte: O autor (2018).

A função do subsistema “BB to IF” é extrair as partes real e imaginária de cada símbolo complexo a ser transmitido, associar um pulso de transmissão  $g(t)$  a cada uma das partes, utilizar os pulsos de transmissão para modular a amplitude das portadoras em fase e em quadratura, e por fim somar as componentes em fase e em quadratura para gerar o sinal resultante QPSK a ser transmitido.

Primeiramente, o pulso de transmissão utilizado é o retangular NRZ. Os pulsos de transmissão são gerados com o uso dos blocos *FIR Interpolation* que podem ser observados na Figura 148. O fator de sobreamostragem (*Interpolation factor*) utilizado é igual a 20, isto é, um pulso retangular constituído de 20 amostras é atribuído para as partes real e imaginária de cada símbolo a ser transmitido. Sendo assim, os coeficientes do filtro são definidos com o uso do comando do MATLAB “ones(1,20)”.

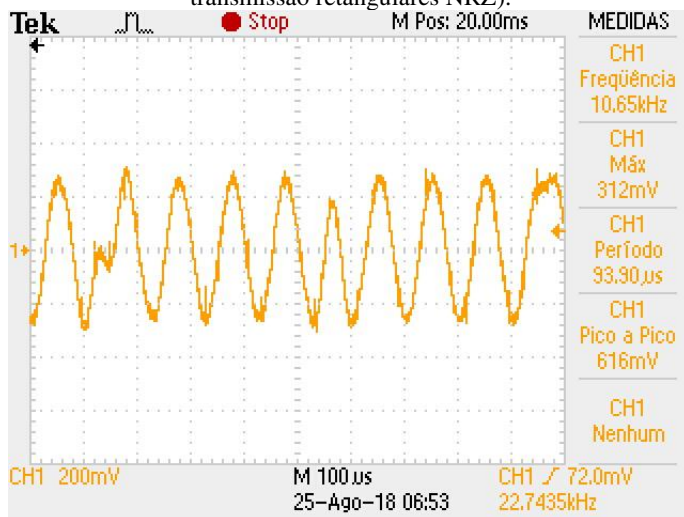
O bloco *Cosine Wave* que pode ser observado na parte superior da Figura 148 é o responsável por gerar a portadora em fase  $\cos(2\pi f_c t)$ . Esse bloco é simplesmente um bloco *Sine Wave* configurado para gerar um sinal senoidal com amplitude unitária, frequência  $f_c = 10$  kHz e fase de  $\pi/2$  radianos.

O bloco *Sine Wave* que pode ser observado na parte inferior da Figura 148, por sua vez, é o responsável por gerar a portadora em quadratura  $-\sin(2\pi f_c t)$ . Esse bloco é simplesmente um bloco *Sine Wave* configurado para gerar um sinal senoidal com amplitude unitária, frequência  $f_c = 10$  kHz e fase de  $\pi$  radianos.

Por fim, no subsistema “AGC”, o valor do ganho aplicado ao sinal pelo bloco *Gain* deve ser definido como  $2^{14} - 1$ .

Após o início da execução do experimento, é possível visualizar na tela do osciloscópio o sinal que pode ser observado na Figura 149. Nesta imagem é possível observar algumas transições de fase de  $\pi/4$  no sinal QPSK gerado.

Figura 149 – Sinal QPSK gerado pelo Raspberry Pi (com o uso de pulsos de transmissão retangulares NRZ).

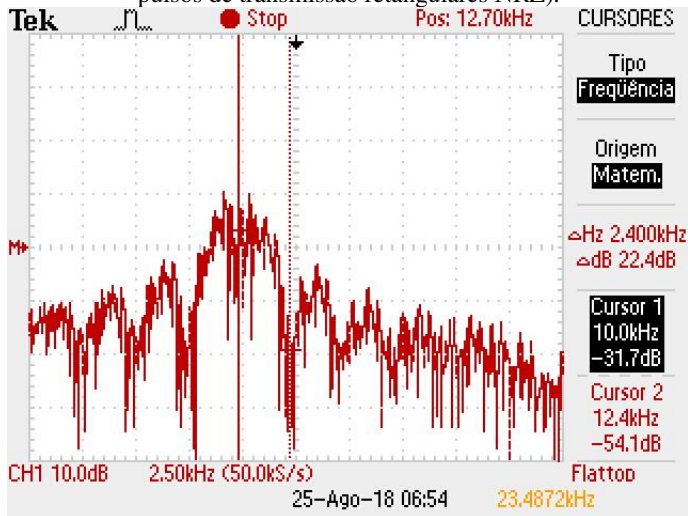


Fonte: O autor (2018).

Utilizando a função FFT do osciloscópio, é possível visualizar o espectro do sinal QPSK gerado pelo Raspberry Pi. Esse espectro pode ser observado na Figura 150. Dois cursores presentes nessa Figura indicam a frequência central do espectro e um ponto de nulo imediatamente acima dessa frequência. É possível observar que o espectro do sinal QPSK corresponde ao espectro do pulso retangular transladado para banda passante, com frequência central igual à frequência das portadoras em quadratura de fase ( $f_c = 10 \text{ kHz}$ ).

Conforme mencionado anteriormente, a taxa de bits do sistema é  $R_b = 4800 \text{ bits/s}$ . Como cada símbolo da constelação QPSK está associado a dois bits, temos que a taxa de símbolos é  $R_s = 2400 \text{ símbolos/s}$ . Um pulso de transmissão é atribuído às partes real e imaginária de cada símbolo. Logo, cada pulso possui duração  $T = 1/2400 \text{ s}$ . Portanto, o espectro é uma função sinc na frequência cujos pontos de nulo ocorrem em frequências múltiplas de  $1/T = 2400 \text{ Hz}$ . Considerando que a largura de banda do sinal QPSK é aproximadamente a banda entre os dois pontos de nulo em torno da frequência central, temos que  $B \approx 4800 \text{ Hz}$ .

Figura 150 – Espectro do sinal QPSK gerado pelo Raspberry Pi (com uso de pulsos de transmissão retangulares NRZ).

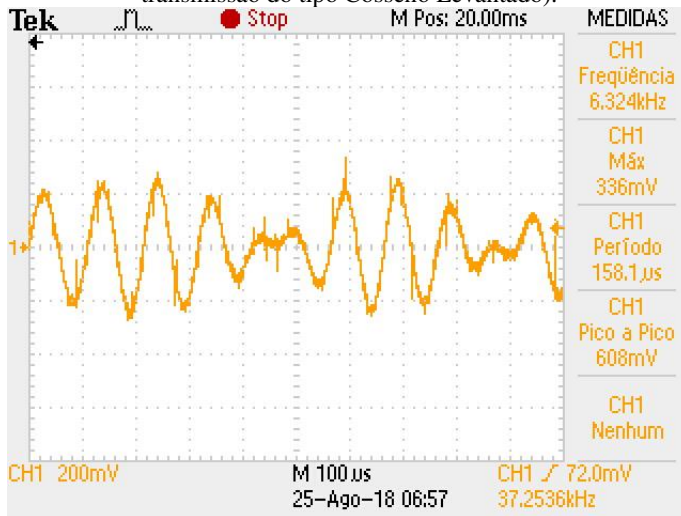


Fonte: O autor (2018).

O modelo do Simulink pode ser modificado para que o pulso de transmissão  $g(t)$  utilizado passe a ser o do tipo Cosseno Levantado. Para isso, basta substituir o bloco *FIR Interpolation* presente no subsistema “BB to IF” pelo bloco *Raised Cosine Transmit Filter*. Os parâmetros de configurações a serem utilizados são os mesmos que podem ser observados na Figura 113: fator de *roll-off* igual a 0,8, duração dos pulsos (parâmetro *Filter span in symbols*) igual a 10 símbolos e fator de sobreamostragem (definido pelo parâmetro *Output samples per symbol*) igual a 20.

Após essa alteração e o reinício da execução do experimento, é possível visualizar na tela do osciloscópio o sinal mostrado na Figura 151.

Figura 151 – Sinal QPSK gerado pelo Raspberry Pi (com o uso de pulsos de transmissão do tipo Cosseno Levantado).

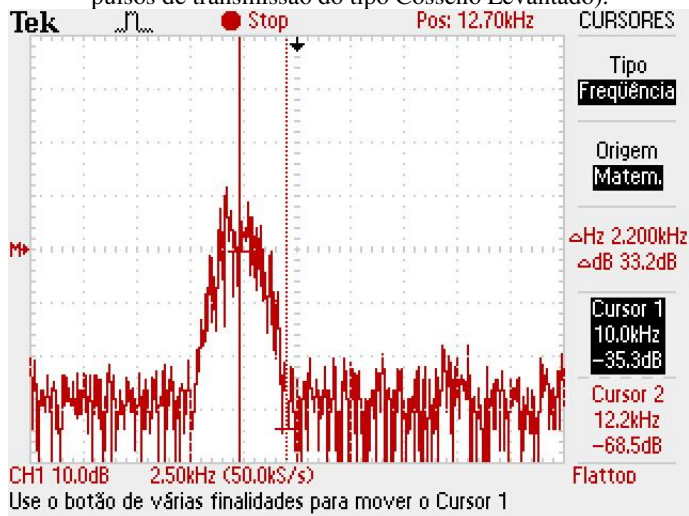


Fonte: O autor (2018).

Novamente utilizando a função FFT do osciloscópio, é possível visualizar o espectro do sinal QPSK gerado pelo Raspberry Pi. Esse espectro pode ser observado na Figura 152. Conforme pode ser observado nessa Figura, o espectro do sinal constituído por pulsos do tipo Cosseno Levantado é limitado em frequência, diferentemente do espectro do sinal constituído por pulsos retangulares.



Figura 152 – Espectro do sinal QPSK gerado pelo Raspberry Pi (com uso de pulsos de transmissão do tipo Cosseno Levantado).



Fonte: O autor (2018).

No espectro mostrado na Figura 152, é possível observar dois cursores. Um deles indica a frequência central do espectro, que é a frequência das portadoras ortogonais  $f_c = 10$  kHz. O outro indica o limiar do espectro, e está localizado em 12.2 kHz.

Sendo  $\alpha$  o fator de *roll-off* e  $T$  a duração do pulso, a largura de banda do pulso Cosseno Levantado (em banda base) é dada por:

$$B = (1 + \alpha) \frac{1}{2T}$$

No experimento, o fator de *roll-off* utilizado é 0.8, e cada pulso de transmissão possui período igual ao inverso da taxa de símbolos, isto é,  $T = 1/2400$  s. Portanto, o valor teórico da largura de banda (em banda base) do sinal QPSK gerado no experimento é dado por:

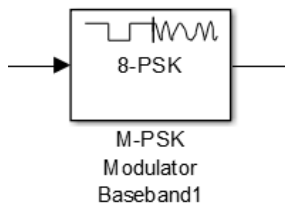
$$B = (1 + 0.8) \frac{1}{2(1/2400 \text{ s})} = 0.9(2400 \text{ Hz}) = 2160 \text{ Hz}.$$

Em banda passante, a largura de banda do sinal QPSK é o dobro do valor acima, isto é,  $B = 4320$  Hz. Esse valor está de acordo com o que pode ser observado na Figura 152.

#### 4.10.1.2 Geração de um sinal 8-PSK com o Raspberry Pi

O modelo do Simulink que transforma o Raspberry Pi num modulador QPSK, observado anteriormente na Figura 145, pode ser alterado para transformar o Raspberry Pi num modulador M-PSK genérico. Para isso, uma única modificação é necessária: a substituição do bloco *QPSK Modulator Baseband* pelo bloco *M-PSK Modulator Baseband*. Esse novo bloco pode ser observado na Figura 153.

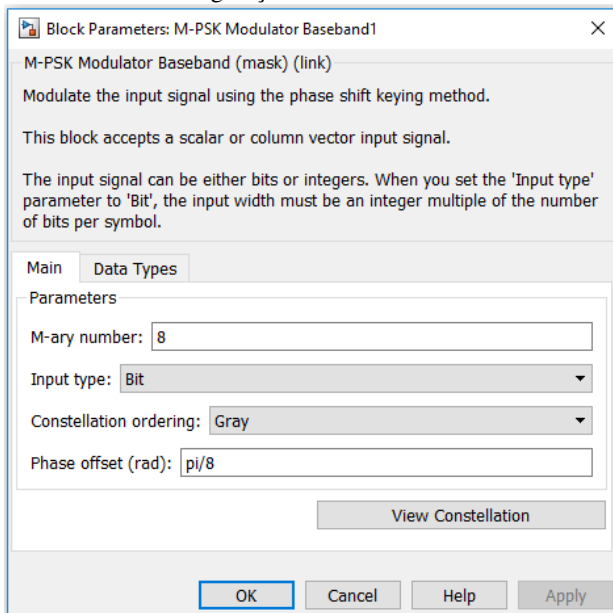
Figura 153 – Bloco *M-PSK Modulator Baseband*, utilizado no modelo do Simulink que transforma o Raspberry Pi num modulador M-PSK.



Fonte: O autor (2018).

A janela de configurações do bloco *M-PSK Modulator Baseband* pode ser observada na Figura 154. Para transformar o Raspberry Pi num modulador 8-PSK, o parâmetro *M-ary number* deve ser definido como 8. Além disso, o *offset* de fase da constelação deve ser definido como  $\pi/8$ .

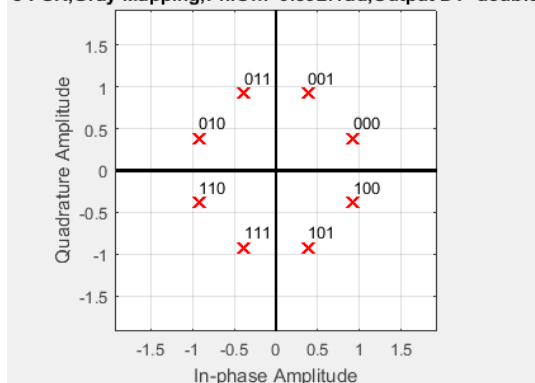
Figura 154 – Janela de configurações do bloco *M-PSK Modulator Baseband*.



Fonte: O autor (2018).

Utilizando a configuração de parâmetros mostrada na Figura 154, a constelação 8-PSK resultante é obtida ao clicar-se no botão *View Constellation*. Essa constelação pode ser observada na Figura 155.

Figura 155 – Constelação 8-PSK adotada no experimento.  
8-PSK, Gray Mapping, Ph. Off.=0.3927rad, Output DT=double



Fonte: O autor (2018).

Conforme pode ser observado na Figura 155, o modulador 8-PSK em banda base atribui um símbolo complexo a cada três bits de informação, utilizando o mapeamento Gray. Esse mapeamento está organizado no Quadro 3.

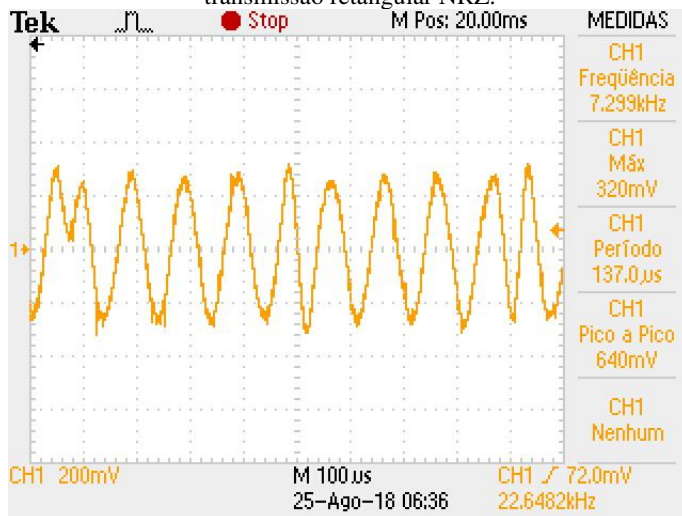
Quadro 3 – Mapeamento de bits em símbolos para a modulação 8-PSK.

| <b>Bits</b> | <b>Símbolo</b>                      |
|-------------|-------------------------------------|
| 000         | $1e^{j\pi/8} = 0.9239 + j0.3827$    |
| 001         | $1e^{j3\pi/8} = 0.3827 + j0.9239$   |
| 011         | $1e^{j5\pi/8} = -0.3827 + j0.9239$  |
| 010         | $1e^{j7\pi/8} = -0.9239 + j0.3827$  |
| 110         | $1e^{j9\pi/8} = -0.9239 - j0.3827$  |
| 111         | $1e^{j11\pi/8} = -0.3827 - j0.9239$ |
| 101         | $1e^{j13\pi/8} = 0.3827 - j0.9239$  |
| 100         | $1e^{j15\pi/8} = 0.9239 - j0.3827$  |

Para a execução do experimento, a taxa de bits gerados pelo bloco *Bernoulli Binary Generator* deve ser alterada para  $R_b = 7200$  bits/s. Dessa forma, a taxa de símbolos passa a ser  $R_s = 2400$  símbolos/s. As configurações dos demais blocos são iguais às utilizadas anteriormente no experimento da modulação QPSK.

O primeiro pulso de transmissão adotado é o retangular NRZ (implementado pelo bloco *FIR Interpolation*, com fator de sobreamostragem igual a 20). Após o início da execução do experimento, é possível observar na tela do osciloscópio o sinal mostrado na Figura 156.

Figura 156 – Sinal 8-PSK gerado pelo Raspberry Pi, com o uso do pulso de transmissão retangular NRZ.

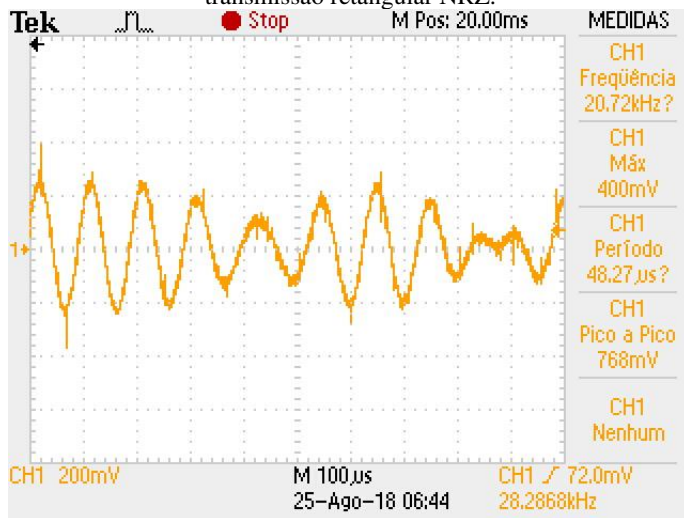


Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível visualizar o espectro do sinal 8-PSK gerado pelo Raspberry Pi observado na Figura 156. Esse espectro é exatamente igual ao espectro do sinal QPSK gerado com pulsos de transmissão retangulares, mostrado anteriormente na Figura 150.

Após a execução da etapa anterior, o pulso de transmissão é alterado para ser do tipo Cosseno Levantado. Isso é feito substituindo-se o bloco *FIR Interpolation* pelo bloco *Raised Cosine Transmit Filter*, com as mesmas configurações utilizadas no experimento da modulação QPSK. Após o início da execução do experimento com essa alteração, é possível observar na tela do osciloscópio o sinal mostrado na Figura 157.

Figura 157 – Sinal 8-PSK gerado pelo Raspberry Pi, com o uso do pulso de transmissão retangular NRZ.



Com o uso da função FFT do osciloscópio, é possível visualizar o espectro do sinal 8-PSK gerado pelo Raspberry Pi observado na Figura 157. Esse espectro é exatamente igual ao espectro do sinal QPSK gerado com pulsos de transmissão do tipo Cosseno Levantado, mostrado anteriormente na Figura 152.

Após a execução dos experimentos, é possível concluir que a modulação 8-PSK permite transmitir dados a uma taxa de bits maior do que a modulação QPSK ocupando uma mesma largura de banda. O custo disso é que a modulação 8-PSK possui uma maior probabilidade de erro de símbolo do que a modulação QPSK, pois na primeira a distância entre os pontos consecutivos da constelação é menor.

Com as devidas alterações nos modelos do Simulink utilizados, é possível observar experimentalmente que, para a mesma taxa de bits, a modulação 8-PSK ocupa uma largura de banda menor do que a modulação QPSK.

#### 4.10.2 Geração de um sinal 16-QAM com o Raspberry Pi

A modulação em amplitude em quadratura (QAM, do inglês *Quadrature Amplitude Modulation*), assim como a modulação PSK apresentada anteriormente, também utiliza duas portadoras em

quadratura de fase. Além disso, da mesma forma que na modulação PSK, a transmissão da informação é realizada através do chaveamento da amplitude dessas duas portadoras.

Dessa forma, o sinal M-QAM também pode ser expresso por:

$$s(t) = s_I(t) \cos(2\pi f_c t) - s_Q(t) \sin(2\pi f_c t).$$

Na expressão acima:

$$s_I(t) = \sum_{k=-\infty}^{\infty} a_I[k]g(t - kT),$$

$$s_Q(t) = \sum_{k=-\infty}^{\infty} a_Q[k]g(t - kT).$$

Os termos  $\cos(2\pi f_c t)$  e  $\sin(2\pi f_c t)$  correspondem a duas portadoras em quadratura de fase utilizadas para transmissão de dados através de modulação por chaveamento de amplitude. Os termos  $s_I(t)$  e  $s_Q(t)$  são as componentes em fase e em quadratura do sinal modulado, respectivamente.

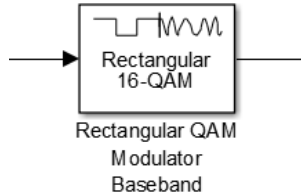
Os símbolos a serem transmitidos são complexos, e podem ser expressos de forma compacta como:

$$a[k] = a_I[k] + ja_Q[k] = Ae^{j\phi[k]}.$$

Na modulação PSK a informação a ser transmitida está contida apenas na fase do símbolo transmitido, enquanto que na modulação QAM a informação a ser transmitida está contida na fase e na amplitude do símbolo transmitido. Esse fato se reflete na forma das constelações de símbolos de cada uma das modulações. A constelação de símbolos da modulação PSK possui a forma de um círculo, enquanto que a constelação de símbolos da QAM possui uma forma quadrada ou retangular.

O modelo do Simulink utilizado para transformar o Raspberry Pi num modulador M-QAM é muito semelhante ao utilizado anteriormente para transformá-lo num modulador M-PSK. A única diferença consiste na substituição do bloco *M-PSK Modulator Baseband* pelo bloco *Rectangular QAM Modulator Baseband*. Esse novo bloco pode ser observado na Figura 158.

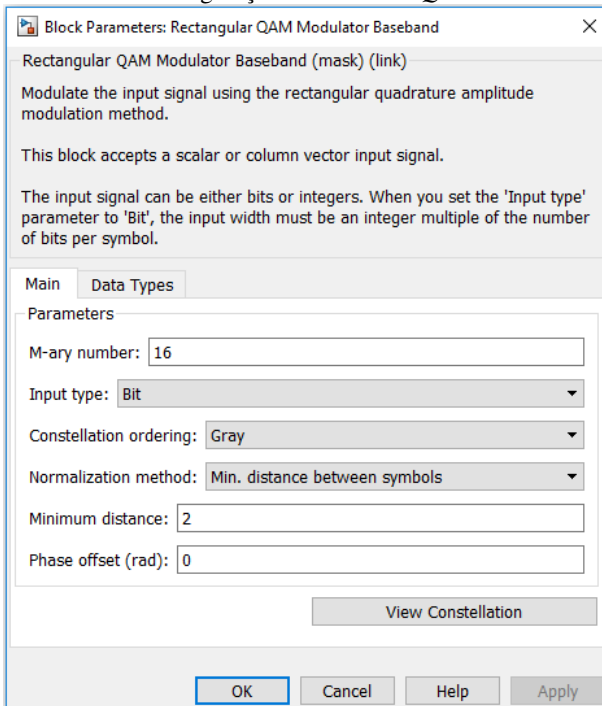
Figura 158 – Bloco *Rectangular QAM Modulator Baseband*, utilizado no modelo do Simulink que transforma o Raspberry Pi num modulador M-QAM.



Fonte: O autor (2018).

A janela de configurações do bloco *M-QAM Modulator Baseband* pode ser observada na Figura 159. Para a execução do exemplo de experimento que será mostrado a seguir, o parâmetro *M-ary number* a ser utilizado é igual a 16, e a distância mínima entre símbolos é igual a 2.

Figura 159 – Janela de configurações do bloco *M-QAM Modulator Baseband*.

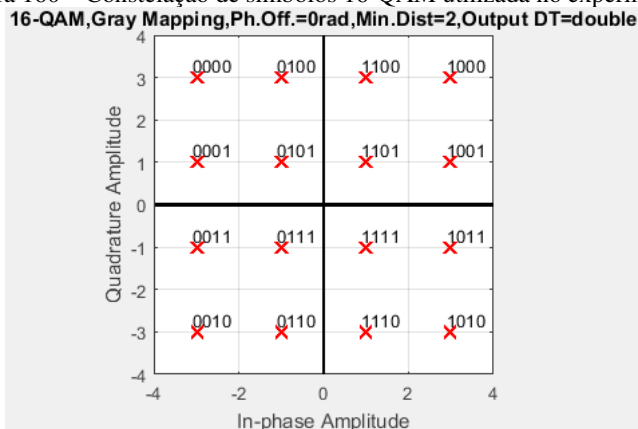


Fonte: O autor (2018).



Ao clicar-se no botão *View Constellation* que pode ser observado na Figura 159, é possível observar a constelação de símbolos resultante. Essa constelação, por sua vez, pode ser observada na Figura 160.

Figura 160 – Constelação de símbolos 16-QAM utilizada no experimento.



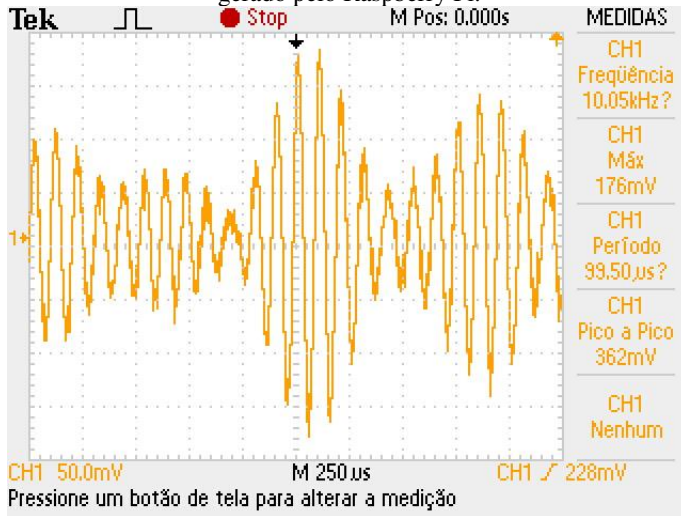
Fonte: O autor (2018).

Conforme pode ser observado na Figura 160, um símbolo complexo é atribuído a cada conjunto de quatro bits de informação a serem transmitidos. A parte real do símbolo corresponde a amplitude da portadora em fase, enquanto que a parte imaginária do símbolo corresponde a amplitude da portadora em quadratura. Como o valor adotado  $M = 16$  é um múltiplo de 4, a constelação é quadrada.

Para a execução do experimento que transforma o Raspberry Pi num modulador 16-QAM, a taxa de bits gerados pelo bloco *Bernoulli Binary Generator* foi alterada para 9600 bits/s. Dessa forma, a taxa de símbolos é de  $R_s = 2400$  símbolos/s. As configurações dos demais blocos são iguais às utilizadas anteriormente nos experimentos das modulações QPSK e 8-PSK. Para a obtenção do sinal que será mostrado na sequência, foi adotado o pulso de transmissão do tipo cosseno levantado.

Após o início da execução do experimento, foi possível observar na tela do osciloscópio o sinal mostrado na Figura 161.

Figura 161 – Sinal 16-QAM com pulso de transmissão cosseno levantado gerado pelo Raspberry Pi.



Fonte: O autor (2018).

Com o uso da função FFT do osciloscópio, é possível observar o espectro do sinal 16-QAM gerado pelo Raspberry Pi. Esse espectro, por sua vez, é igual aos espectros dos sinais QPSK e 8-PSK gerados anteriormente, e pode ser observado na Figura 152. Esse resultado era o esperado, visto que o pulso de transmissão utilizado e a taxa de símbolos se mantiveram inalterados.

## 5 CONCLUSÃO

O principal objetivo proposto no início deste trabalho era replicar a plataforma didática baseada no Raspberry Pi proposta por Pasolini, Bazzi e Mirabella (2016), reproduzir os experimentos propostos por esses autores, e propor novos experimentos (sendo alguns desses adaptações dos já existentes). Esse objetivo foi cumprido com sucesso.

Os experimentos desenvolvidos neste trabalho podem ser aplicados em disciplinas introdutórias de sinais e sistemas, comunicações e processamento digital de sinais. Diferentemente da grande maioria dos experimentos didáticos usualmente adotados nessas disciplinas, baseados em simulações computacionais, os experimentos que utilizam a plataforma didática baseada em Raspberry Pi permitem que sinais reais sejam gerados e mensurados com o uso de instrumentos reais. Essa abordagem inovadora tem o potencial de aumentar a motivação dos alunos que estão cursando essas disciplinas, e consequentemente aprimorar o processo de aprendizado.

A grande maioria dos laboratórios de engenharia elétrica e eletrônica em universidades já dispõe de recursos como osciloscópios, geradores de funções e bons computadores com licenças válidas dos softwares MATLAB e Simulink. Os demais recursos que compõe a plataforma didática (o Raspberry Pi, a placa de som USB, os cabos e adaptadores diversos) podem ser adquiridos a custos razoáveis, o que permite que a plataforma seja replicada em escala para utilização por turmas em aulas práticas nesses laboratórios.

Após a aquisição de todos os recursos necessários, as etapas para instalação dos pacotes de suporte ao hardware do Raspberry Pi nos softwares MATLAB & Simulink, instalação do sistema operacional Raspbian no Raspberry Pi e configuração da placa de som USB constituem algumas das etapas mais desafiadoras do trabalho. Isso porque requerem um grande número de etapas a serem executadas, uma boa conexão com a internet e algumas horas a serem dispendidas. Felizmente, essas etapas só precisam ser realizadas uma única vez.

A execução do primeiro experimento proposto nesse trabalho é extremamente importante, pois esse experimento permite que todos os recursos de hardware e software da plataforma sejam testados para verificação de seu funcionamento adequado. Além disso, nesse experimento os estudantes realizam procedimentos no Simulink para a complicação do modelo e execução desse experimento que também deverão ser replicados em todos os demais experimentos.

O segundo experimento, que tem como objetivo transformar o Raspberry Pi num gerador de funções, também é importante para a execução dos demais experimentos. Nesse experimento, os estudantes tem a oportunidade de aprender a utilizar a função matemática FFT do osciloscópio, que permite a visualização de sinais no domínio da frequência. Essa ferramenta é amplamente utilizada nos experimentos seguintes. Mais ainda, apesar das limitações de amplitude e frequência impostas pela placa de som USB, esse experimento mostra que o Raspberry Pi pode ser eventualmente utilizado para substituir um gerador de funções em outras aplicações, com a vantagem de possibilitar a geração de qualquer forma de onda periódica arbitrária.

Os demais experimentos propostos neste trabalho (sobre temas como filtragem digital, quantização, modulações analógicas, modulação digital em banda base e modulação digital em banda passante) permitem que estudantes obtenham *insights* que frequentemente não conseguem obter somente com o uso de livros-texto e simulações computacionais.

O experimento sobre modulação AM permite que os estudantes visualizem em sinais reais a propriedade do deslocamento do espectro da Transformada de Fourier. Já o experimento sobre modulação FM permite que os estudantes visualizem a complexidade do espectro de um sinal FM real.

Os experimentos sobre modulações digitais permitem que os estudantes visualizem o impacto que a taxa de bits e a escolha de um pulso de transmissão têm sobre o espectro de sinais modulados reais. Eles poderão visualizar, por exemplo, que um aumento da taxa de bits acarreta um aumento da banda ocupada pelo sinal transmitido, e que um pulso de transmissão retangular (por ser limitado no tempo) faz com que o sinal modulado ocupe uma banda muito maior do que um sinal gerado com o uso do pulso do tipo Cosseno Levantado.

É importante lembrar que os experimentos didáticos propostos neste trabalho são complementares, isto é, não substituem os experimentos tradicionais que utilizam simulações computacionais.

O autor espera que os experimentos desenvolvidos para este trabalho possam ser utilizados com sucesso em aulas práticas dos cursos de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina, onde o trabalho foi desenvolvido, e também em quaisquer outros cursos de Engenharia Elétrica e Eletrônica no Brasil e no exterior. Mais ainda, o autor deseja que esse trabalho seja um material de referência e motivação para o desenvolvimento de outros trabalhos semelhantes. Pasolini, Bazzi e Mirabella (2016) propuseram um experimento didático que transforma o Raspberry Pi num modulador

OFDM. Esse experimento é complexo demais para ser utilizado em disciplinas introdutórias, porém é um exemplo de que a plataforma didática também poderia ser utilizada em aulas práticas de disciplinas mais avançadas, que tratam de temas como Comunicações Móveis.

Os experimentos sobre modulações analógicas e digitais desenvolvidos neste trabalho apenas compreendiam a etapa de modulação. Uma proposta para continuação deste trabalho seria desenvolver os experimentos correspondentes para a demodulação dos sinais gerados. Assim, duas placas Raspberry Pi seriam utilizadas simultaneamente: uma para a geração dos sinais modulados e outra para a demodulação. A comunicação entre as placas poderia ser feita por um cabo P2-P2 interligando a saída analógica da placa de som conectada ao Raspberry modulador à entrada analógica da placa de som conectada ao Raspberry demodulador. Nos modelos do Simulink utilizados, algumas imperfeições poderiam ser inseridas para que os estudantes pudessem observar seus efeitos, como a adição de ruído ou o descasamento entre as frequências das portadoras do modulador e do demodulador.

Nos experimentos propostos por Pasolini, Bazzi e Mirabella (2016) e nos apresentados neste trabalho, a geração de código executável automática a partir de modelos do Simulink (diagramas de blocos) foi utilizada. Esse paradigma de geração de código mostrou ser uma ferramenta extremamente útil do ponto de vista didático, pois permite que sistemas de comunicação relativamente complexos possam ser implementados de forma rápida e fácil por estudantes que não tem grande conhecimento ou experiência com linguagens de programação. Todavia, um trabalho futuro que abordasse experimentos semelhantes com a implementação dos sistemas diretamente no Raspberry Pi (isto é, sem o uso do computador com os softwares MATLAB e Simulink instalados) utilizando alguma linguagem de programação também seria extremamente interessante. Por exemplo, com o uso de uma linguagem de programação de baixo nível, os estudantes poderiam visualizar como um pulso de transmissão do tipo Cosseno Levantado é realmente gerado com o uso de um filtro FIR. Esses experimentos poderiam ser utilizados em disciplinas que tivessem como foco específico o ensino de programação para Sistemas Embarcados.



## REFERÊNCIAS

- ADAFRUIT. **Raspberry Pi Analog to Digital Converters**. Disponível em: <<https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/overview>>. Acesso em: 21 de março de 2018.
- BAZZI, A; PASOLINI, G; ZABINI, F. **A Raspberry Pi-Based Platform for Signal Processing Education**. *IEEE Signal Processing Magazine*, Julho de 2017, p. 151-158.
- BOURDON, Bastien. **Quickly setup a USB sound card on a Raspberry Pi 3**. [Online]. Disponível em: <<https://bastienbourdon.com/2016/raspberrypi-usb-sound/>>. Acesso em: 21 jun. 2018.
- GOLDBERGER, A. L.; AMARAL, L. A.; GLASS, L.; HAUSDORFF, J. M.; IVANOV, P. C.; MARK, R. G.; MIETUS, J. E.; MOODY, G. B.; PENG, C. K.; STANLEY, H. E. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. **Circulation**, v. 101, n. 23, p. E215-20, 13 jun. 2000.
- LATHI, B. P; DING, Z. **Sistemas de Comunicações Analógicas e Digitais Modernos**. 4. ed. Rio de Janeiro: LTC, 2015.
- MATHWORKS. **Raspberry Pi Support from Mathworks**. [Online]. Disponível em: <<https://www.mathworks.com/hardware-support/raspberry-pi-simulink.html>>. Acesso em: 18 de março de 2018.
- PASOLINI, G; BAZZI, A; MIRABELLA, M. **Raspberry Pi based SDR experiences**. Disponível em: <<http://www.simulinkdefinedradio.com/>>. Acesso em: 18 de março de 2018.
- PASOLINI, G; BAZZI, A; MIRABELLA, M. **Simulink Defined Radio with Raspberry Pi2**. (2016, online) Disponível em: <<http://www.simulinkdefinedradio.com/down/>>. Acesso em: 18 de março de 2018.
- RASPBERRY PI FOUNDATION. **Raspberry Pi**. Disponível em: <<https://www.raspberrypi.org/learning/hardware-guide/components/raspberry-pi/>>. Acesso em: 18 de março de 2018.

RASPBERRY PI FOUNDATION. **Raspberry Pi Software Guide**. Disponível em: <<https://www.raspberrypi.org/learning/software-guide/quickstart/>>. Acesso em: 18 de março de 2018.

RASPBERRY PI FOUNDATION. **SD Card**. Disponível em: <<https://www.raspberrypi.org/learning/hardware-guide/components/noobs-card/>>. Acesso em: 18 de março de 2018.

RASPBERRY PI FOUNDATION. **Raspberry Pi Power Supply**. Disponível em: <<https://www.raspberrypi.org/learning/hardware-guide/components/power-supply/>>. Acesso em: 18 de março de 2018.

SILVA, Danilo. **Transmissão Digital em Banda Passante**. Disciplina EEL7062 – Princípios de Sistemas de Comunicações. (2017). Notas de aula. Universidade Federal de Santa Catarina.

STEWART, R. W. *et al.* **Software Defined Radio Using MATLAB & Simulink**, 1. ed. Glasgow, Scotland, UK: 2015. p. 1-11.

SPARKFUN. **Raspberry Pi GPIO**. Disponível em: <<https://learn.sparkfun.com/tutorials/raspberry-gpio/gpio-pinout>>. Acesso em: 21 de março de 2018.

UPTON, E. *et al.* **Learning Computer Architecture with Raspberry Pi**. Indianápolis: Wiley, 2016.



## APÊNDICE A

Neste Apêndice, serão descritas as operações preliminares que devem ser executadas para a utilização do Raspberry Pi com os softwares MATLAB e Simulink. Os procedimentos aqui apresentados são adaptações dos procedimentos descritos por Pasolini, Bazzi e Mirabella (2016). Esses autores desenvolveram o trabalho para o Raspberry Pi 2 e para a versão do MATLAB R2015a. Todavia, para a execução dos procedimentos apresentados foram utilizados o Raspberry Pi 3 e a versão do MATLAB R2016a.

Os procedimentos apresentados neste Apêndice permitem a comunicação entre a placa Raspberry Pi 3 e um computador com os softwares MATLAB e Simulink. Com isso, torna-se possível programar o Raspberry através desses softwares, sem a necessidade de conectar a placa a um monitor, um mouse e um teclado.

Para a execução desses procedimentos, serão necessários os seguintes itens:

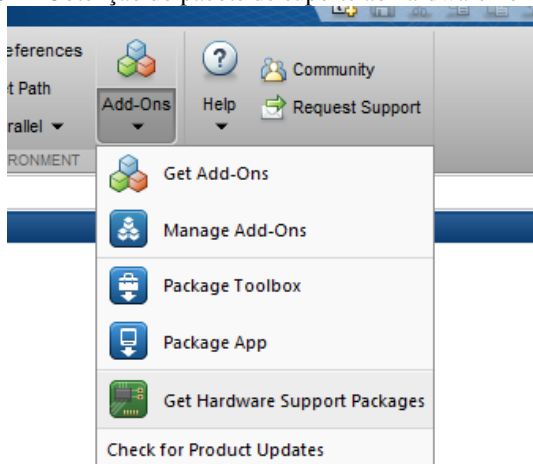
- Uma placa Raspberry Pi 3;
- Um cabo micro USB para alimentação do Raspberry;
- Um cartão micro SD e seu adaptador SD;
- Um cabo Ethernet;
- Um adaptador USB para LAN.

### 5.1 INSTALAÇÃO DO PACOTE DE SUPORTE AO HARDWARE

De acordo com Pasolini, Bazzi e Mirabella (2016), a popularização de placas de prototipação, microcontroladores e computadores *single-board* fizeram com que a Mathworks desenvolvesse pacotes de suporte ao hardware (*Hardware Support Packages*) específicos para esses dispositivos. Esses pacotes de suporte ao hardware permitem a interface entre os softwares MATLAB e Simulink e dispositivos desenvolvidos por terceiros, como placas de prototipagem Arduino, receptores RTL-SDR e placas Raspberry Pi.

O primeiro passo a ser realizado é a inicialização do MATLAB. Em seguida, o pacote de suporte ao hardware do Raspberry Pi pode ser instalado a partir do próprio ambiente do MATLAB em **Add-Ons** → **Get Hardware Support Packages**, como mostra a Figura 162.

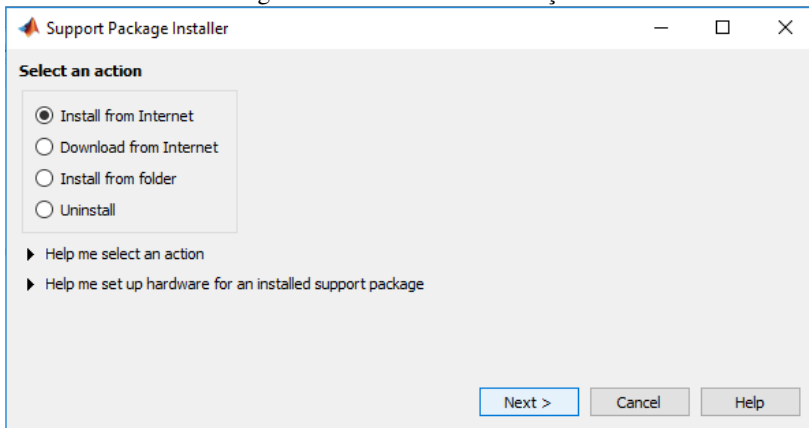
Figura 162 – Obtenção do pacote de suporte ao hardware no MATLAB.



Fonte: O autor (2018).

Em seguida, a janela mostrada na Figura 163 aparecerá. Deve-se então selecionar a ação **Install from Internet** e continuar, clicando em **Next**.

Figura 163 – Selecionar uma ação.

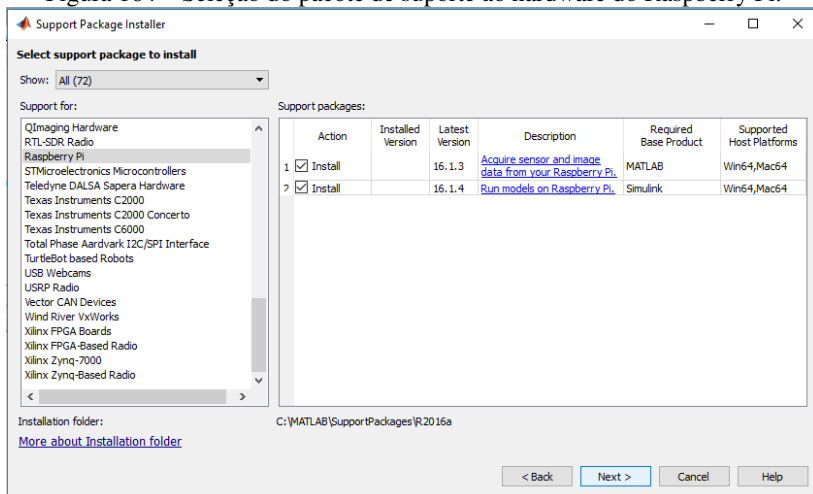


Fonte: O autor (2018).

Após clicar em Next na janela mostrada na Figura 163, a janela mostrada na Figura 164 aparecerá. Deve-se selecionar então os dois

pacotes de suporte ao hardware do Raspberry Pi (para o MATLAB e para o Simulink) e então clicar em **Next**.

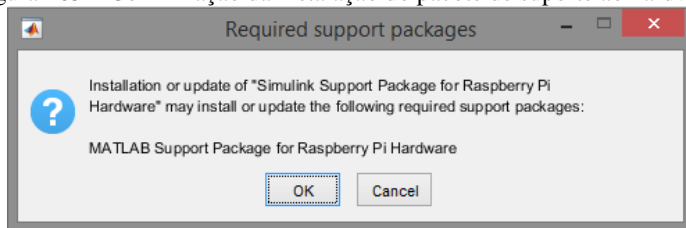
Figura 164 – Seleção do pacote de suporte ao hardware do Raspberry Pi.



Fonte: O autor (2018).

Ao clicar em **Next** na janela mostrada na Figura 164, a janela mostrada na Figura 165 aparecerá. Nessa janela, deve-se clicar em **OK**.

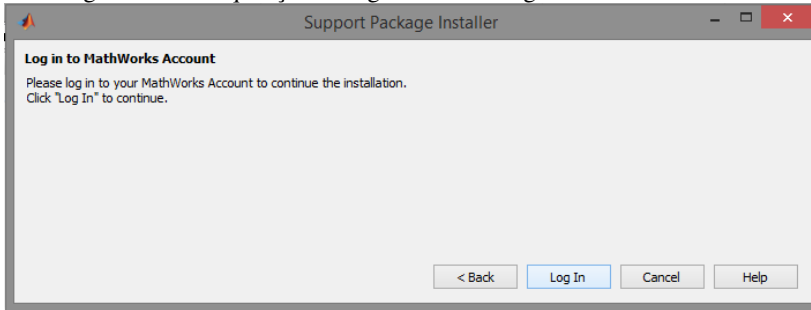
Figura 165 – Confirmação da instalação do pacote de suporte ao hardware.



Fonte: O autor (2018).

Na sequência, a janela mostrada na Figura 166 aparecerá. Para continuar o procedimento de instalação do pacote de suporte ao hardware do Raspberry, será necessário que o usuário tenha uma conta registrada da Mathworks. Se o usuário já a possui, deve-se então clicar em **Log In**.

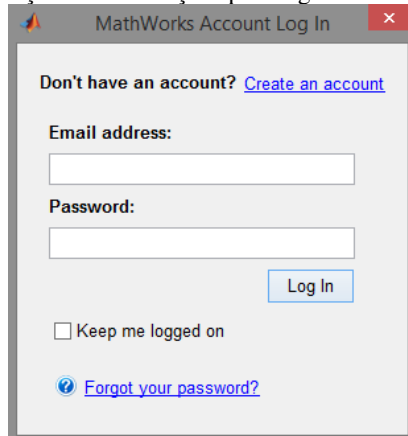
Figura 166 – Requisição de log in na conta registrada da Mathworks.



Fonte: O autor (2018).

As informações para log in na conta da Mathworks (e-mail e senha) devem ser inseridas nos campos correspondentes na janela mostrada na Figura 167. Após isso, deve-se clicar em **Log In**.

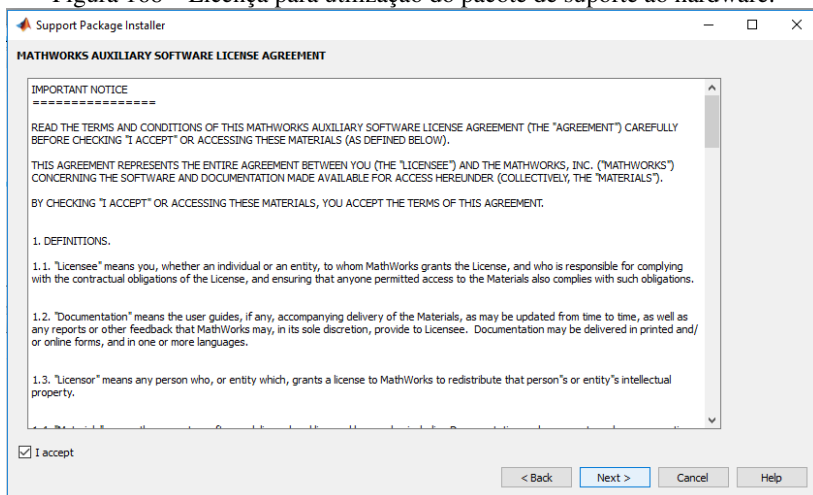
Figura 167 – Inserção de informações para log in na conta da Mathworks.



Fonte: O autor (2018).

Após a realização do log in, a janela mostrada na Figura 168 aparecerá. Nessa janela, será exibida os termos da licença para utilização do pacote de suporte ao hardware do Raspberry Pi. Deve-se então aceitar as condições da licença, e clicar em **Next**.

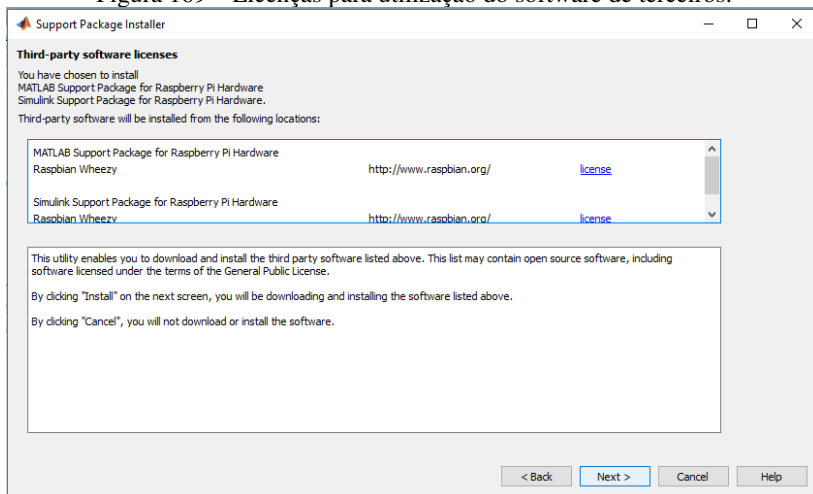
Figura 168 – Licença para utilização do pacote de suporte ao hardware.



Fonte: O autor (2018).

Após o usuário aceitar os termos da licença, a janela mostrada na Figura 169 aparecerá. Nessa nova janela, deve-se clicar em **Next**.

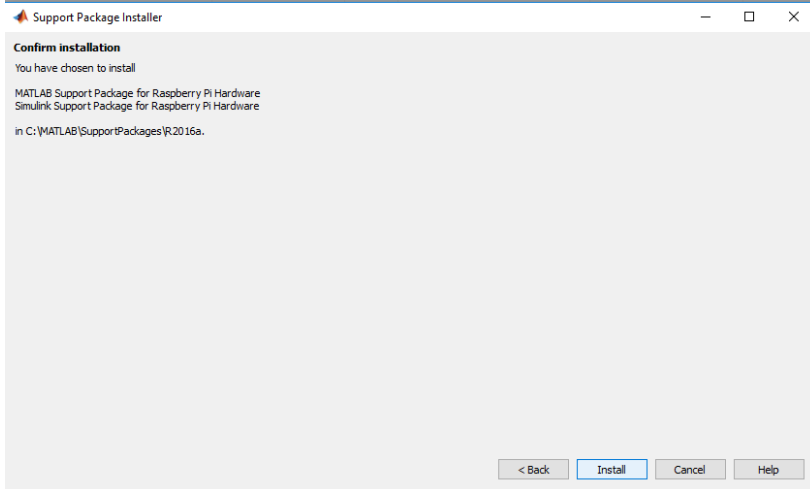
Figura 169 – Licenças para utilização do software de terceiros.



Fonte: O autor (2018).

Após clicar em Next na janela mostrada na Figura 169, surgirá a janela mostrada na Figura 170. Finalmente, ao clicar em **Install** nessa nova janela, a instalação do pacote de suporte ao hardware do Raspberry Pi será iniciada.

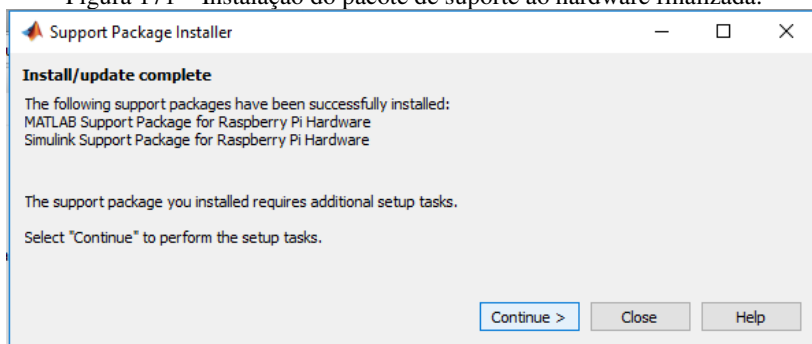
Figura 170 – Confirmação da instalação do pacote de suporte ao hardware.



Fonte: O autor (2018).

Após clicar-se no botão Install da janela mostrada na Figura 170, tem-se início a instalação do **MATLAB Support Package for Raspberry Pi** e do **Simulink Support Package for Raspberry Pi**. Ao final da instalação, a janela mostrada na Figura 171 será mostrada. Essa janela indica que a instalação dos dois pacotes foi realizada com sucesso, e portanto os softwares MATLAB e Simulink já estão prontos para interagir com placas Raspberry Pi.

Figura 171 – Instalação do pacote de suporte ao hardware finalizada.



Fonte: O autor (2018).

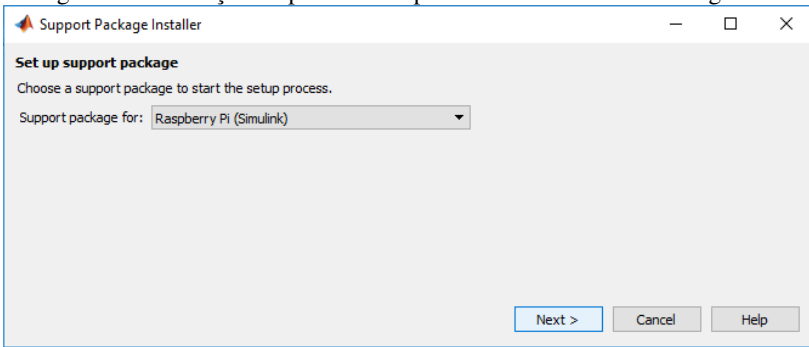
## 5.2 INSTALAÇÃO DO RASPBIAN E CONFIGURAÇÃO DA REDE

Pasolini, Bazzi e Mirabella (2016) apontam que a comunicação entre o MATLAB/Simulink e o Raspberry é realizada através de uma conexão de rede. A configuração dessa rede pode ser realizada durante os procedimentos de instalação do pacote de suporte ao hardware, ou depois desses procedimentos, acessando-se diretamente os arquivos de configuração de rede do Raspberry Pi.

Conforme apontam os autores, a primeira opção é a mais rápida. Uma vez que o sistema operacional do Raspberry Pi já instalado em seu cartão de memória, é possível controlar a placa através de um computador com os softwares MATLAB e Simulink, sem a necessidade de conectar um monitor, um mouse e um teclado.

Para iniciar os procedimentos de instalação do sistema operacional do Raspberry Pi num cartão de memória, deve-se clicar em **Continue** na janela mostrada na Figura 171. Após isso, a janela mostrada na Figura 172 aparecerá. Nessa janela, deve-se selecionar o pacote de suporte ao hardware a ser configurado, e em seguida clicar-se em **Next**.

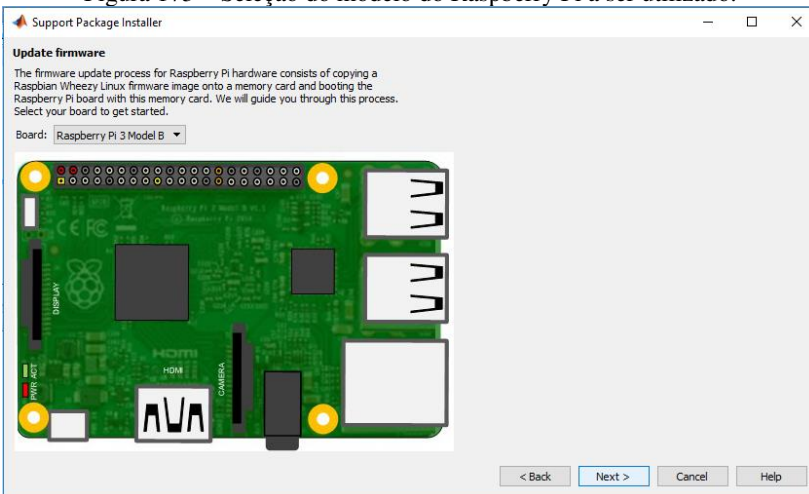
Figura 172 – Seleção do pacote de suporte ao hardware a ser configurado.



Fonte: O autor (2018).

Na sequência, a janela mostrada na Figura 173 aparecerá. Nessa janela, devemos selecionar o modelo do Raspberry Pi a ser utilizado, que no caso deste trabalho é o Raspberry Pi 3 Model B, e em seguida clicar em **Next**. Nas próximas etapas, serão executados os passos para a instalação do sistema operacional Raspbian num cartão micro SD. O Raspbian é uma distribuição Linux desenvolvida exclusivamente para o Raspberry Pi.

Figura 173 – Seleção do modelo do Raspberry Pi a ser utilizado.

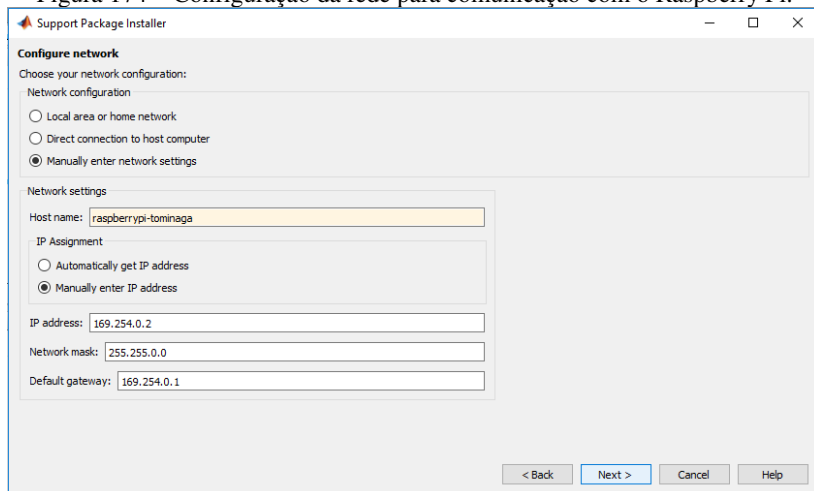


Fonte: O autor (2018).



A janela que aparecerá na sequência está mostrada na Figura 174. Nessa janela serão definidas as configuração da conexão de rede entre o computador e o Raspberry Pi. Os procedimentos descritos na sequência irão diretamente modificar os arquivos de sistema do Raspberry Pi contendo os parâmetros da rede.

Figura 174 – Configuração da rede para comunicação com o Raspberry Pi.



Fonte: O autor (2018).

Conforme mostra a Figura 174, devemos selecionar a opção **Manually enter network settings**. Em seguida, devemos definir o **Host name** que identificará o Raspberry Pi a ser configurado. Para o Raspberry Pi utilizado neste trabalho, o host name foi definido como “raspberrypi-tominaga”, porém essa escolha é absolutamente arbitrária.

Na janela mostrada na Figura 174, deve-se também selecionar a opção **Manually enter IP address**. De acordo com Pasolini, Bazzi e Mirabella (2016), a escolha do endereço de IP para o Raspberry Pi dependerá da disponibilidade de endereços na rede local (LAN, sigla para *local-area network*). Os autores apontam que a disponibilidade de um IP pode ser verificada no *prompt* de comando do Windows com o uso do comando *PING #IPAddress*. Os parâmetros da rede inseridos na janela mostrada na Figura 174 se encontram no Quadro 4.

Quadro 4 – Parâmetros utilizados para a configuração da rede.

|                 |             |
|-----------------|-------------|
| IP address      | 169.254.0.2 |
| Network mask    | 255.255.0.0 |
| Default gateway | 169.254.0.1 |

Fonte: O autor (2018).

Pasolini, Bazzi e Mirabella (2016) apresentam também os procedimentos para configuração de uma rede com múltiplas placas Raspberry Pi sendo controladas simultaneamente. Todavia, neste trabalho somente uma única placa Raspberry Pi será utilizada.

Após clicar em **Next** na janela mostrada na Figura 174, a janela mostrada na Figura 175 aparecerá. Essa janela informa ao usuário que é preciso inserir um cartão micro SD (com um adaptador apropriado, se necessário) no *slot* apropriado do computador. Além disso, o cartão SD deve possuir no mínimo 4 GB de memória. Após inserir o cartão, deve-se clicar em **Refresh** para certificar-se de que ele foi reconhecido pelo computador. Na sequência, deve-se clicar em **Next**.

Figura 175 – Inserção de um cartão micro SD no computador.



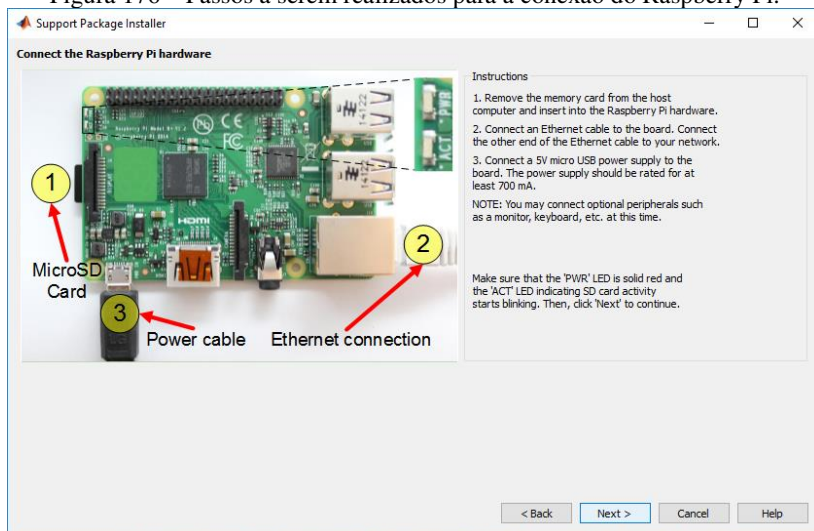
Fonte: O autor (2018).

Uma nova janela aparecerá, informando ao usuário que a operação para gravação do Raspian no cartão SD levará muitos minutos,

e que esse procedimento apagará todos os arquivos presentes no cartão. Nessa janela, o usuário deve clicar em **Write**, e em seguida aguardar até que a gravação esteja completada.

Após o Raspbian ter sido gravado com sucesso no cartão SD, a janela mostrada na Figura 176 aparecerá. Essa janela informa os passos necessários para realizar a conexão do Raspberry Pi com o computador.

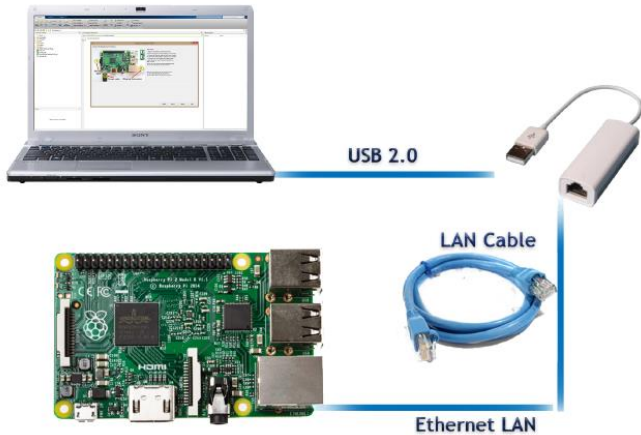
Figura 176 – Passos a serem realizados para a conexão do Raspberry Pi.



Fonte: O autor (2018).

Primeiramente, deve-se remover o cartão micro SD do computador e inseri-lo no Raspberry. Em seguida, uma extremidade do cabo Ethernet deve ser conectada ao Raspberry Pi, e a outra extremidade deve ser conectada ao computador (nessa etapa, o adaptador USB para LAN é utilizado para permitir que a extremidade do cabo Ethernet seja conectada a uma entrada USB do computador). Esse esquema de conexão pode ser observado na Figura 177.

Figura 177 – Esquema utilizado para estabelecer a conexão entre o Raspberry Pi e o computador.



Fonte: Pasolini, Bazzi e Mirabella (2016).

Por fim, o Raspberry Pi deve ser alimentado. A alimentação pode ser proveniente de uma fonte DC externa de 5 V (com capacidade para pelo menos 700 mA) ou proveniente de um cabo micro USB conectado a outra entrada USB do computador. Após a realização de todos esses passos, deve-se clicar em **Next** na janela mostrada na Figura 176.

Antes de realizar os próximos passos no MATLAB para a configuração do Raspberry Pi, é necessário a realização de alguns procedimentos para a configuração do adaptador USB para Ethernet.

### 5.3 CONFIGURAÇÃO DO ADAPTADOR USB PARA ETHERNET

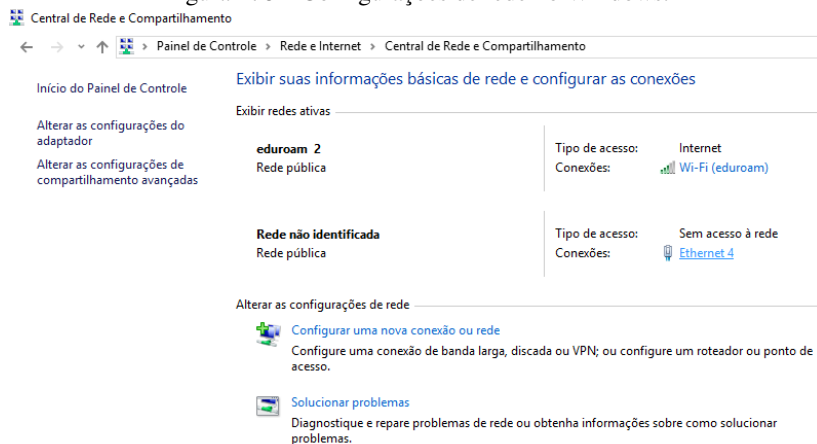
De acordo com Pasolini, Bazzi e Mirabella (2016), a comunicação entre o computador e o Raspberry Pi é feita através de uma conexão Ethernet. Isso implica que uma conexão direta entre eles ocupa a porta Ethernet do computador, que muitas vezes é utilizada para conectá-lo à internet. A utilização do adaptador USB para Ethernet evita esse problema.

A primeira etapa da configuração do adaptador USB para Ethernet consiste em conectá-lo a uma das entradas USB do computador. Para alguns sistemas operacionais, é necessário instalar o driver do adaptador. Porém, no caso do Windows 10, por exemplo, essa etapa não é necessária.

É importante salientar que os parâmetros da rede a serem configurados nessa etapa (IP do adaptador, máscara da rede e gateway) devem ser consistentes com os utilizados anteriormente nos procedimentos realizados no MATLAB (ver Quadro 4).

Em seguida, as configurações de rede do Windows devem ser acessadas em **Painel de Controle** → **Rede e Internet** → **Central de Rede e Compartilhamento**, conforme pode ser observado na Figura 178.

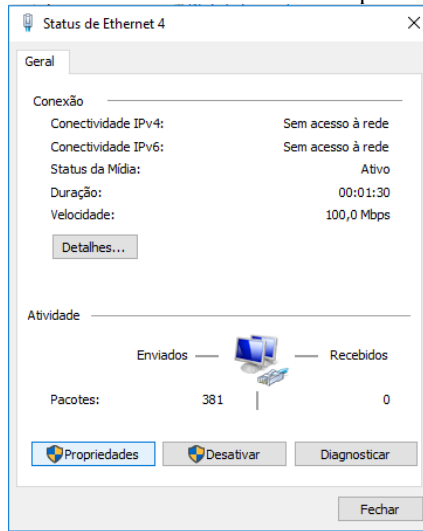
Figura 178 – Configurações de rede no Windows.



Fonte: O autor (2018).

Ao conectarmos o Raspberry Pi com o computador através do adaptador, uma rede local é criada. Essa rede é identificada como **Ethernet 4** na Figura 178. Ao clicarmos nesse nome, a janela mostrada na Figura 179 é exibida.

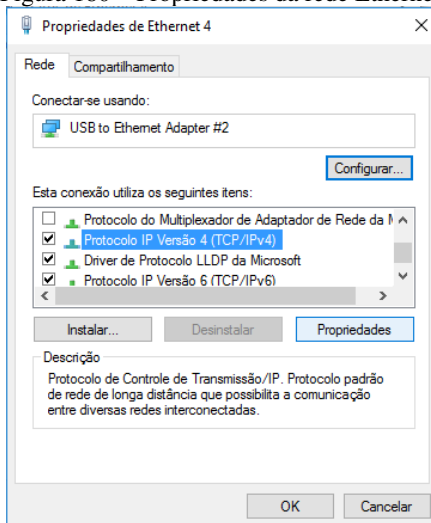
Figura 179 – Rede Ethernet estabelecida entre o computador e o Raspberry Pi.



Fonte: O autor (2018).

Na janela mostrada na Figura 179, deve-se clicar em **Propriedades**. Após isso, a janela mostrada na Figura 180 é exibida.

Figura 180 – Propriedades da rede Ethernet.



Fonte: O autor (2018).

Na janela mostrada na Figura 180, deve-se selecionar **Protocolo IP Versão 4 (TCP/IPv4)**, e em seguida deve-se clicar em **Propriedades**. Como resultado dessa ação, a janela mostrada na Figura 181 será exibida.

Figura 181 – Propriedades do protocolo de IP Versão 4 (TCP/IPv4).

Propriedades de Protocolo IP Versão 4 (TCP/IPv4)

Gerar

As configurações IP podem ser atribuídas automaticamente se a rede oferecer suporte a esse recurso. Caso contrário, você precisa solicitar ao administrador de rede as configurações IP adequadas.

Obter um endereço IP automaticamente

Usar o seguinte endereço IP:

Endereço IP: 169 . 254 . 0 . 2

Máscara de sub-rede: 255 . 255 . 0 . 0

Gateway padrão: 169 . 254 . 0 . 1

Obter o endereço dos servidores DNS automaticamente

Usar os seguintes endereços de servidor DNS:

Servidor DNS preferencial: . . .

Servidor DNS alternativo: . . .

Validar configurações na saída

Avançado...

OK Cancelar

Fonte: O autor (2018).

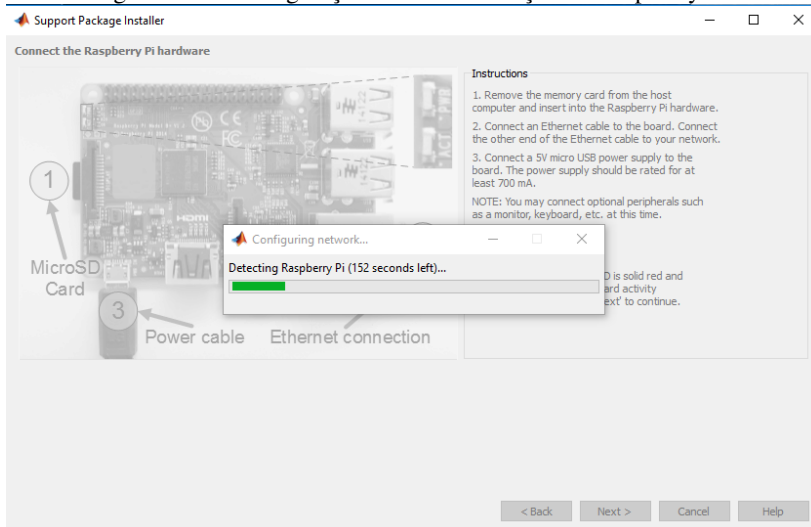
Na janela mostrada na Figura 181, deve-se selecionar a opção **Usar o seguinte endereço IP**. Nos campos correspondentes a **Endereço IP**, **Máscara de sub-rede** e **Gateway Padrão**, o usuário deve inserir os parâmetros que estão listados no Quadro 4. Após isso, deve-se clicar em **Ok** na janela mostrada na Figura 181, **Ok** na janela mostrada na Figura 180 e **Fechar** na janela mostrada na Figura 179.

Após a realização desses procedimentos, a rede Ethernet entre o computador e o Raspberry Pi está devidamente configurada. Com isso, o usuário pode retornar à tela de configuração no MATLAB mostrada na Figura 176.

## 5.4 INICIALIZAÇÃO DO RASPBERRY PI

Uma vez cumpridos todos os procedimentos descritos anteriormente, deve-se clicar no botão **Next** da janela mostrada na Figura 176. Após isso, o MATLAB iniciará um procedimento para configuração da rede e detecção do Raspberry Pi conectado ao computador. Durante esse procedimento, que levará alguns minutos, a janela mostrada na Figura 182 será exibida.

Figura 182 – Configuração da rede e detecção do Raspberry Pi.

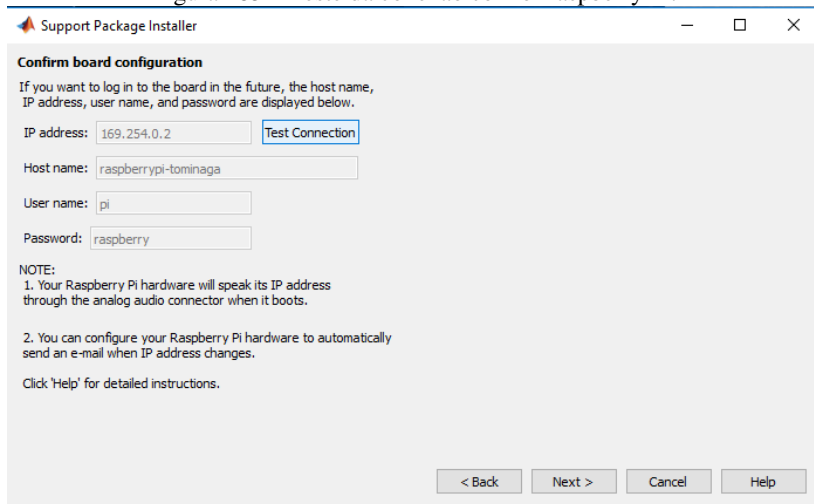


Fonte: O autor (2018).

Ao final do procedimento de configuração da rede e detecção do Raspberry Pi, a janela mostrada na Figura 183 será mostrada. Nessa janela, podem ser observados o endereço de IP e o *host name* do Raspberry Pi, definidos previamente. Além disso, também podem ser observados o nome de usuário e a senha, parâmetros cujo conhecimento é necessário para a utilização do Raspberry Pi. Por padrão, **User name** e **password** são definidos como “pi” e “raspberry”, respectivamente.



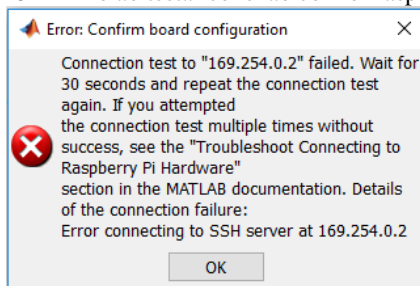
Figura 183 – Teste da conexão com o Raspberry Pi.



Fonte: O autor (2018).

Na janela mostrada na Figura 183, deve-se clicar em **Test Connection** para testar a comunicação do Raspberry Pi com o computador. É provável que, na primeira vez na qual a conexão for testada, ocorra um erro e mensagem mostrada na Figura 184 seja mostrada.

Figura 184 – Erro ao testar conexão com o Raspberry Pi.



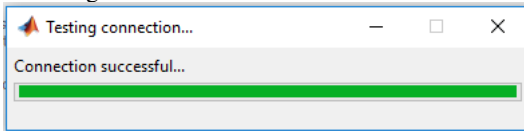
Fonte: O autor (2018).

Para resolver esse erro, basta reiniciar o Raspberry Pi. Isso pode ser feito desconectando a placa da fonte de alimentação, e em seguida conectando-a novamente. Deve-se então esperar até que a conexão Ethernet entre o Raspberry Pi e o computador seja reestabelecida. O

status da conexão pode ser acompanhado em Painel de Controle → Rede e Internet → Central de Rede e Compartilhamento.

Após a conexão ter sido restabelecida, deve-se clicar novamente no botão **Test connection** mostrado na Figura 183. Se a comunicação for bem sucedida, então a mensagem mostrada na Figura 185 poderá ser observada.

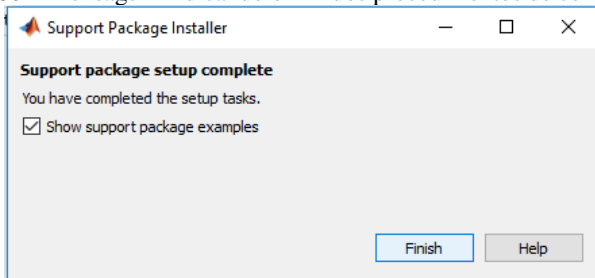
Figura 185 – Mensagem indicando sucesso na conexão com o Raspberry Pi.



Fonte: O autor (2018).

Após a conexão ter sido testada com sucesso, a janela mostrada na Figura 186 será exibida. Essa janela indica o fim dos procedimentos de configuração do pacote de suporte ao hardware, do Raspberry Pi e da conexão entre o Raspberry Pi e o computador. Após clicar-se no botão **Finish**, o usuário pode começar a controlar o Raspberry através dos softwares MATLAB e Simulink.

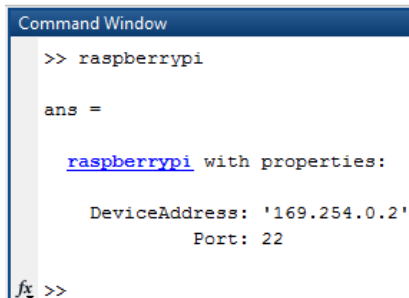
Figura 186 – Mensagem indicando o fim dos procedimentos de configuração.



Fonte: O autor (2018).

Toda vez que o MATLAB for iniciado e o Raspberry Pi conectado ao computador, a conexão entre a placa e o software pode ser testada na **Command Window** do MATLAB com o uso do comando “`raspberrypi`”. Se a conexão for bem sucedida, o comando retorna as informações mostradas na Figura 187.

Figura 187 – Teste da conexão entre o Raspberry Pi e o MATLAB na Command Window.



```
Command Window
>> raspberrypi

ans =

    raspberrypi with properties:
    DeviceAddress: '169.254.0.2'
    Port: 22

fx >>
```

Fonte: O autor (2018).

É possível invocar o Shell do Linux (sistema operacional do Raspberry Pi) através da Command Window. Utilizando comandos no Shell, é possível acessar as configurações do Raspberry Pi, bem como diretórios e arquivos contidos em sua memória. Os dois comandos que devem ser executados sequencialmente na Command Window para invocar o Shell podem ser observados na Figura 188.

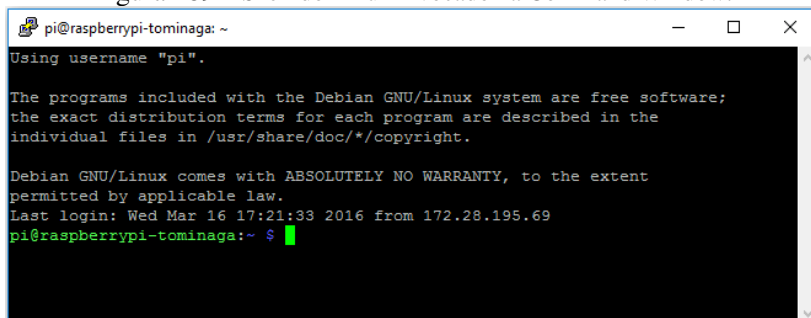
Figura 188 – Comandos utilizados para invocar o Shell do Linux na Command Window.

```
>> h=raspberrypi('169.254.0.2');
>> h.openShell
```

Fonte: O autor (2018).

Após a execução dos comandos mostrados na Figura 188, o Shell do Linux aparecerá, conforme pode ser observado na Figura 189.

Figura 189 – Shell do Linux invocado na Command Window.



```
pi@raspberrypi-tominaga: ~
Using username "pi".

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 16 17:21:33 2016 from 172.28.195.69
pi@raspberrypi-tominaga:~ $
```

Fonte: O autor (2018).

Na próxima seção, alguns comandos serão utilizados no Shell do Linux para a configuração da placa de som USB.

## 5.5 CONFIGURAÇÃO DA PLACA DE SOM USB EXTERNA

Pasolini, Bazzi e Mirabella (2016) apresentam os procedimentos para configuração da placa de som USB externa. Porém, os procedimentos apresentados por esses autores se aplicam somente ao Raspberry Pi 2. Os procedimentos para a configuração da placa de som no Raspberry Pi 3, que serão descritos nesta subseção, foram propostos por Bourdon (2018).

De acordo com Pasolini, Bazzi e Mirabella (2016), para a placa de som USB externa ser utilizada, primeiramente é necessário modificar alguns dos parâmetros dos drivers de som do Raspberry Pi. Mais especificamente, o primeiro passo é definir a placa de som como o dispositivo de áudio primário.

Após a placa de som ter sido conectada a uma das entradas USB do Raspberry Pi, é possível checar sua presença como dispositivo de saída do Raspberry com o uso do comando “`aplay -l`” no Shell do Linux, conforme pode ser observado na Figura 190.

Figura 190 – Utilização do comando “`aplay -l`” no Shell do Linux para checar a presença da placa de som USB externa como dispositivo de saída.

```

pi@raspberrypi-tominaga: ~
individual files in /usr/share/doc/*/copyright.

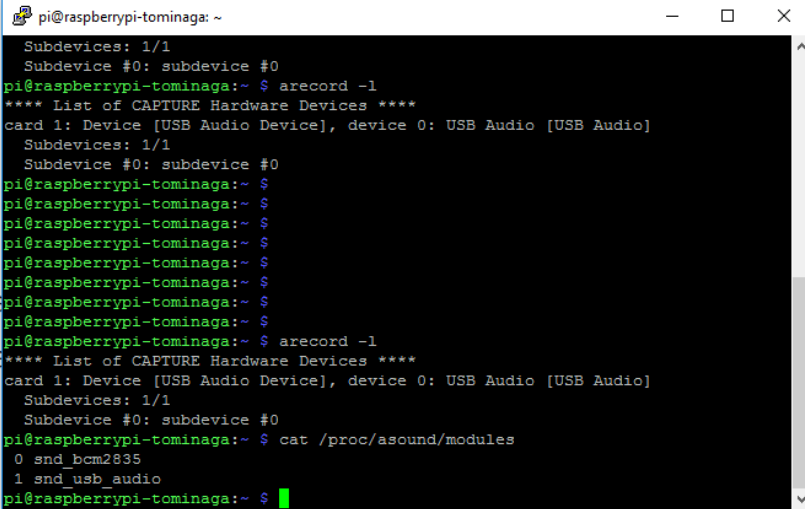
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 16 17:32:26 2016 from 169.254.68.31
pi@raspberrypi-tominaga:~$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 8/8
    Subdevice #0: subdevice #0
    Subdevice #1: subdevice #1
    Subdevice #2: subdevice #2
    Subdevice #3: subdevice #3
    Subdevice #4: subdevice #4
    Subdevice #5: subdevice #5
    Subdevice #6: subdevice #6
    Subdevice #7: subdevice #7
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
  Subdevices: 1/1
    Subdevice #0: subdevice #0
card 1: Device [USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
    Subdevice #0: subdevice #0
pi@raspberrypi-tominaga:~$

```

Fonte: O autor (2018).



Figura 192 – Utilização do comando “cat /proc/asound/modules” para verificar a ordem de prioridade dos dispositivos de áudio conectados ao Raspberry Pi.



```

pi@raspberrypi-tominaga: ~
Subdevices: 1/1
Subdevice #0: subdevice #0
pi@raspberrypi-tominaga:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: Device [USB Audio Device], device 0: USB Audio [USB Audio]
Subdevices: 1/1
Subdevice #0: subdevice #0
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: Device [USB Audio Device], device 0: USB Audio [USB Audio]
Subdevices: 1/1
Subdevice #0: subdevice #0
pi@raspberrypi-tominaga:~ $ cat /proc/asound/modules
0 snd_bcm2835
1 snd_usb_audio
pi@raspberrypi-tominaga:~ $

```

Fonte: O autor (2018).

Para definir a placa de som externa como o dispositivo de áudio primário, é preciso alterar seu parâmetro de 1 para 0, e também alterar o parâmetro do DAC do Raspberry Pi de 0 para 1. Para isso, primeiramente é preciso criar um arquivo de configuração denominado “alsa-base.conf” no diretório “/etc/modprobe.d”. Esse arquivo deve ser criado com o uso de editor de texto *nano* do Linux, que deve ser invocado com o uso do comando de *super user* “*sudo*”. O comando completo a ser usado no Shell do Linux pode ser observado na Figura 193.

Figura 193 – Utilização do comando “sudo nano /etc/modprobe.d/alsa-base.conf” no Shell do Linux para a criação de um arquivo de configuração.

```

pi@raspberrypi-tominaga: ~
**** List of CAPTURE Hardware Devices ****
card 1: Device [USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
    Subdevice #0: subdevice #0
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: Device [USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
    Subdevice #0: subdevice #0
pi@raspberrypi-tominaga:~ $ cat /proc/asound/modules
0 snd_bcm2835
1 snd_usb_audio
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $
pi@raspberrypi-tominaga:~ $ sudo nano /etc/modprobe.d/alsa-base.conf
pi@raspberrypi-tominaga:~ $ sudo nano /etc/modprobe.d/alsa-base.conf

```

Fonte: O autor (2018).

Após a utilização do comando “sudo nano /etc/modprobe.d/alsa-base-config”, o editor de texto nano será aberto. Em seguida, o trecho de código mostrado na Figura 194 deverá ser inserido.

Figura 194 – Trecho de código a ser inserido no arquivo de configuração.

```

GNU nano 2.2.6 File: /etc/modprobe.d/alsa-base.conf Modified
# This sets the index value of the cards but doesn't reorder:
options snd_usb_audio index=0
options snd_bcm2835 index=1

# Does the reordering:
options snd_slots=snd_usb_audio,snd_bcm2835

```

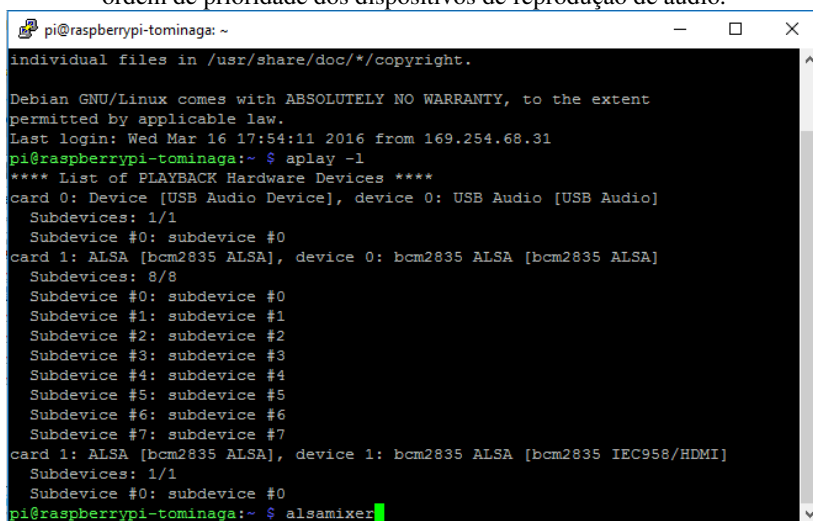
Fonte: O autor (2018).

A primeira parte do código mostrado na Figura 194 define o parâmetro da placa de som externa como 0, e o parâmetro do DAC do





Figura 196 – Utilização do comando “aplay -l” no Shell do Linux para checar a ordem de prioridade dos dispositivos de reprodução de áudio.



```

pi@raspberrypi-tominaga: ~
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 16 17:54:11 2016 from 169.254.68.31
pi@raspberrypi-tominaga:~ $ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: Device [USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 1: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
pi@raspberrypi-tominaga:~ $ alsamixer

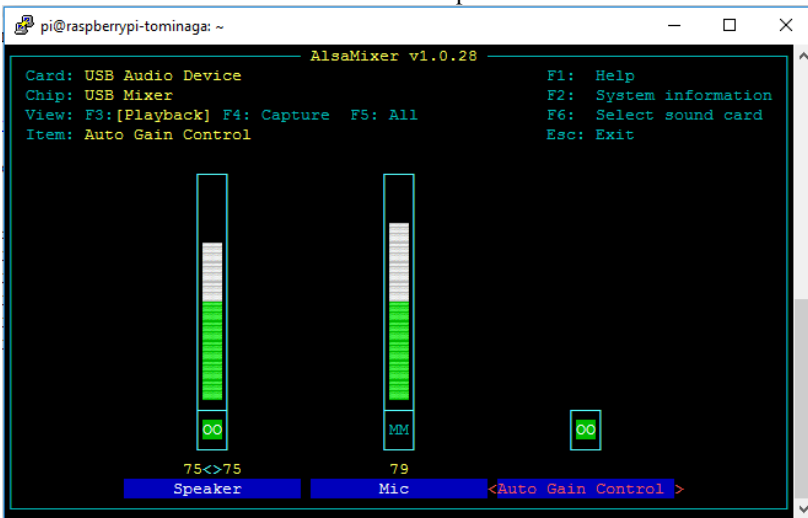
```

Fonte: O autor (2018).

Conforme pode ser observado na Figura 196, os procedimentos realizados para definir a placa de som USB externa como o dispositivo de áudio primário foram realizados com sucesso: o dispositivo **USB Audio Card** possui o parâmetro 0 (**card 0**), enquanto que o dispositivo **bcm2855** possui o parâmetro 1 (**card 1**).

Para utilizarmos adequadamente a entrada e a saída presentes na placa de som USB externa, é preciso definir adequadamente os níveis de som (volumes) correspondentes. Para isso, o comando “alsamixer” deve ser executado nesse comando. Ao ser executada, a janela que pode ser observada na Figura 197 aparecerá no Shell.

Figura 197 – Utilização do Alsamixer, invocado no Shell do Linux, para definir os volumes da entrada e da saída da placa de som USB externa.



Fonte: O autor (2018).

Por definição, o volume da entrada de microfone da placa de som é definido inicialmente como 0. Com a utilização das setas do teclado, os volumes da entrada e da saída podem ser ajustados. Com o uso tecla F6 é possível acessar a configuração de outras placas de som que podem estar conectadas ao Raspberry Pi.

