UNIVERSIDADE FEDERAL DO PARANÁ - DEPARTAMENTO DE ENGENHARIA ELÉTRICA

GABRIEL DOS SANTOS HAVEROTH
ROBSON YOSHIHIRO FUJIOKA

SISTEMA DE *BIOFEEDBACK* ATRAVÉS DO MONITORAMENTO DA ATIVIDADE RESPIRATÓRIA

CURITIBA
2018

GABRIEL DOS SANTOS HAVEROTH
ROBSON YOSHIHIRO FUJIOKA

SISTEMA DE *BIOFEEDBACK* ATRAVÉS DO MONITORAMENTO DA ATIVIDADE RESPIRATÓRIA

Trabalho apresentado como requisito parcial à obtenção de nota da disciplina de Trabalho de Conclusão de curso B do Curso de Engenharia Elétrica com Ênfase em Sistemas Eletrônicos Embarcados.
Orientador: Dr. Marcos Vinicio Haas Rambo

CURITIBA
2018

TERMO DE APROVAÇÃO

GABRIEL DOS SANTOS HAVEROTH
ROBSON YOSHIHIRO FUJIOKA

SISTEMA DE *BIOFEEDBACK* ATRAVÉS DO MONITORAMENTO DA ATIVIDADE RESPIRATÓRIA

Trabalho apresentado como requisito parcial à obtenção de nota da disciplina de Trabalho de Conclusão de curso B do Curso de Engenharia Elétrica com Ênfase em Sistemas Eletrônicos Embarcados.
Orientador: Dr. Marcos Vinicio Haas Rambo

BANCA EXAMINADORA:

Orientador

Dr. Henri Frederico Eberspacher
Departamento de Engenharia Elétrica,
UFPR

Dr. João da Silva Dias
Departamento de Engenharia Elétrica,
UFPR

Curitiba, 04 de Dezembro de 2018.

Dedico este trabalho ao meu pai, Yoshiyuki Fujioka, que mesmo não estando mais presente fisicamente, estará sempre comigo de algum lugar especial.

AGRADECIMENTOS

Agradecemos às nossas famílias, pelo suporte e pelo incentivo que nos proporcionou a chegar ao término deste trabalho.

Aos nossos amigos e colegas, que tornaram a jornada dentro da universidade um caminho menos árduo e muito mais leve.

Às nossas queridas namoradas, que nos apoiaram e entenderam a necessidade de ausência em muitos momentos desta jornada.

À Universidade Federal do Paraná e todos os seus docentes, que nos proporcionaram uma educação gratuita de qualidade e também nos ensinaram o quão é importante a busca constante e autônoma pelo conhecimento.

Em especial, agradecemos ao Prof. Dr. Marcos Vinicio Haas Rambo, que nos orientou e incentivou durante a realização deste trabalho de conclusão de curso.

RESUMO

Neste trabalho, através do monitoramento da atividade respiratória, pretende-se desenvolver um sistema de *biofeedback*. Este tipo de sistema faz a aquisição de informações fisiológicas e retorna ao indivíduo. Com essas informações é possível realizar um treinamento para se obter um autocontrole das funções fisiológicas, tais como o controle de respiração, batimento cardíaco, entre outras. À vista disso, o *biofeedback* pode auxiliar no tratamento de pessoas com transtornos, crises de ansiedade, assim como no gerenciamento do estresse. Neste trabalho foi desenvolvido um sistema de monitoramento da atividade respiratória através de um transdutor de deformação posicionado no abdômen do indivíduo. O trabalho contemplou a elaboração de um circuito de condicionamento de sinal para tratar os sinais provenientes deste transdutor e que são enviados através da comunicação sem-fio *Wi-Fi* a um dispositivo externo. No dispositivo externo, aplicou-se técnicas de processamento digital de sinais para processar e quantificar a atividade respiratória, retornando as informações ao usuário através de uma interface gráfica.

Palavras-chaves: *Biofeedback*, respiração, monitoramento, *Wi-Fi*.

ABSTRACT

In this work, through the monitoring of respiratory activity, we intend to develop a biofeedback system. This type of system makes the acquisition of information and returns to the individual. Thus, it is possible to perform a training to obtain a control of the physiological functions, such as breathing control, heart-beat, among others. Therefore, biofeedback can assist in the treatment of people with disorders, anxiety crisis, as well as in stress management. In this context, in the development of this work, we intend to monitor the respiratory activity through extensometers positioned in the individual's abdominal and thoracic. In this way, the signal conditioning circuit for the signals from this transducer will be elaborated. Then, the digital processing of these signals will be performed where the information will be sent to a device through the WI-FI wireless communication. In this way, the device that receives the data will make them available to the user through an application program.

Key-words: Biofeedback, Wi-Fi, respiratory, monitoring.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	*Analog Digital Converter*
C	Capacitor
CMRR	Common-Mode Rejection Ratio
dB	Decibel
FR	Frequência Respiratória
GPIO	*General Purpose Input/Output*
Hz	Hertz
K	Fator do Extensômetro
L	Comprimento Inicial
MCU	*MicroController*
nF	nano Faraday
Ni	Níquel
R	Resistência
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UDP	*User Datagram Protocol*
V	Tensão
VC	Volume Corrente
VR	Volume Residual
VRE	Volume de Reserva expiratório
VRI	Volume de Reserva inspiratório
W	Watts

Wi-Fi	Wireless Fidelity
$\triangle$L	Variação do comprimento
$\triangle$R	Variação da resistência devido a deformação
Ω	Ohm

SUMÁRIO

1 INTRODUÇÃO

O termo *biofeedback* começou a ser utilizado na década de 60, sendo uma forma de terapia para aprendizado e autocorreção. Esta técnica envolve o monitoramento através de dispositivos sensoriais que realizam a aquisição de informações fisiológicas do indivíduo e retorna-as a ele. Desta forma, utilizando-se aparelhos eletrônicos, é possível promover uma interface para que diversas grandezas fisiológicas possam ser monitoradas e que retornem estas informações necessárias ao paciente. Com acesso a estas informações e o tratamento adequado, o paciente poderá melhorar suas reações fisiológicas e emocionais (ZHU; KONG; XIE, 2012). Outro termo que vem sendo amplamente utilizado é o *wearable*, que refere-se a tecnologias vestíveis. Estes dispositivos eletrônicos vestíveis tem sido utilizado algumas áreas, especialmente em esportes, cuidados com a saúde e entretenimento (JEYHANI et al., 2017). Agregando esta tecnologia com a transmissão de dados sem-fio e fazendo-se uma análise em tempo real, obtêm-se uma ferramenta poderosa na área de serviços de saúde.

Neste contexto, para certos quadros clínicos, o monitoramento da respiração torna-se uma importante ferramenta (GARGIULO et al., 2015). O uso do *biofeedback* combinado com a tecnologia vestível pode auxiliar no tratamento de pessoas com transtornos, distúrbios, crises de ansiedade e do pânico e também no gerenciamento do estresse. É possível até mesmo auxiliar um atleta a aumentar seu desempenho, utilizando técnicas de *biofeedback* em seus treinos (SILVA, 1999).

De acordo com Jonas (2001), o *biofeedback* começou com as primeiras pesquisas clínicas realizando tratamentos com a enxaqueca, hipertensão e entre outras. Atualmente, o campo de aplicação do *biofeedback* é vasto, combinando com outras áreas de estudos da medicina e da psicologia, este pode se tornar uma ferramenta muito poderosa. Com o auxílio do *biofeedback*, o indivíduo poderá modificar suas reações fisiológicas e com isto ele pode amenizar o uso de tratamentos farmacológicos, de maneira a evitar efeitos colaterais.

Deste modo, haja visto as diversas aplicações do uso do *biofeedback* e de que a respiração está fortemente interligado com vários outros ritmos do corpo. Assim, a respiração torna-se uma fonte importante de *biofeedback*, que pode transmitir diversas informações (FREY et al., 2018). Neste âmbito da ciência e tecnologia, observa-se a

crescente necessidade do desenvolvimento e aplicação de equipamentos eletrônicos que promovam essa técnica, onde estes sejam capazes de monitorar grandezas fisiológicas de forma prática e confortável ao usuário.

À vista disso, neste projeto pretende-se desenvolver um sistema eletrônico vestível de *biofeedback*, de tal modo que este seja capaz de realizar a aquisição da variação da respiração através do uso de transdutores de deformação, digitalizando estes sinais e enviando-os à um dispositivo externo através de comunicação sem-fio *Wi-Fi*. Pretende-se que assim, seja desenvolvida uma ferramenta que possa ser utilizada durante treinamentos ou terapias, onde o indivíduo acompanhará a sua atividade respiratória por meio de um *software* de interface gráfica promovendo um *biofeedback*. Considerando-se estas características, durante o desenvolvimento deste trabalho, pretendeu-se obter um sistema integrado, sendo este composto por uma placa de aquisição e envio dos sinais respiratórios e também um *software* de visualização e processamento dos sinais recebidos.

1.1 OBJETIVOS

Os objetivos deste trabalho são apresentados a seguir, separados de forma a apresentar o objetivo geral e os objetivos específicos.

1.1.1 Objetivo Geral

Este projeto, tem por objetivo o desenvolvimento de um sistema eletrônico e computacional para o monitoramento da atividade respiratória aplicado como uma ferramenta de *biofeedback*. Através de um transdutor de deformação posicionado no abdomen, o sistema será capaz de realizar o monitoramento da respiração e desta forma retornar as informações ao usuário por meio da interface de um *software* aplicativo.

1.1.2 Objetivos Específicos

Dentre os principais objetivos específicos, destacam-se:

- Realizar revisão da literatura necessária para o desenvolvimento do projeto, reconhecendo as características de sistemas de *biofeedback* e integrando seus aspectos ao desenvolvimento do trabalho;

- Desenvolver o circuito de condicionamento do sinal da leitura do do transdutor de deformação elástico, para quantificação da atividade respiratória;
- Desenvolver o algoritmo da leitura dos dados provenientes da atividade respiratória;
- Elaborar o algoritmo de envio da atividade respiratória para um dispositivo externo;
- Desenvolver o *software* de interface do sistema, realizado os cálculos necessários para quantificar a atividade respiratória (algoritmo de calculo da taxa respiratória) e exibição de informações para o usuário.

1.2 PROBLEMA DO PROJETO

O problema central que norteia o desenvolvimento deste trabalho, configura-se no estabelecimento de um sistema para a realização do monitoramento da atividade respiratória. Neste contexto, pretende-se fundamentar uma arquitetura de sistema que possa mensurar a atividade respiratória, de tal forma que seja possível estabelecê-la como um sistema de *biofeedback*. Através de um extensômetro elástico, o sistema será capaz de realizar o monitoramento da respiração abdominal e desta forma retornar as informações através da comunicação sem-fio *Wi-Fi*. Estas informações serão disponibilizadas em um programa aplicativo. Deste modo, tem-se o desafio da integração de elementos eletrônicos e de *software* que estabelecerão o usabilidade e aplicabilidade do protótipo.

1.3 JUSTIFICATIVA

O conceito de *biofeedback* determina que através do monitoramento de atividades fisiológicas é possível adquirir maior controle sobre estas, podendo-se até mesmo manipulá-las em resposta ao estimulo causado pelo retorno das informações fisiológicas. A aplicação desta metodologia tem se mostrado altamente benéfica no tratamento médico de uma grande variedade de problemas médicos (ZHU; KONG; XIE, 2012).

Neste contexto, a atividade respiratória torna-se um foco de aplicação interessante desta técnica. O *biofeedback* pode ser utilizado em técnicas de treinamento respiratório, onde este visa adequar a atividade respiratória, possibilitando benefícios

como o aumento da ventilação pulmonar e redução da taxa respiratória. Além disso, aplicar a técnica de *biofeedback* também possui aplicação em estudos psicofisiológicos, dado que a atividade respiratória tem influencia no controle do sistema nervoso autônomo (ARIMA HIROTO ARAKI, 2015).

Deste modo, observa-se a importância do desenvolvimento de aplicações em equipamentos eletrônicos que promovam a técnica do *biofeedback*, onde estes sejam capazes de monitorar grandezas fisiológicas de forma prática e confortável ao usuário, tal como este trabalho propõe através da utilização de extensômetros no monitoramento da atividade respiratória.

1.4 METODOLOGIA

A metodologia para o desenvolvimento deste trabalho tem como base uma pesquisa exploratória que visa o desenvolvimento de um protótipo. O sistema que será projetado e desenvolvido, será utilizado para quantificar e qualificar a atividade respiratória para promover um *biofeedback*. Dentro do desenvolvimento deste projeto, tem-se os procedimentos:

1. Analisar a literatura, através de uma pesquisa bibliográfica, em busca de soluções para o desenvolvimento do protótipo, focando nas característica do transdutor utilizado e nas técnicas disponíveis para aquisição de seu sinal bem como o interfaceamento do sistema com o seu usuário;
2. propor a arquitetura do sistema que abrange o desenvolvimento do protótipo, utilizando os conceitos encontrados na pesquisa bibliográfica;
3. desenvolver o circuito de aquisição de sinais;
4. desenvolver o *software* de interface gráfica, através da definição de requisitos feitos previamente;
5. testar o circuito para validar os conceitos e a funcionalidade do protótipo;
6. integrar o circuito ao modulo que irá digitalizar e enviar os sinais provenientes do circuito;
7. integrar o *software* ao sistema de aquisição;
8. realizar a montagem do sistema em um individuo da para teste do sistema como um todo, será feita a aquisição dos sinais durante da atividade respiratória em um dos integrantes da equipe, analisando-se a amplitude, o período e o espectro em

frequência do sinal;

9. analisar os dados de teste feitos com o sistema, levantando-se as possibilidades de melhoria.

1.5 ESTRUTURA DO TRABALHO

Com os assuntos apresentados, no atual capítulo mostro-se a motivação pelo projeto a ser desenvolvido bem como os objetivos e a metodologia. No capítulo 2 é feito uma revisão bibliográfica dos temas a serem abordados durante o projeto, são estes: *biofeedback*, tipos de respiração, transdutor de deformação, circuitos de condicionamento de sinal e técnicas de processamento e quantificação da atividade respiratória. No capítulo 3 é apresentado o desenvolvimento do projeto, explicitando as razões pelas escolhas dos componentes e técnicas para a implementação deste. No capítulo 4 apresentam-se os resultados obtidos com o desenvolvimento do protótipo proposto no projeto. No capítulo 5, tem-se o cronograma a ser seguido e no capítulo 6 são apresentadas as conclusões sobre projeto, fazendo uma avaliação da sua aplicabilidade e projeções para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, primeiramente será abordado o conceito de *biofeedback*. Em seguida, serão estudadas os tipos de respiração e os aspectos que poderão ser quantificados para a confecção de um sistema que realiza seu monitoramento. Serão utilizados extensômetros elásticos para realização desta funcionalidade, relacionando a medida de extensão das partes do corpo envolvidas no processo de respiração. Desta forma, serão estudados quais os aspectos físicos e formas de utilização dos extensômetros, bem como práticas disponíveis para o tratamento do sinal proveniente deste tipo de transdutor. Os sinais de saída do circuito de aquisição dos extensômetros serão transmitidos via *Wi-Fi* para uma aplicação em um computador que processará estes dados. Desta forma são estudadas técnicas de filtros digitais e de cálculo de taxas respiratórias. Será utilizado o módulo *Wi-Fi* ESP32, assim serão descritos alguns dos seus aspectos e funcionalidades.

2.1 *BIOFEEDBACK*

O *biofeedback* é um procedimento que permite ao indivíduo, através de uma técnica de terapia e exercícios, obter a capacidade de autorregulação de suas reações fisiológicas. O termo *biofeedback* vem da junção de duas palavras, *bio* do grego, que significa vida e o *feedback* do inglês que a tradução é o retorno de informações. Desta forma, entende-se por *biofeedback* como sendo o retorno imediato de informações no qual o individuo está sujeito ao tratamento. As aquisições destas informações podem ser obtidas através de sensores eletrônicos, devidamente calibrados, os quais realizam a leitura de dados dos processos fisiológicos como atividade cerebral, frequência cardíaca, pressão arterial, respiração e entre outras atividades (ROSSI, 2017).

O indivíduo, com o auxílio do retorno destas informações, pode melhorar sua qualidade de vida, assim como seu desempenho realizando esta técnica. À vista disso, percebe-se que estas atividades, que alguma das vezes são considerados eventos involuntários, passam a ter um controle mais efetivo com auxílio desta técnica. Sendo possível conscientizar o indivíduo, o desenvolvendo um melhor controle sobre estes processos fisiológicos, para assim poderem adquirir mais confiança no controle

voluntário dos mesmos. Com isto, conforme há variação da resposta fisiológicas, o paciente passa a observar as informações de imediato e com precisão, seja através do monitor ou outro meio que seja possível a disponibilização dos dados.

De acordo com Chaves (2017), esta técnica tem crescido muito com o avanço da tecnologia, possibilitando que seja aplicada em uma vasta área para tratamentos, sendo tanto área psicológica quanto fisiológicas. Entre as áreas pode-se citar aluma das doenças:

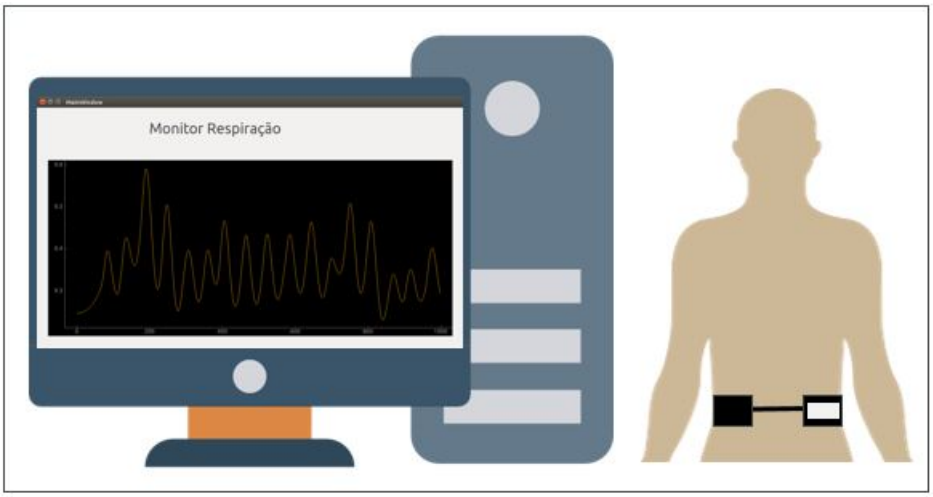
- quadro ansiosos, síndrome do pânico;
- depressão;
- problemas musculares;
- insônia;
- gerenciamento de estresse.

Vale ressaltar que o *biofeedback* não fica limitado apenas a estes tipos de terapia, segundo Silva (1999), esta técnica de retorno imediato de informações podem auxiliar um atleta a alcançar um desempenho muito superior do qual já se encontra, através de treinamentos e monitoramento dos ciclos respiratórios devidamente realizados. Enquanto o atleta realiza seu treino, com o auxílio do *biofeedback*, pode verificar sua respiração, assim como as atividades cerebrais, sendo estes processos fisiológicos aos quais o atleta não tem acesso sem o auxílio desta ferramenta. Assim, ainda de acordo com Silva (1999), defende o uso do *biofeedback* em treinamentos de atletas para auxiliarem no controle destes processos, ajudando a obter um controle melhor na respiração. Com isto, pode-se obter uma redução nos sintomas de ansiedade nos atletas em épocas de competição, em que, normalmente, estes níveis de estresse e ansiedade são mais elevados.

Além dessa grande variedade de aplicações, esta técnica tem outros fatores positivos pelo fato de ser um procedimento não invasivo e indolor. Pode-se, ainda, citar que não há efeitos colaterais devido ao fato de ser um tratamento não farmacológico, ou seja, livre de medicamentos. Desta forma, observa-se que é um tratamento seguro e confiável, no qual o indivíduo consegue ter um controle melhor em sua evolução durante o tratamento. São vários equipamentos capazes de realizar realizar tal procedimento, sendo a eletromiografia a mais usada. No entanto, tem-se a eletroencefalograma que monitora as atividades cerebrais. O esfigmomanômetro auxilia no sistema vascular e cardiovascular do indivíduo (CHAVES, 2017).

Na FIGURA 2.1, tem-se a ilustração de um exemplo de sistema de *biofeedback*, neste exemplo pode-se evidenciar a utilização de sensores que quantizam os sinais biológicos, sendo estes disponibilizados ao usuário através de um programa aplicativo que irá ser a interface do sistema.

FIGURA 2.1 – REPRESENTAÇÃO DE *BIOFEEDBACK*



Fonte: O autor (2018)

2.2 TIPOS DE RESPIRAÇÃO

Para as células do corpo humano reproduzirem energia, necessita-se de oxigênio no sistema, o qual é possível captar através da respiração.

A respiração é um processo indispensável ao ser humano, sendo este dividido basicamente, em dois movimentos, o primeiro é inspiração e depois a expiração.

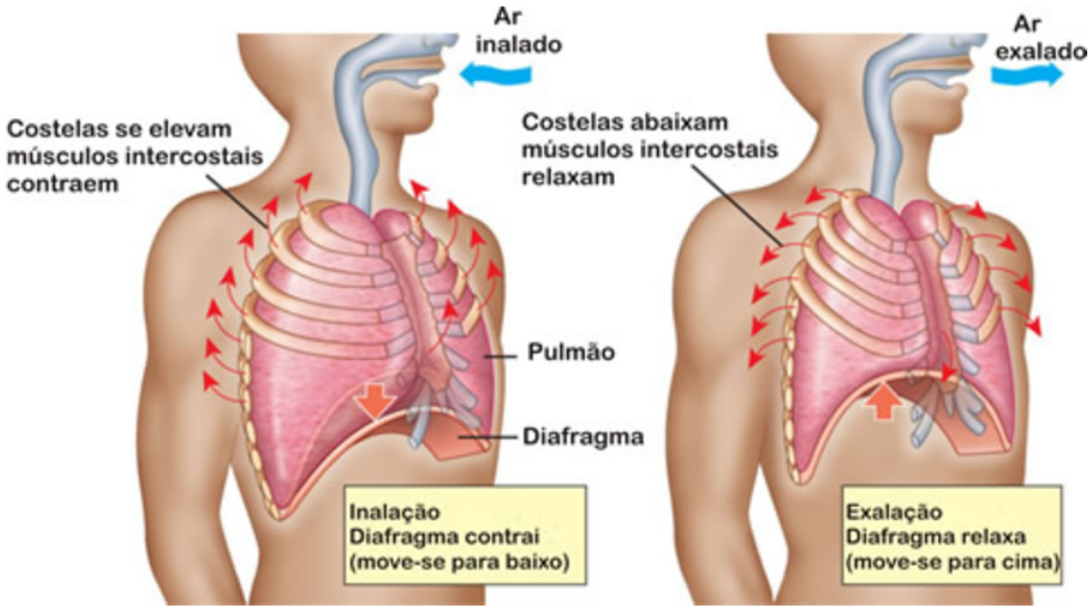
O movimento de inspiração da respiração, FIGURA 2.2, é o processo que permite a entrada de ar, ou seja, a sucção de ar para o organismo. Neste movimento ocorre a contração da musculatura do diafragma e dos músculos intercostais, de maneira que as costelas elevam-se (tórax), ou seja, a parede lateral do corpo se expande. Este movimento cria um devido espaço para a entrada do ar, desta forma, tem-se o aumento no volume da caixa torácica (AMABIS, 1999).

Já no movimento de expiração ocorre a saída de ar absorvido no processo anterior. Com isto, tem-se o relaxamento da musculatura do diafragma, assim como dos músculos intercostais, que são os músculos entre as costelas (GUIDI, 2017).

Como observado na FIGURA 2.2, o processo descrito é o oposto da inspiração, ou seja, neste processo tem-se uma redução do volume da caixa torácica, retornando

ao seu tamanho de repouso. Desse modo, tem-se a retração dos pulmões resultando na pressão interna maior que a externa, ao mesmo tempo que abdômen expande-se culminando então, na saída do ar anteriormente inspirado (AMABIS, 1999).

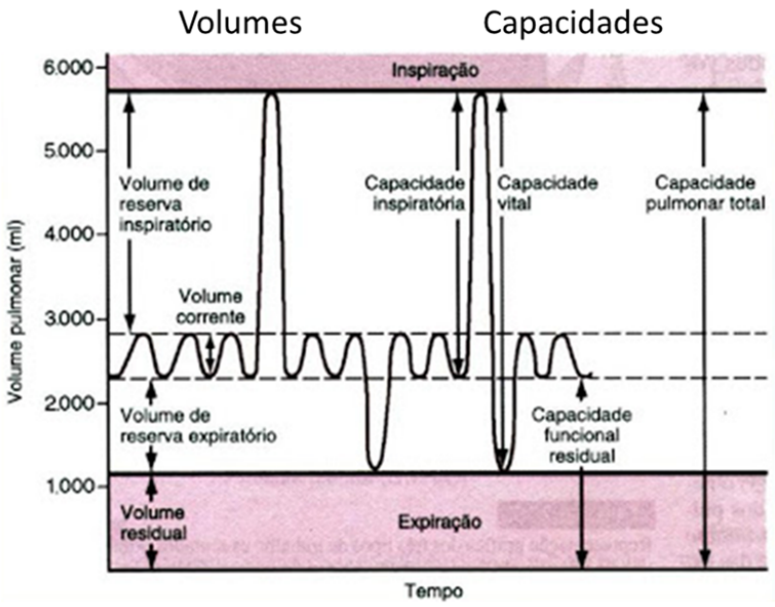
FIGURA 2.2 – PROCESSO INSPIRAÇÃO E EXPIRAÇÃO



Fonte: Guidi (2017)

Quando tem-se os dois movimentos de inspiração e expiração, um após o outro, é realizado o ciclo ventilatório. A quantidade de vezes que este ciclo é completado durante um minuto, é chamado de frequência respiratório (FR) e normalmente uma pessoa adulta, tem uma FR de cerca de 12 a 16 ciclos por minuto (GOZZI, 2017). Nos movimentos de inspiração e respiração descritos anteriormente, o volume pulmonar varia com o tempo em medida ao qual o processo ocorre. Tem-se, neste processo, quatro volumes pulmonares, sendo o volume de reserva inspiratório (VRI), o volume de reserva expiratório (VRE), o volume corrente (VC), e o volume residual (VR). Em cada ciclo respiratório tem-se uma variação do volume com a inspiração ou expiração, este volume é o VC. O VRI é a variação do volume com uma inspiração profunda, um pouco mais forçada. Já o VRE é ao contrário do VRI, sendo a variação do volume com uma expiração mais forçada. No entanto, mesmo realizando uma expiração forçada, tem-se um volume de ar que permanece no interior dos pulmões A FIGURA 2.3 representa estes quatro volumes (GOZZI, 2017).

FIGURA 2.3 – PROCESSO INSPIRAÇÃO E EXPIRAÇÃO



Fonte: Gozzi (2017)

2.3 TRANSDUTOR DE DEFORMAÇÃO

Transdutores são componentes que permitem a conversão de uma forma de energia em outra. Dentre os tipos de transdutores existentes, destacam-se os transdutores resistivos, que caracterizam-se por variarem a sua resistência em torno de um valor inicial em detrimento da variação de uma grandeza física (NORTHROP, 2005). Para realizar a medição de deformações, utilizam-se transdutores denominados extensômetros ou *Strain Gauges*. Estes sofrem uma alteração em sua resistência devido a mudanças sofridas no comprimento, diâmetro e resistividade (WEBSTER, 2009).

Para evidenciar o funcionamento dos extensômetros, observar-se a equação 2.1 que descreve de basicamente a resistência em função da resistividade $\rho\ (\Omega \cdot m)$, do comprimento $L\ (m)$ e da área $A\ (m^2)$. Levando-se em conta uma mudança diferencial de resistência R, pode-se também definir a equação 2.2 (WEBSTER, 2009).

$$R = \frac{\rho \cdot L}{\mathsf{A}} \tag{2.1}$$

$$dR = \frac{\rho \cdot dL}{A} - \rho \cdot A^{-2} \cdot dA + L \cdot \frac{d\rho}{A} \tag{2.2}$$

Considerando-se uma mudança finita na variação dos elementos e também dividindo-se a equação 2.2 pela equação 2.1, obtém-se a equação 2.3, que descreve mudanças relativas de resistência em função de um valor inicial (WEBSTER, 2009).

$$\frac{\Delta R}{R} = \frac{\Delta L}{L} - \frac{\Delta A}{A} + \frac{\Delta \rho}{\rho} \tag{2.3}$$

Segundo Webster (2009), através do coeficiente de Poisson (μ), pode-se relacionar a mudança de diâmetro com o comprimento. Desta maneira, a equação 2.3 pode ser reescrita na forma da equação 2.4. O primeiro termo da equação representa o efeito dimensional na variação da resistência e o segundo está relacionado ao efeito piezoelétrico.

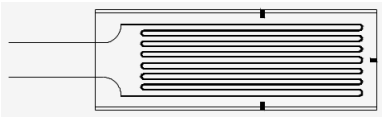
$$\frac{\Delta R}{R} = (1 + 2 \cdot \mu)\frac{\Delta L}{L} + \frac{\Delta \rho}{\rho} \tag{2.4}$$

Dividindo toda a equação 2.4, pelo seu segundo termo (relação de comprimento), define-se o fator do extensômetro G, apresentado na equação 2.5. De acordo Webster (2009), para metais condutores, o fator de maior influência é o dimensional e para semicondutores é o fator piezoelétrico.

$$G = \frac{\dfrac{\Delta R}{R}}{\dfrac{\Delta L}{L}} = (1 + 2 \cdot \mu) + \frac{\dfrac{\Delta \rho}{\rho}}{\dfrac{\Delta L}{L}} \tag{2.5}$$

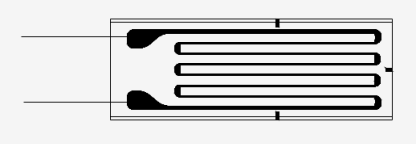
De acordo com Andolfato, Camacho e Brito (2004), existem alguns tipos de extensômetros comercias para medir a deformação em corpos de prova, podendo-se citar o do tipo fio (FIGURA 2.4) e do tipo lâmina (FIGURA 2.5).

FIGURA 2.4 – EXTENSÔMETRO DO TIPO FIO



Fonte: Andolfato, Camacho e Brito (2004)

FIGURA 2.5 – EXTENSÔMETRO DO TIPO LÂMINA

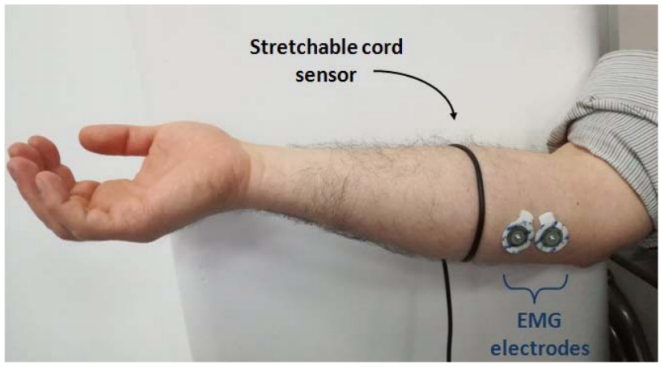


Fonte: Andolfato, Camacho e Brito (2004)

Além de *Strain Gauges* que são comumente disponíveis comercialmente, têm-se os *strain gauges* de resistência elástica. Eles são extremamente utilizados em aplicações biomédicas, especialmente na determinação de atividades cardiorrespiratórias e no monitoramento pletismográfico (medição de volume) (WEBSTER, 2009).

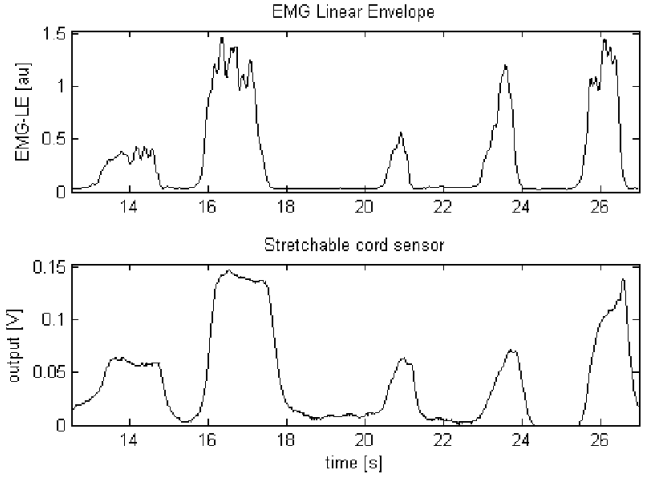
Neste trabalho, será utilizado um tubo condutivo elástico que quando esticado sobre variação na resistência. Utilizando-se este tipo de transdutor, Bifulco et al. (2017) realizou medidas de contrações musculares para controle uma prótese de mão. Na descrição dos experimentos, através de um circuito de instrumentação, foi possível obter resultados positivos na medição dos movimentos musculares do braço (FIGURA 2.6) e compara-los com envelopes calculados a partir de sinais eletromiográficos (FIGURA 2.6).

FIGURA 2.6 – POSICIONAMENTO DE DO CONDUTOR ELÁSTICO



Fonte: Bifulco et al. (2017)

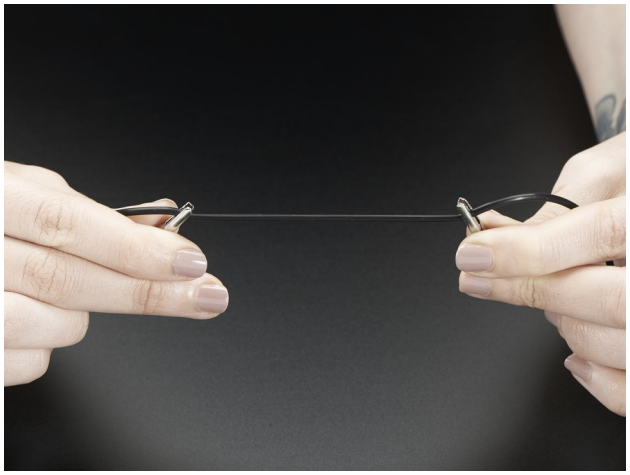
FIGURA 2.7 – COMPARATIVO ELETROMIOGRAFIA E CONDUTOR ELÁSTICO



Fonte: Bifulco et al. (2017)

Comercialmente, existem alguns transdutores resistivos flexíveis, estes são usualmente fabricados através de uma combinação de silicone e carbono, sendo materias originalmente utilizados para selagem eletrostática (OSMAN; HAYDAR-AHMAD; HAGE-DIAB, 2015). Neste sentido, no artigo apresentado por Varaki, Breen e Gargiulo (2017), analisou-se uma corda elástica condutiva que é distribuída pela empresa Adafruit (FIGURA 2.8). Este consiste em um tubo altamente elástico impregnado com carbono e que tem sua resistência variada quando é esticado. Estes transdutores são baratos, a prova de água e podem ser aplicados junto a roupas fazendo medidas do volume respiratório ou cardíaco (VARAKI; BREEN; GARGIULO, 2017).

FIGURA 2.8 – TUBO ELÁSTICO CONDUTIVO (ADAFRUIT)



Fonte: Adafruit (2018)

Pode-se também citar o transdutor fabricado pela empresa Images IC, segundo Yildiz, Mutlu e Alici (2016) este componente é feito de um polímero que varia a resistência quando esticado. Em Yildiz, Mutlu e Alici (2016) tal componente foi analisado, colocando-se em evidência a sua alta sensibilidade.

FIGURA 2.9 – TUBO ELÁSTICO CONDUTIVO (IMAGES IC)

Fonte: Yildiz, Mutlu e Alici (2016)

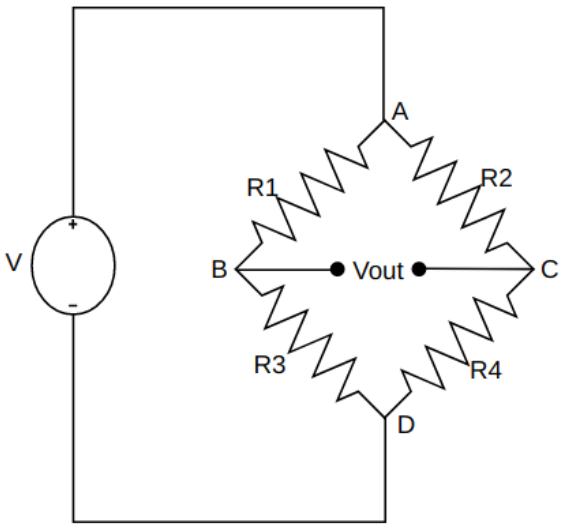
2.4 CIRCUITOS DE CONDICIONAMENTO DE SINAL

Como comentado anteriormente, será utilizado um condutor flexível de resistência variável como transdutor. Pretende-se utiliza-lo para monitorar a intensidade da atividade respiratória através da movimentação do abdômen. Deste modo, necessita-se que técnicas de condicionamento de sinais sejam aplicadas para que o sinal proveniente do dispositivo seja quantizado e analisado posteriormente.

2.4.1 Ponte de Wheatstone

Na medição do sinal proveniente do extensômetro, tem-se que uma deformação aplicada ao dispositivo, causará uma pequena mudança em sua resistência inicial. Assim, para garantir que a sua resposta a uma deformação seja medida com maior fidelidade possível, usualmente utiliza-se o circuito ponte de Wheatstone (NORTHROP, 2005). A ponte de *Wheatstone* configura o circuito da FIGURA 2.10. Este circuito permite mensurar uma pequena mudança de resistência relativo a um alto valor de resistência inicial, provendo uma alta resolução e sensibilidade. A ponte de Wheatstone é composta pela combinação de quatro resistores, onde um destes pode ser substituído pelo extensômetro. A saída do circuito é uma tensão diferencial (FIGURA 2.10) e será obtida através da análise das tensões presentes no circuito (WEBSTER, 2009).

FIGURA 2.10 – CIRCUITO PONTE DE *WHEATSTONE*



Fonte: O autor (2018)

Tomando-se como base o circuito da FIGURA 2.10, pode-se obter a tensão

V_{AB} fazendo-se um divisor de tensão, chegando-se na equação 2.6.

$$V_{AB} = \frac{R_1 V}{R_1 + R_3} \tag{2.6}$$

Da mesma forma, para tensão V_{AC} tem-se a equação 2.7.

$$V_{AC} = \frac{R_2 V}{R_2 + R_4} \tag{2.7}$$

Pela lei das malhas, tem-se a equação 2.8.

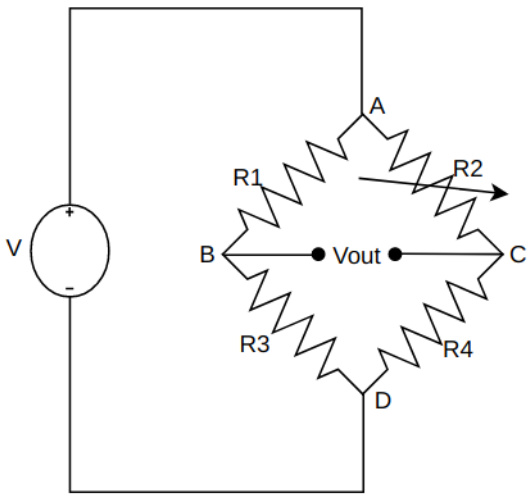
$$\mathrm{V}_{out} = -V_{AB} + V_{AC} \tag{2.8}$$

Substituindo a equação 2.6 e a equação 2.7 em 2.8 e manipulando-se os termos, obtém-se a equação 2.9.

$$\mathrm{V}_{out} = \mathrm{V}\frac{R_2 R_3 - R_1 R_4}{(R_1 + R_3)(R_2 + R_4)} \tag{2.9}$$

Das equações obtidas, observa-se que se R_1 = R_2 = R_3 = R_4, então V_{out} = 0 V. Neste trabalho, a deformação será medida apenas em uma direção, desta forma, se terá como base de análise o circuito da FIGURA 2.11, que tem R_2 como resistor variável, onde em sua posição o extensômetro elástico será posicionado.

FIGURA 2.11 – CIRCUITO PONTE DE *WHEATSTONE* COM RESISTOR VARIÁVEL



Fonte: O autor (2018)

Para análise deste circuito, considera-se a situação em que R_2 é o único resistor de valor variável tal como apresentado na equação 2.10.

$$R_2 = \Delta R + R \tag{2.10}$$

E os demais resistores possuem o mesmo valor:

$$R_1 = R_3 = R_4 = R \tag{2.11}$$

Substituindo-se 2.10 e 2.11 na equação 2.9, tem-se a equação 2.24.

$$V_{out} = \frac{\Delta R}{4R + 2\Delta R} V \tag{2.12}$$

Observa-se que para $R \gg \Delta R$, então pode-se considerar a equação 2.13.

$$\mathsf{V}_{out} = \frac{\Delta R}{4R} V \tag{2.13}$$

Analisando-se a equação 2.13, pode-se observar que a saída do circuito de ponte de Wheatstone, quando corretamente calibrado, terá a tensão de saída proporcional a variação de resistência de forma linear. Com isto, sendo esta característica relevante na conversão da variação de resistência do extensômetro em tensão (ANDOLFATO; CAMACHO; BRITO, 2004).

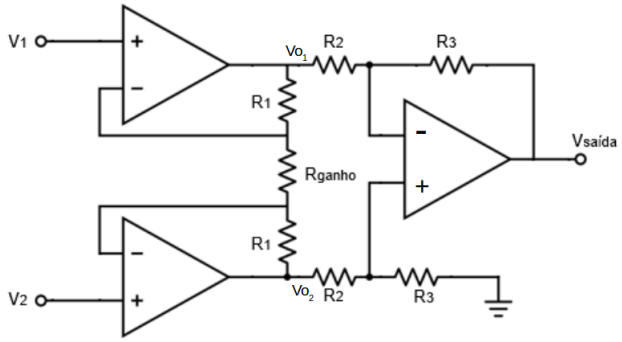
2.4.2 Amplificadores de instrumentação

Dado a análise feita anteriormente, observa-se que saída do circuito Wheatstone será uma saída de tensão diferencial de baixa amplitude, neste caso necessita-se que o sinal seja amplificado. Para amplificar sinais diferencias, utilizam-se amplificadores operacionais na topologia diferencial. A eficácia de um amplificador diferencial é medida em detrimento da facilidade pela qual esse amplificador rejeita os sinais de modo comum ao amplificar sinais diferenciais. Esta propriedade é identificada por uma grandeza denominada CMRR (*Common-Mode Rejection Ratio*) (SEDRA; SMITH, 2005). Ou seja, o CMRR é uma propriedade que mostra a capacidade do amplificador diferencial em atenuar qualquer tipo de ruído comum as suas duas entradas. Na prática, evidencia-se que amplificadores operacionais de qualidade, em topologias diferencias, devem possuir CMRR de no mínimo 100 dB (PERTENCE, 2007).

Visando evidenciar ao máximo as características provenientes do sinal do extensômetro, para amplificá-lo, será utilizada a topologia de amplificador de instrumentação. Este amplificador é um tipo especial de amplificador operacional, que permite obter características desejadas, tais como uma resistência de entrada extremamente alta, resistência de saída baixa e CMMR acima de 100 dB (PERTENCE, 2007).

Na FIGURA 2.12 tem-se uma topologia de amplificador de instrumentação amplamente disponível em circuitos comerciais.

FIGURA 2.12 – AMPLIFICADOR DE INSTRUMENTAÇÃO TOPOLOGIA 1



Fonte: O autor (2018)

Dado o circuito FIGURA 2.12, por meio da Lei das Correntes dos Nós de Kirchhoff, pode-se deduzir a tensão de saída do primeiro amplificador operacional. Considerado o curto virtual, tem-se a equação 2.14.

$$\frac{V_1 - V_2}{R_{ganho}} + \frac{V_1 - V_{o_1}}{R_1} = 0 \tag{2.14}$$

Isolando V_{o_1}, tem-se a equação 2.15:

$$V_{o_1} = \frac{R_1(V_1 - V_2)}{R_{ganho}} + V_1 \tag{2.15}$$

Da mesma forma, aplicando-se a análise para o segundo amplificador operacional, encontra-se a equação 2.16.

$$\frac{V_2 - V_1}{R_{ganho}} + \frac{V_2 - V_{o_2}}{R_1} = 0 \tag{2.16}$$

Isolando-se V_{o_2}, obtém-se:

$$V_{o_2} = \frac{R_1(V_2 - V_1)}{R_{ganho}} + V_2 \tag{2.17}$$

A configuração do terceiro amplificador é diferencial, sendo sua saída dada pela equação 2.18 (PERTENCE, 2007). Substituído as equações 2.15 e 2.17 e rearranjando os termos obtém-se a equação 2.19.

$$V_{saida} = \frac{R_3}{R_2}(V_{o_2} - V_{o_1}) \tag{2.18}$$

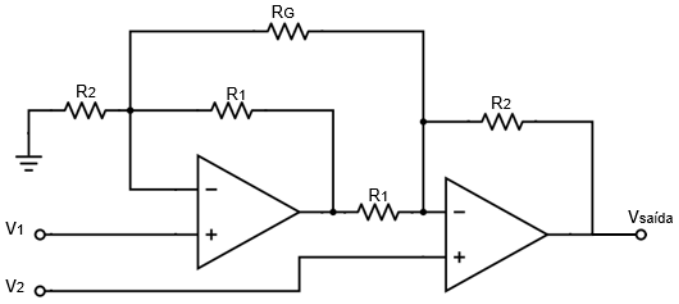
$$V_{saida} = (1 + \frac{2R_1}{R_{ganho}})\frac{R_3}{R_2}(V_2 - V_1) \tag{2.19}$$

Dado a 2.19, obtém-se a equação 2.20, que é o ganho do circuito a partir de uma entrada diferencial:

$$Ganho = (1 + \frac{2R_1}{R_{ganho}})\frac{R_3}{R_2} \tag{2.20}$$

Outra possibilidade de implementação de amplificador de instrumentação, seria utilizando-se apenas 2 amplificadores operacionais, tal como apresentado na FIGURA 2.13. Este amplificador possui características semelhantes a implementação anteriormente apresentada (KITCHIN; COUNTS, 2006). O ganho e a tensão de saída desta implementação encontra-se nas dadas equações 2.21 e 2.21 respectivamente.

FIGURA 2.13 – AMPLIFICADOR DE INSTRUMENTAÇÃO TOPOLOGIA 2



Fonte: O autor 2018

$$Ganho = 1 + \frac{R_2}{R_1} + \frac{2R_2}{R_G} \tag{2.21}$$

$$V_{saida} = (V_2 - V_1)Ganho \tag{2.22}$$

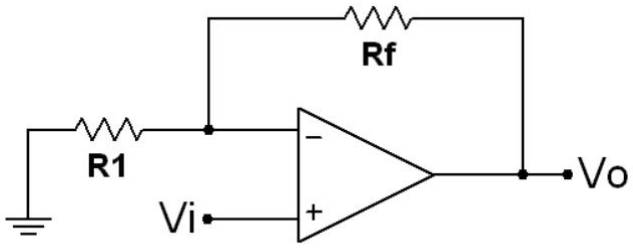
2.4.3 Amplificador não-inversor

Após a amplificação diferencial do sinal da saída da ponte de *Wheatstone*, este será amplificado novamente, dado a necessidade de ajuste do sinal a faixa de tensões do conversor analógico digital que será utilizado para a digitalização do sinal. Desta forma escolheu-se o amplificador não-inversor.

Este modo é mais uma das estruturas básicas de um amplificador operacional possível, diferentemente do inversor este modo não há uma defasagem no sinal de

saída. A configuração do não-inversor (FIGURA 2.14) tem-se a realimentação na entrada negativa, ou seja, uma parte do sinal de saída é reaplicado na entrada não-inversora do amplificador operacional (SEDRA; SMITH, 2005).

FIGURA 2.14 – AMPLIFICADOR NÃO-INVERSOR



Fonte: O autor (2018)

Observa-se na figura 2.14 os resistores R_1 e R_f estão conectados na entrada não-inversora. Através destes, é possível calcular o ganho com a equação 2.23:

$$Ganho = 1 + \frac{R_f}{R_1} \tag{2.23}$$

Desta forma, tem-se que a tensão de saída do amplificador não-inversor com a equação

$$V_o = (1 + \frac{R_f}{R_1})V_i \tag{2.24}$$

2.4.4 Filtros *anti-aliasing*

Como já comentado, o circuito de condicionamento do extensômetro será utilizado para o monitoramento da atividade respiratória. Onde durante esta atividade, as sequências de deformações relativas ao abdômen serão os parâmetros desejados para a saída do circuito na forma de uma tensão de saída.

Neste trabalho, necessita-se otimizar a estrutura do sistema que irá fazer a aquisição dos dados, procurando manter o compromisso de extrair a maior quantidade informação possível do circuito. Assim, serão utilizadas técnicas de filtragem de sinais.

Sabe-se que a taxa respiratória em adultos é em média entre 12 à 20 respirações por minuto (CLEVELANDCLINC, 2014), observa-se assim que a região espectral do sinal de saída da ponte pertence a baixíssimas frequências. Em Jeyhani et al. (2017) também evidência esta questão, mostrando-se que podem ser consideradas taxas de respirações 4 e 60 respirações por minuto. Para exemplificar, considerando-se que

a taxa de respiração é em média 20 respirações por minuto, transformando-a para frequência em hertz, obtém-se aproximadamente 0,3 Hz.

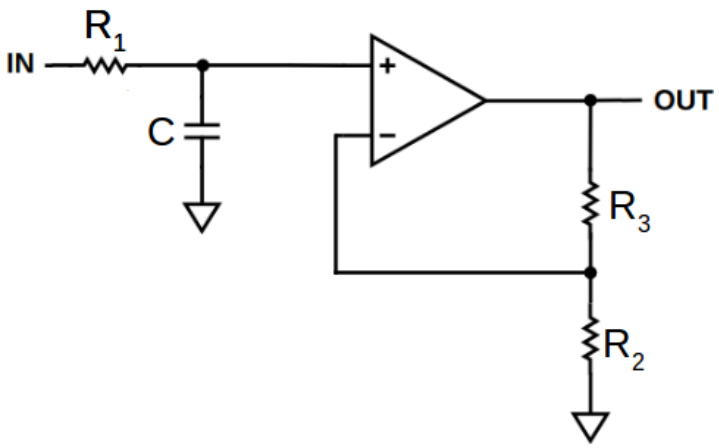
O sinal será processado pelo conversor analógico-digital do microcontrolador utilizado, deste modo, este deve possuir uma região espectral que é pelo menos a metade da frequência de amostragem do conversor (Teorema de Nyquist) (BAKER, 1999). Dado a característica de baixa frequência dos sinais envolvidos no processo de respiração, necessita-se que componentes de alta frequência provenientes de fontes de ruído sejam eliminadas, para garantir que este seja amostrado corretamente. Esta característica é garantida pelo condicionamento do sinal através de um filtro passa baixas também denominado filtro anti-*aliasing* (BAKER, 1999).

Os filtros passa-baixas atenuam as frequências acima da frequência de corte f_c desejada, sendo que esta característica é realizada por diversas topologias que podem ser realizadas de forma ativa e passiva (PERTENCE, 2007). Na FIGURA 2.16, pode-se observar um filtro ativo passa-baixas de primeira ordem. Nesta configuração, sinais com frequência acima da frequência de corte irão atenuar com um decaimento -20 dB/década. A frequência de corte e o ganho desta configuração são dados pelas equações 2.25 e 2.26 (PERTENCE, 2007).

$$f_c = \frac{1}{2\pi R_1 C} \tag{2.25}$$

$$Ganho = 1 + \frac{R_3}{R_2} \tag{2.26}$$

FIGURA 2.15 – FILTRO PASSA BAIXAS DE PRIMEIRA ORDEM



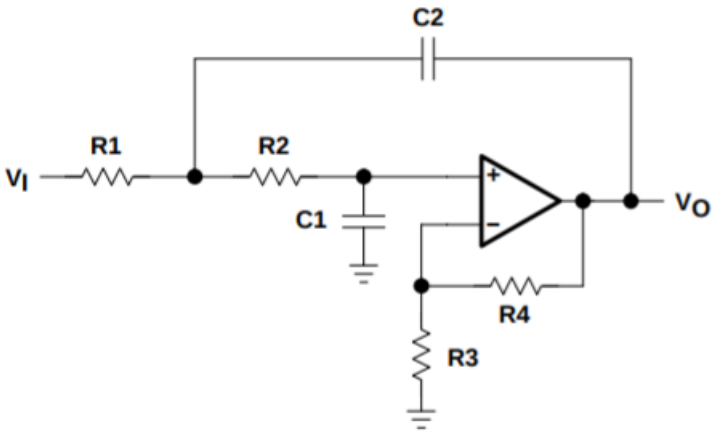
Fonte: Adaptado de Pertence (2007)

Uma topologia de filtro ativo de segunda ordem amplamente utilizada está apresentada na FIGURA 2.15, esta é denominada de *Sallen Key*. Esta topologia é classificada como de segunda ordem pois garante um decaimento de -40 dB/década, ou seja, o dobro da topologia de primeira ordem apresentada anteriormente (PERTENCE, 2007), onde na FIGURA 2.17, tem-se a comparação entre as respostas em frequência das duas topologias. Nas equações 2.27 e 2.28, tem-se respectivamente a frequência de corte e ganho do filtro *Sallen Key*.

$$f_c = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}} \tag{2.27}$$

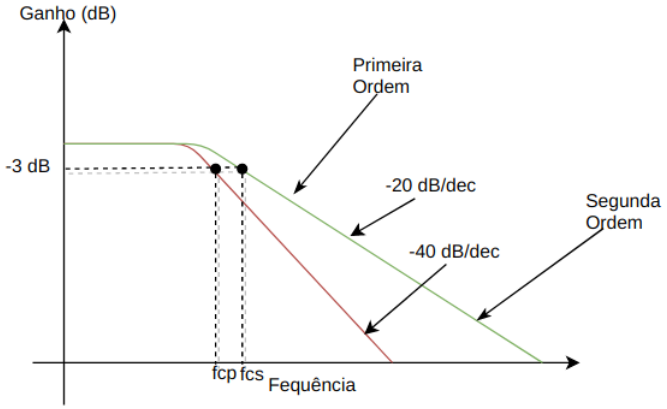
$$Ganho = 1 + \frac{R_4}{R_3} \tag{2.28}$$

FIGURA 2.16 – FILTRO PASSA BAIXAS DE SEGUNDA ORDEM



Fonte: Adaptado de Pertence (2007)

FIGURA 2.17 – RESPOSTAS FILTROS DE PRIMEIRA E SEGUNDA ORDEM



Fonte: Adaptado de Pertence (2007)

2.4.5 Filtros Digitais

Após a passagem do sinal pelo filtro *anti-aliasing* analógico, este será amostrado e enviado para um dispositivo externo (microcomputador) que realizará a filtragem do sinal recebido digitalmente, preprocessando-o tanto para visualização como para realização dos demais cálculos relacionados a atividade respiratória.

Os filtros digitais são sistemas discretos lineares invariantes no tempo (LIT). Frequentemente em aplicações práticas, estes são aplicados através de um *hardware* digital que tem por objetivo tratar um sinal de tempo contínuo que esteve sob uma amostragem periódica e que sofreu uma conversão de analógico para digital. O projeto e a implementação de um filtro digital tem por objetivo a determinação dos parâmetros da função de transferência (domínio da frequência) ou da equação a diferenças a coeficientes constantes (domínio do tempo) dentro de limites especificados no projeto (OPPENHEIM; SCHAFER, 2014).

Neste sentido, segundo Oppenheim e Schafer (2014), no processamento de sinais de tempo contínuo através de filtros digitais, se o critério de Nyquist for respeitado, ou seja, a frequência de amostragem for maior ou igual a duas vezes a frequência do sinal amostrado. Então, o sistema do filtro discreto se comporta em frequência como um sistema de tempo contínuo efetivo, que tem sua resposta em frequência contínua Ω (rad/s), definida pela equação 2.29.

$$H_{eff}(j\Omega) = \begin{cases} H(e^{j\Omega T}), & |\Omega| < \dfrac{\pi}{T} \\ 0, & |\Omega| \geq \dfrac{\pi}{T} \end{cases} \tag{2.29}$$

Observando-se a equação 2.29, tem-se o período de amostragem T como um de seus parâmetros, tornando-se possível converter a especificações do filtro de tempo contínuo para um filtro discreto através da relação $\omega = \Omega T$. Desta maneira, obtém-se a equação 2.30, que representa o mapeamento das frequências no domínio contínuo para domínio discreto normalizado (OPPENHEIM; SCHAFER, 2014):

$$H(e^{j\omega}) = H_{eff}\left(j\frac{\omega}{T}\right), |\omega| < \pi \tag{2.30}$$

Dado as características apresentadas, serão evidenciadas quais as técnicas e tipos de filtros que poderão ser utilizados no desenvolvimento deste trabalho, colocando em pauta os seus princípios de funcionamento.

2.4.5.1 Filtros IIR

Os filtros do tipo IIR (*Infinite Impulse Response*), são sistemas LIT de característica recursiva, ou seja, a saída do sistema depende de valores de saída anteriormente calculados (SMITH et al., 1997).

A representação matemática do filtro do tipo IIR pode ser feita por meio da sua função de transferência na forma de uma Transformada z, tal como apresentada na equação 2.31 (INGLE; PROAKIS, 2010) . Onde a_n e b_n são os coeficientes do filtro.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{n=0}^{M} b_n z^{-n}}{\sum_{n=0}^{N} a_n z^{-n}} = \frac{b_0 + b_1 z^{-1} + ... + b_M z^{-M}}{a_0 + a_1 z^{-1} + ... + a_N z^{-N}} \tag{2.31}$$

Considerando-se que $a_0 = 1$, a representação do filtro em equação a diferenças é identificada na equação 2.32, nota-se que a ordem do filtro genérico do tipo IIR é de valor N (INGLE; PROAKIS, 2010).

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k] \tag{2.32}$$

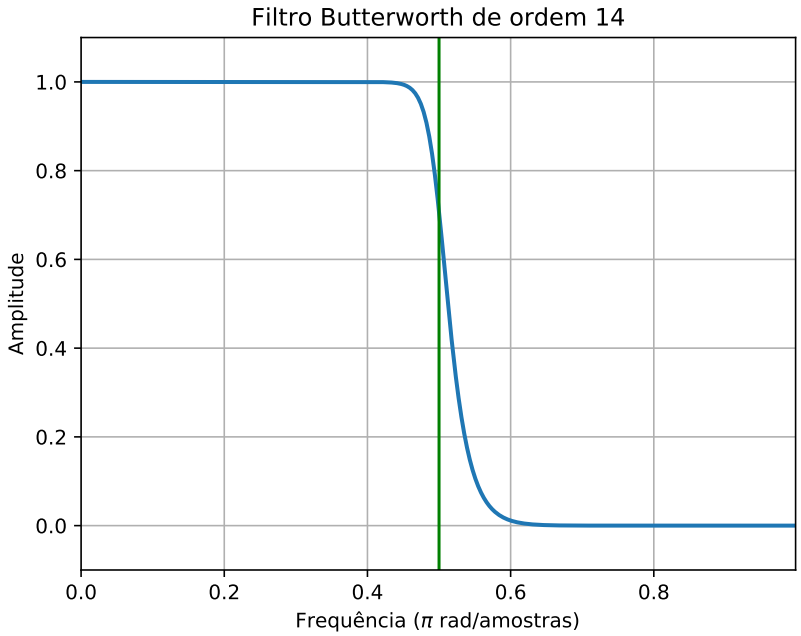
Dentro do contexto de projeto de filtros IIR, a técnica geralmente aplicada considera funções de aproximação utilizadas no projeto de filtros de tempo contínuo (analógicos), onde as funções são adaptadas e podem ser utilizadas para o projeto de filtros discretos diretamente, existindo *sofwares* que realizam o projeto dos filtros facilitando o encontro dos valores dos coeficientes (OPPENHEIM; SCHAFER, 2014).

Dentre as funções mais utilizadas, pode-se citar os polinômios de Butterworth, Chebyshev I, Chebychev II e Elíptico. O filtro passa baixas que utiliza a resposta Butterworth não possui oscilações, tanto na banda de passagem como na banda de rejeição (FIGURA 2.18). Já os filtros do tipo Chebychev do tipo I, possuem oscilações somente na faixa de passagem (FIGURA 2.19). Em contraste com o anterior, filtros Chebychev do tipo II possuem oscilações na faixa de rejeição e não na faixa de passagem (FIGURA 2.20). Por fim, o filtro Elíptico possui oscilações tanto na banda de passagem como na de rejeição (FIGURA 2.21).

Nota-se, a necessidade da análise da relação entre a ordem necessária para as alcançar as especificações do filtro e o tipo de resposta escolhida. Filtros com a mesma especificação de frequência de corte podem ser obtidos com diferentes ordens, porém escolhendo-se uma determinada resposta, obtém-se uma saída com distintos graus

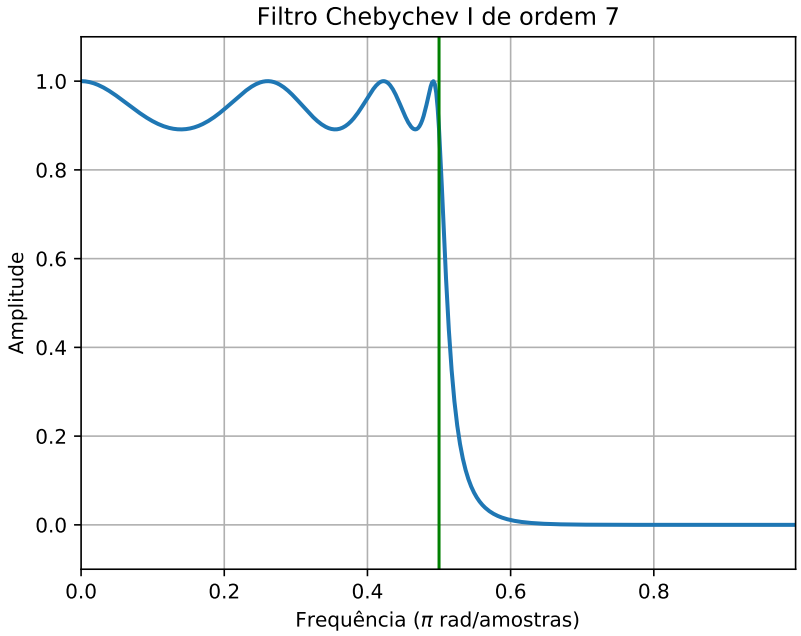
de oscilação tanto na faixa de passagem como na faixa de rejeição (OPPENHEIM; SCHAFER, 2014).

FIGURA 2.18 – FILTRO BUTTERWORTH DE ORDEM 14



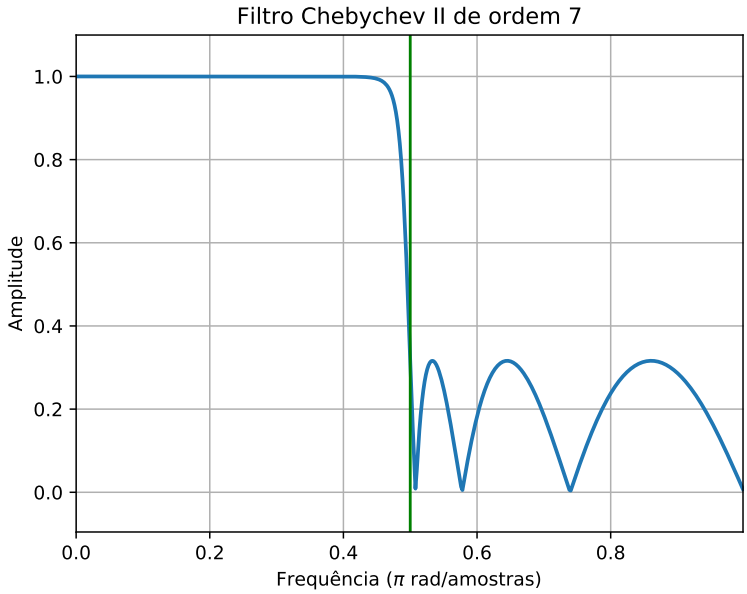
Fonte: Autor (2018)

FIGURA 2.19 – FILTRO CHEBYCHEV I DE ORDEM 7



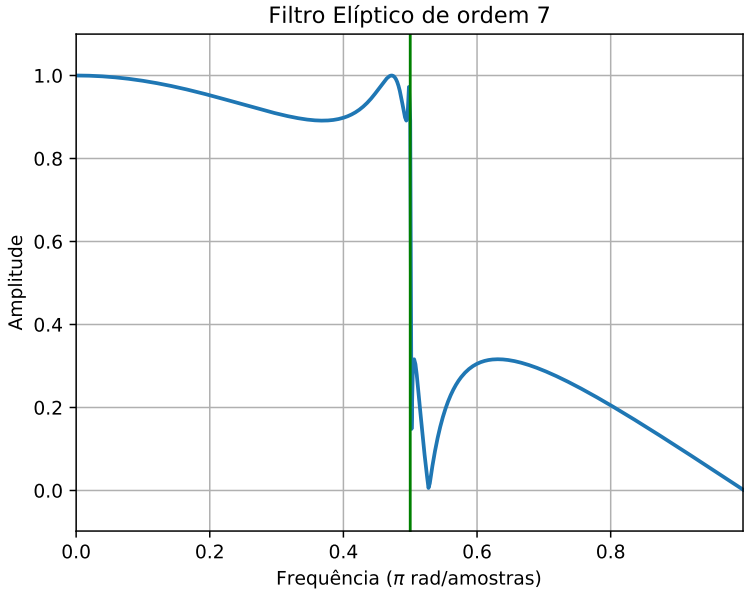
Fonte: O autor (2018)

FIGURA 2.20 – FILTRO CHEBYCHEV II DE ORDEM 7



Fonte: O autor (2018)

FIGURA 2.21 – FILTRO ELÍPTICO DE ORDEM 7



Fonte: O autor (2018)

2.4.5.2 Filtros FIR

Em contraste com os filtros do tipo IIR, os filtros FIR (*Finite Impulse Response*) são filtros não recursivos, ou seja, só dependem de entradas atuais e anteriores (SMITH

et al., 1997). Na equação 2.33, podendo-se também definir a resposta ao impulso de um filtro de ordem M por meio de sua Transformada z (INGLE; PROAKIS, 2010).

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{n=0}^{M} b_n z^{-n} = b_0 + b_1 z^{-1} + ... + b_M z^{-M} \tag{2.33}$$

De forma direta, a resposta ao impulso $h(n)$ é representada pela equação 2.36. Pode-se também definir a equação 2.35, que apresenta a generalização da equação a diferenças do filtro FIR (INGLE; PROAKIS, 2010).

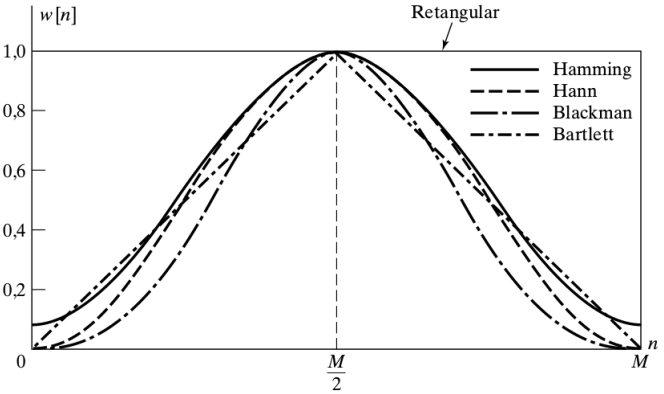
$$h(n) = \begin{cases} b_n, & 0 < n \leq M \\ 0, & fora \end{cases} \tag{2.34}$$

$$y[n] = b_0 x[n] + b_1 x[n-1] + ... + b_M x[n-M] \tag{2.35}$$

Segundo Oppenheim e Schafer (2014), o projeto de filtros FIR se baseia na utilização da técnica de janelamento, que tem por objetivo truncar a resposta de um filtro ideal, por uma janela de comprimento finito, resultando-se no em um filtro com a resposta ao impulso representeada pela equação 2.36 no tempo discreto. Na FIGURA 2.22, tem-se exemplos das principais janelas referenciadas na literatura. Também exemplifica-se FIGURA 2.23 um filtro FIR projetado com uma janela de tipo Hamming, observa-se que no projeto destes filtros, em sua maioria utiliza-se ordens elevadas para obter as especificações desejadas.

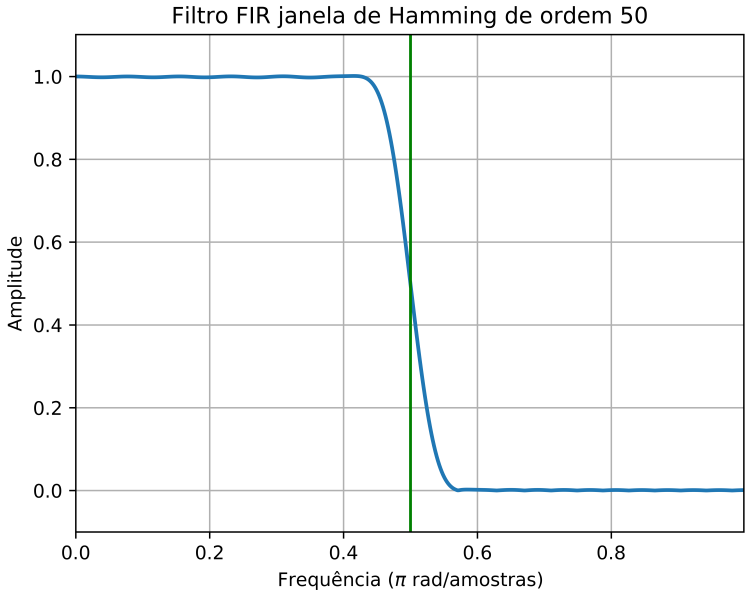
$$h[n] = \frac{sen[\omega_c(n - M/2)]}{\pi(n - M/2)} w[n] \tag{2.36}$$

FIGURA 2.22 – TIPOS DE JANELAMENTO



Fonte: Oppenheim e Schafer (2014)

FIGURA 2.23 – FILTRO FIR (JANELA HAMMING)



Fonte: O autor (2018)

2.5 ALGORITMOS DE CÁLCULO TAXA DE RESPIRATÓRIA

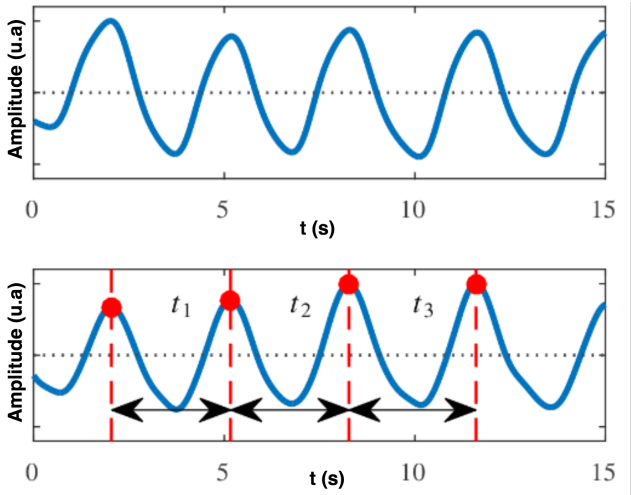
Após realizar a filtragem do sinal recebido, será desenvolvido um algoritmo de cálculo de taxa respiratória, ou seja, através deste algoritmo serão estimadas quantas respirações ocorreram em um determinado intervalo de tempo. De acordo Jeyhani et al. (2017), a taxa respiratória é considerada como um dos parâmetros mais importantes para caracterização das condições de saúde de um ser humano. Sendo possível derivá-la indiretamente a partir de sinais vitais tal como a eletrocardiografia, ou diretamente através de sistemas de medição de volume abdominal ou do tórax (pneumografia). Assim, serão apresentados os principais algoritmos relacionados ao cálculo de taxas respiratórias a partir de sinais extraídos dos movimentos respiratórios. Neste sentido, tem-se estimadores que realizam cálculos no domínio do tempo e estimadores no domínio da frequência (JEYHANI et al., 2017).

2.5.1 Detecção de Picos

Neste método temporal, escolhe-se um intervalo fixo de amostras para analisar e então os máximos locais são calculados. Cada máximo local corresponde a ocorrência de um ciclo respiratório que possui um correspondente instante de tempo de ocorrência. Após encontrados os instante de tempo de ocorrência de máximos locais, calcula-se a

distâncias de tempo entre estes instantes. Finalmente calcula-se a média das distâncias encontradas. Dividindo-se o valor encontrado pela frequência de amostragem e então multiplicando-se por 60, obtém-se a taxa respiratória estimada em respirações por minuto. Neste método também é estabelecido um critério de miníma distância horizontal entre ocorrência de respirações, que pode ser definido como 1 segundo. Ou seja, a ocorrência de máximos locais em intervalos menores que 1 segundo são descartados (JEYHANI et al., 2017). Na FIGURA 2.24, tem-se a ilustração do funcionamento deste algoritmo, observa-se a detecção dos picos e distâncias temporais, onde segundo Jeyhani et al. (2017) é um método aplicado a sinais que não possuem distorções e apresentam claramente a atividade respiratória como principal informação.

FIGURA 2.24 – ESTIMAÇÃO TAXA RESPIRATÓRIA (DETECÇÃO DE PICOS)



Fonte: Adaptado de Jeyhani et al. (2017)

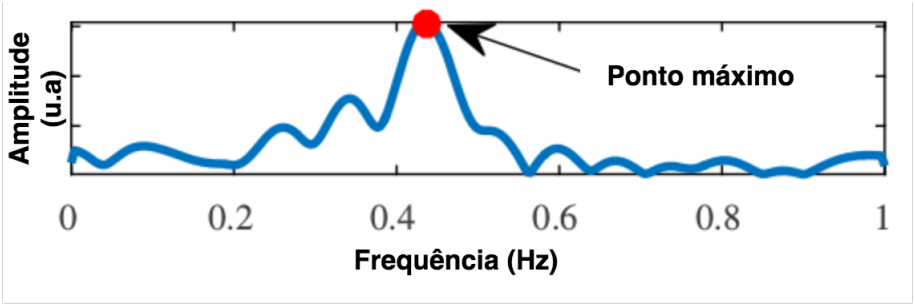
2.5.2 Contagem de Ciclos Respiratórios

Segundo Charlton, Villarroel e Salguiero (2016), também pode-se citar métodos temporais que estimam a taxa respiratória através da contagem de ciclos respiratórios. Estes métodos têm por objetivo detectar ciclos respiratórios em função de pares de máximos locais que excedam um valor de limiar. Após calcular estes valores, então são identificados como respirações confiáveis os pares que contém somente um valor de mínimo menor que zero em relação aos dados do sinal respiratório. Este tipo de abordagem pode ser empregada em situações onde o sinal possui oscilações que podem ser confundidas com a ocorrência de uma respiração (JEYHANI et al., 2017).

2.5.3 FFT (*Fast Fourier Transform*)

Este método destaca-se por calcular a taxa respiratória através da aplicação da transformada rápida de Fourier no sinal correspondente à atividade respiratória. Nesta técnica, dado um intervalo de amostras, inicialmente dizima-se o sinal (diminui-se a taxa de amostragem) para melhorar a resolução do espectro e considerar somente frequências em um intervalo correspondente a taxas respiratórias plausíveis ao ser humano (4 a 60 respirações por minuto) (CHARLTON; VILLARROEL; SALGUIERO, 2016). Em seguida, calcula-se a amplitude máxima do espectro do sinal, tornando-se possível encontrar a frequência correspondente que estimará a taxa respiratória das amostras analisadas (FIGURA 2.25) (JEYHANI et al., 2017).

FIGURA 2.25 – ESTIMAÇÃO TAXA RESPIRATÓRIA (FFT)



Fonte: Adaptado de Jeyhani et al. (2017)

2.6 MÓDULO DE ESP32

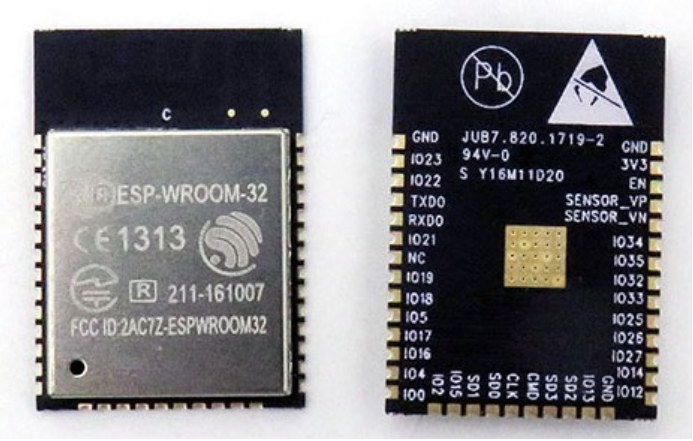
O ESP32 é um módulo *Wi-FI* (FIGURA 2.27), originalmente da empresa Espressif, que contém diversos modelos. Entre eles o modelo ESP-WROOM-32, que é um *System-on- a-Chip (SoC)* com *Wi-Fi* embutido e dois *Micro Controller Unit* (MCU) Xtensa LX6 de 32 bits e *clock* de 240MHz. Além do *Wi-Fi* embutido, tem-se o módulo de *Bluetooth* integrado. Desta maneira, este não é apenas um módulo *Wi-Fi*, podendo se tornar uma solução completa para o equipamento eletrônico que deseja um bom processador e com necessidade de conexão sem-fio (ESPRESSIF, 2018). O módulo possui uma memória interna para seu *firmware* e outra memória *flash* para armazenar dados. Pode ser no modo "*stand-alone*" e no modo estação, podendo se conectar a uma rede sem-fio, também possuindo o modo "*Access Point*".

Além dessas características o módulo chama atenção pelo seu tamanho reduzido com um elevado número de funcionalidades com um custo baixo (ESPRESSIF,

2018). Outras características relevantes do módulo são:

- antena *Wi-Fi* e *Bluetooth* embutida;
- padrão 802.11 b/g/n;
- protocolo TCP/IP embutido;
- consumo em *standby* menor que 1 mW;
- conversor ADC de 12 bits;
- tensão de operação 2,3 V à 3,6 V;
- programação pode ser feita nas linguagens LUA, C, *microPython*.

FIGURA 2.26 – MÓDULO ESP32



Fonte: Espressif (2018)

Neste projeto, levando-se em consideração a facilidade de utilização e também a de prototipagem, será utilizado o kit de desenvolvimento *NodeMCU*, que traz o chip do ESP32, em conjunto com os circuitos de alimentação e programação do módulo.

FIGURA 2.27 – NODEMCU - ESP32

Fonte: Espressif (2018)

3 DESENVOLVIMENTO

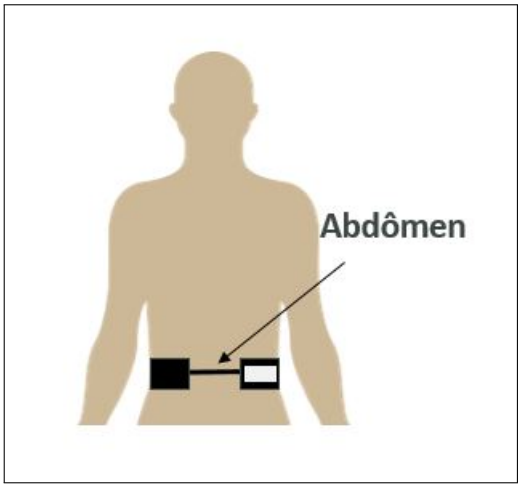
Neste trabalho, desenvolveu-se um sistema de *biofeedback* para fornecimento de informações da atividade respiratória. Dentro das etapas do trabalho, foram desenvolvidos o circuito de aquisição de sinais, o programa de leitura e envio de dados e por fim o *software* de processamento e retorno das informações ao usuário do sistema.

3.1 MONITORAMENTO RESPIRAÇÃO

A atividade respiratória será quantizada através de um dispositivo disposto na forma de cinto e que será tal como ilustrado na FIGURA 3.1. O cinto será acoplado a um transdutor elástico (extensômetro) e o seu respectivo circuito de condicionamento de sinal, dado as características apresentadas anteriormente na revisão bibliográfica. A partir movimentos da movimentação abdominal provocada pela atividade respiratória, o monitoramento ocorrerá por meio da variação da amplitude da tensão da saída do circuito.

O sinal será digitalizado por meio do conversor analógico-digital e será enviado através da comunicação sem-fio *Wi-Fi* para um dispositivo externo utilizando o módulo ESP32. Em seguida, através de *software* que será desenvolvido para um microcomputador, a atividade será monitorada e quantificada, promovendo um *biofeedback* ao usuário do sistema.

FIGURA 3.1 – ILUSTRAÇÃO POSICIONAMENTO DO CINTO



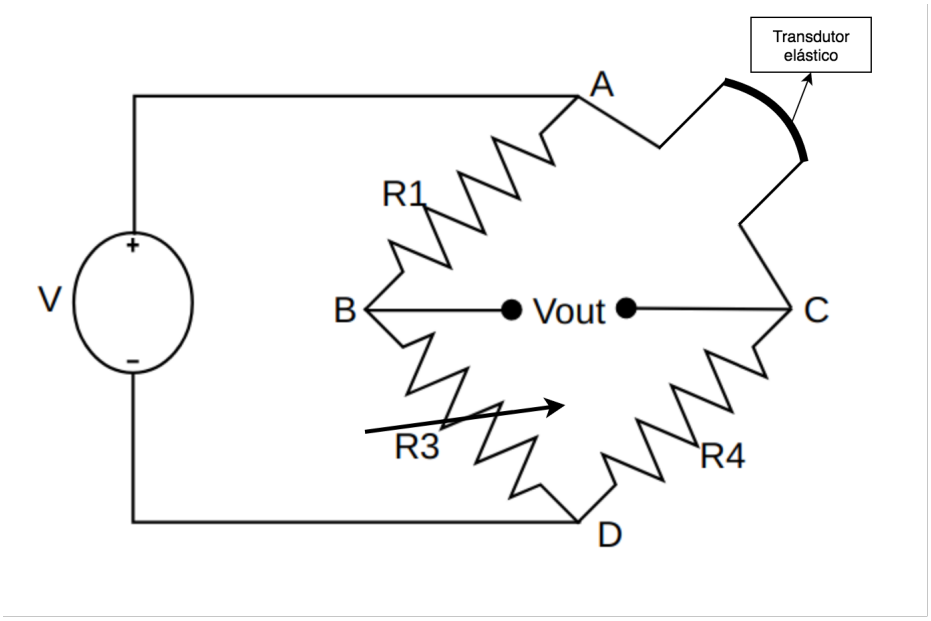
Fonte: O autor (2018)

3.1.1 Circuito de Aquisição

Como já comentado, no condicionamento do sinal gerado pelo extensômetro (condutor elástico), necessita-se da utilização de uma ponte de Wheatstone. A variação da resistência deste transdutor acontece a partir de uma resistência nominal fixa que é conhecida, sendo necessário calibrar os demais resistores para que estes possuam, em valor nominal, um valor próximo ao valor medido para o extensômetro em repouso (ANDOLFATO; CAMACHO; BRITO, 2004). Na FIGURA 3.2 observa-se a configuração que foi utilizada neste projeto, onde o transdutor elástico é posicionado como um dos elementos da ponte e também utiliza-se um *trimpot* para ajustar a saída da ponte para um valor de tensão positivo.

O transdutor utilizado é o condutor elástico (apresentado na seção 2.3) da fabricante Images IC, que possui uma resistência nominal medida de $690\ \Omega$. Desta forma, utilizou-se os resistores $R_1 = R_4 = 680\ \Omega$ com tolerância de 1% e um *trimpot* (R_3) de $1\ k\Omega$ para ajuste da saída.

FIGURA 3.2 – EXTENSÔMETRO NA PONTE DE *WHEATSTONE*



Fonte: O autor (2018)

A saída de tensão da ponte de Wheatstone é diferencial, geralmente necessitando de amplificação (NORTHROP, 2005). Para realizar a função de amplificação, usualmente utilizam-se amplificadores de instrumentação comerciais. Tais circuitos integrados possuem alta impedância de entrada e também altíssimo CMRR.

Desta forma, neste trabalho, utilizou-se o amplificador de instrumentação denominado INA128 da fabricante *Texas Instruments*. Este amplificador tem uma topologia com três amplificadores operacionais e possui baixo consumo e alta precisão, sendo altamente recomentado para aplicações de instrumentação biomédica. Na FIGURA 3.3 tem-se o esquemáticos e valores do componentes interno deste amplificador.

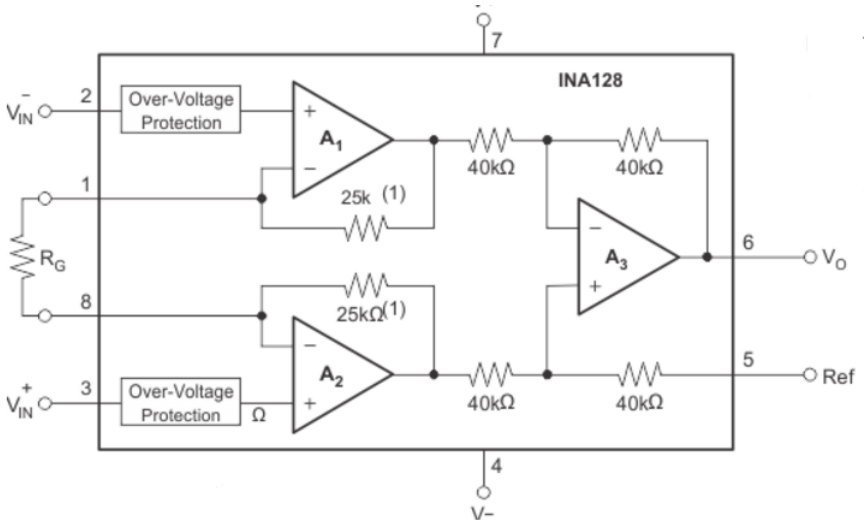
Tal como desejado, este amplificador de instrumentação possui um altíssimo CMRR, chegando até 130 dB (FIGURA 3.4). Ele utiliza um resistor externo para ajustar o ganho da tensão de saída do circuito, facilitando o seu ajuste dado o sinal de entrada, onde este ganho é dado pela equação 3.1 (TEXAS, 2015a).

Visando a amplificação diferencial da saída, para um ganho de aproximadamente $2\ V/V$, tem-se $R_G = 56\,k\Omega$ (comercial), substituindo na equação 3.1, obtém-se o resultado em 3.2. Observa-se que escolheu-se um ganho relativamente pequeno, pois neste projeto, a principal função do amplificador de instrumentação é amplificar o sinal diferencial de tal modo que possíveis ruídos de modo comum sejam eliminados. O ajuste da tensão de saída será feita por um amplificador não-inversor, trazendo-se maior flexibilidade, evitando-se possíveis saturações na saída do circuito.

$$Ganho = 1 + \frac{50k\Omega}{R_G} \tag{3.1}$$

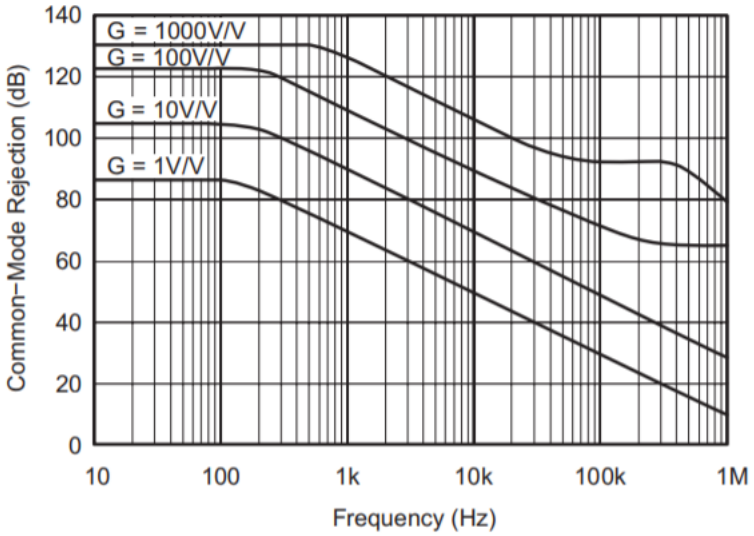
$$Ganho = 1 + \frac{50k\Omega}{56k\Omega} = 1,89\,V/V \tag{3.2}$$

FIGURA 3.3 – AMPLIFICADOR DE INSTRUMENTAÇÃO INA128



Fonte: Texas (2015a)

FIGURA 3.4 – COMPORTAMENTO CMRR INA128

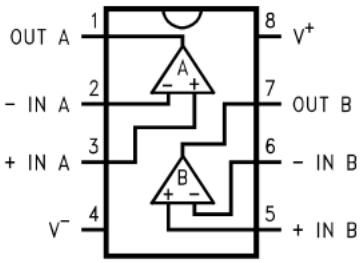


Fonte: Texas (2015a)

Após a passagem pelo amplificador de instrumentação, o sinal será amplificado novamente (ajuste de ganho) e filtrado (*anti-aliasing*). Desta forma, para o desenvolvimento destas funcionalidades, o circuito integrado LM6142 foi utilizado (FIGURA 3.5).

Este é um Amplificador Operacional duplo que tem a característica *rail-to-rail*, o que significa que a sua saída alcança aproximadamente a mesma tensão da alimentação. Dado este fato, este componente torna-se uma excelente escolha, pois provém a máxima excursão possível para sinais de entrada, sendo indicado para aplicações de instrumentação que necessitem operar com somente uma fonte, tal como uma bateria. As tensão de alimentação do LM6142 varia de $1,8\ V$ a $24\ V$ (TEXAS, 2013).

FIGURA 3.5 – ESQUEMÁTICO LM6142



Fonte: Texas (2013)

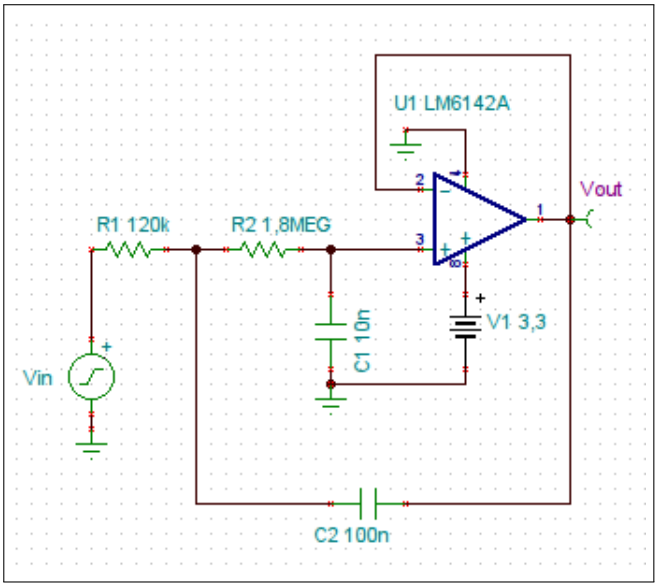
Como já comentado na revisão bibliográfica, utiliza-se filtros passa-baixas

para que evitar o efeito *aliasing* (BAKER, 1999). Dentro das topologias encontradas, escolheu-se por utilizar o filtro passa-baixas de segunda ordem, da topologia *Sallen Key*. Utilizando a equação 2.27, pôde-se calcular a frequência de corte para o circuito, visa-se eliminar o ruído da rede elétrica e os possíveis ruídos em alta frequência que poderiam interferir na amostragem do sinal que é feita pelo conversor analógico-digital que será utilizado. Assim, para os seguintes valores de componentes (valores comerciais), $C_1 = 10\ nF$, $C_2 = 100\ nF$, $R_1 = 120\ k\Omega$ e $R_2 = 1,8\ M\Omega$, obtém-se:

$$f_c = \frac{1}{2\pi\sqrt{1,8 \cdot 10^6 \cdot 120 \cdot 10^3 \cdot 10 \cdot 10^{-9} \cdot 100 \cdot 10^{-9}}} = 10,83\ Hz \tag{3.3}$$

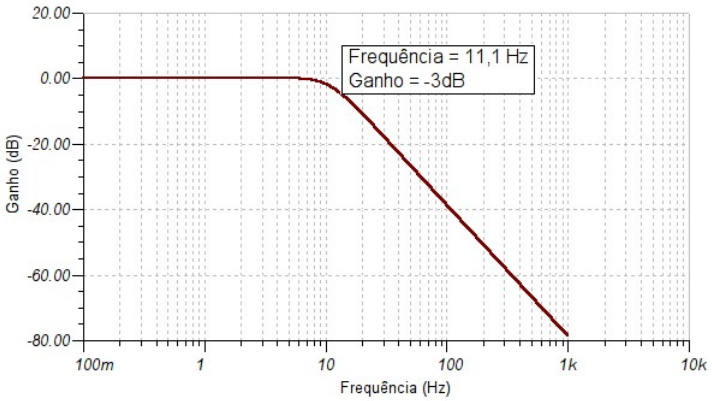
O sinal proveniente da atividade respiratória é de baixa frequência, tal como visto anteriormente, tem-se em média uma taxa de 20 respirações por segundo (CLEVELANDCLINC, 2014) para adultos, tendo-se assim que um sinal variará em uma frequência que é menor que $1\ Hz$ $(0,3\ Hz)$. Assim, a frequência encontrada torna a faixa de passagem da saída do sistema suficientemente confiável para realização da amostragem do sinal com frequências relativamente baixas. Utilizando-se o modelo *spice* do LM6142 e simulando-se o filtro através de uma simulação do tipo AC no *software* TINA-TI (FIGURA 3.6), obteve-se o a resposta frequência do circuito na FIGURA 3.7. Observa-se que obteve-se um valor de frequência de corte coerente relação a que foi calculada, onde para $-3\ dB$, tem-se $f_c = 11,1\ Hz$

FIGURA 3.6 – ESQUEMÁTICO SIMULAÇÃO DO FILTRO



Fonte: O autor (2018)

FIGURA 3.7 – RESPOSTA EM FREQUÊNCIA DO FILTRO



Fonte: O autor (2018)

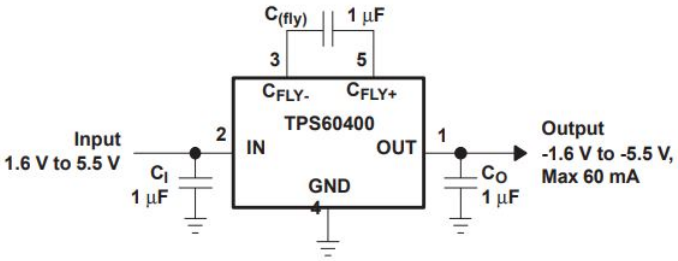
3.1.2 Alimentação

Tendo em vista que o dispositivo a ser implementado será um protótipo de *wearable device*, decidiu-se pela utilização de uma bateria de LiPo (*Lithium Polymer*) na forma de um *Power-bank* do fabricante Ideus. Este tipo de bateria tem ganhado espaço no mercado devido ao fato de poder terem mais de um *pack* e célula, também pela sua versatilidade no formato. A bateria terá uma carga elétrica nominal de 2500 mAh (IDEUS, 2018), sendo possível alimentar o ESP32 e os demais componentes do sistema de aquisição. Dentro deste contexto, deve-se analisar a alimentação dos demais componentes do circuito, tais como amplificadores operacionais e o amplificador de instrumentação.

Em específico, para a alimentação positiva do amplificador de instrumentação será possível utilizar o próprio pino de V_{cc} disponível no ESP32. No entanto, para suprir a necessidade da alimentação negativa deste, decidiu-se pela utilização do circuito integrado TPS6040.

O CI TPS6040 é um inversor de tensão chaveado, com faixa de operação entre $1,6\ V$ à $5,5\ V$ (TEXAS, 2015b). Para obter uma saída regulada negativamente, deve-se acrescentar capacitores de $1\ \mu F$ conforme visto na FIGURA 3.8 . Assim, dado que a tensão proveniente do pino V_{cc} do ESP32 será uma tensão de $3,3\ V$, conectando-se esta ao pino 2 do CI, tem-se uma tensão de saída para $-3,3\ V$. Com a utilização deste CI foi possível reduzir a complexidade do circuito e possíveis componentes discretos que poderiam ser utilizados para se gerar a tensão simétrica para o amplificador de instrumentação.

FIGURA 3.8 – TPS6040

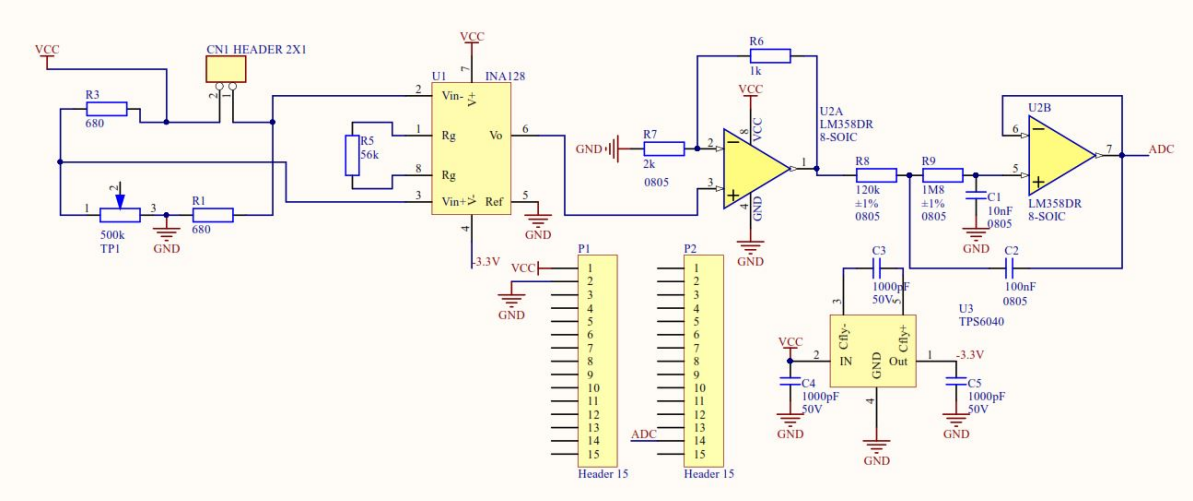


Fonte: Texas (2015b)

3.1.3 Desenvolvimento (Placa de Circuito Impresso)

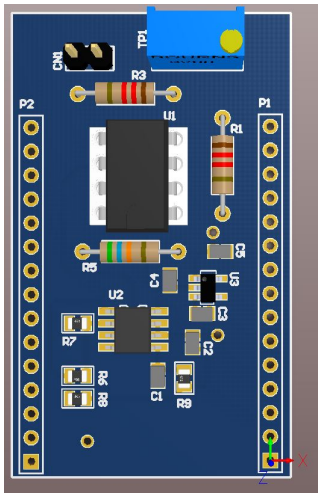
Para confecção da placa de circuito impresso (PCI), decidiu-se por utilizar o máximo possível de componentes *surface-mount device* (SMD), pois estes componentes normalmente são menores que os *Pin-through-hole* (PTH). Desta maneira, é possível reduzir consideravelmente o tamanho da PCI. Nesta placa, tem-se o circuito de condicionamento do sinal conforme discutido previamente. Observa-se o esquemático (FIGURA 3.9) que há dois conectores "P1"e "P2"conectados ao circuito, assim, estes servirão de *shield* para o ESP32 *NodeMCU*, O qual ficará disposto acima dos demais componentes na PCI. Desta forma, observa-se na representação 3D (FIGURA 3.10 que a PCI terá um tamanho reduzido, sendo um dos objetivos para a confecção desta placa para que seja possível acoplar no cinto juntamente com o transdutor elástico.

FIGURA 3.9 – ESQUEMÁTICO DA PCI



Fonte: O autor (2018)

FIGURA 3.10 – CAMADA SUPERIOR DA PCI EM 3D



Fonte: O autor (2018)

3.1.4 Programa do módulo ESP32

Para a integração do circuito de aquisição da atividade respiratória com o módulo ESP32, tem-se a necessidade do desenvolvimento de um *firmware* que realizará a leitura do conversor analógico digital e enviará os dados para um dispositivo externo (microcomputador) através da comunicação sem-fio *Wi-Fi*. O programa será desenvolvido na linguagem de programação Micropython, sendo esta uma implementação enxuta e eficiente de Python 3, que contém um subconjunto de bibliotecas padrão, otimizada para ser executada em microcontroladores e em ambientes restritos (MICROPYTHON, 2017). Assim, dado estas considerações, o desenvolvimento deste *firmware* deverá visar o cumprimento dos seguintes requisitos gerais:

- amostrar o sinal do circuito de tratamento em uma taxa adequada;
- enviar as informações do conversor analógico digital para o microcomputador;
- realizar uma interface de conexão com o dispositivo externo (computador).

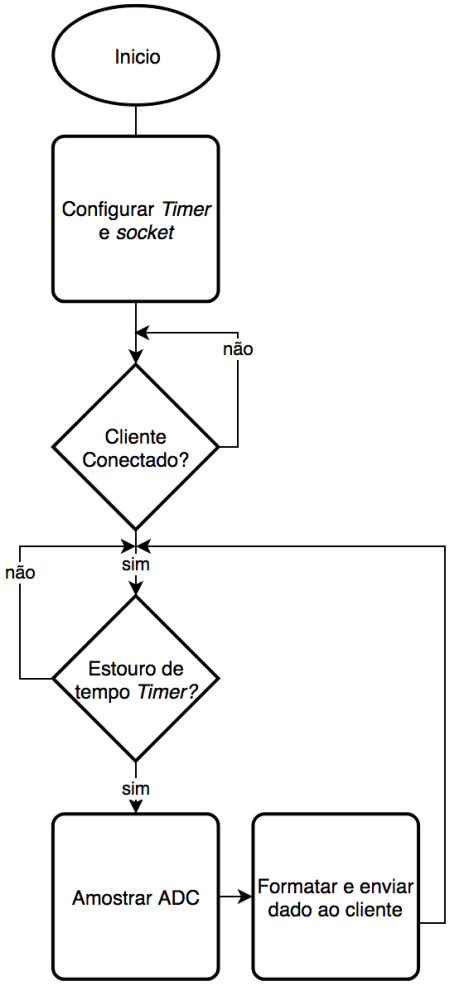
Dado o cumprimento dos requisitos, a funcionalidade da amostragem do sinal do circuito de aquisição foi desenvolvida através da utilização da estrutura de um *timer* do microcontrolador presente no módulo. Esta funcionalidade do programa consiste no monitoramento de uma variável que é modificada em toda ocorrência de interrupção do *timer*. O *timer* é configurado com um determinado período de contagem e com uma função de *callback* que é executada toda vez que ocorre o estouro de tempo. Tal função modifica uma variável global ao *firmware*, que então é verificada em um laço principal

que dispara a amostragem do sinal.

Dado o filtro de *anti-liasing*, pode-se realizar a amostragem do sinal com frequências maiores ou iguais a $2fc$, ou seja, a mínima frequência a ser utilizada é aproximadamente $22\ Hz$, neste sentido, realiza-se a amostragem do sinal a cada $20\ ms$ ($50\ Hz$), possibilitando maior flexibilidade no processamento que será realizado pelo *software* desenvolvido para o microcomputador.

Por fim, os dados do conversor analógico digital são formatados em um tamanho de *string* fixo e são enviados ao microcomputador. O envio dos dados ocorrerá imediatamente após a amostragem do sinal através de *sockets* TCP, onde o módulo é configurado em como servidor e espera até que o *software* do dispositivo externo se conecte como cliente para então iniciar o envio das informações da atividade respiratória. No fluxograma da FIGURA 3.11 ilustra-se esta funcionalidade e no APÊNDICE A apresenta-se o código completo.

FIGURA 3.11 – FLUXOGRAMA AMOSTRAGEM E ENVIO DE DADOS



Fonte: O autor (2018)

Utilizando-se a biblioteca denominada *network* para configuração do ESP32 e em conjunto com *sockets* UDP e TCP, desenvolveu-se um sistema de gerenciamento de conexão do módulo com dispositivos externos. Neste sistema, tem-se a possibilidade tanto a listagem e escolha de conexão do ESP32 a redes locais, bem como conectar-se a ele diretamente em modo *AP* (*Access Point*).

Inicialmente quando o módulo ESP32 é energizado, dado a implementação do *firmware* Micropython, um *script* denominado *main.py* é executado. Baseando-se nesta estrutura, dentro deste *script* realiza-se a verificação de configuração do módulo. Tal como proposto por Tayfu (2017), caso o módulo não estiver configurado, este torna-se um ponto de acesso e inicia-se um servidor *Web* em um IP fixo, que enviará uma página de configuração *html* (*HyperText Markup Language*) que lista a escolha de pontos de acesso disponíveis e também um campo de inserção de senha (FIGURA 3.12), realizando dentre outras funcionalidades, o tratamento de exceções e erros da inserção de uma senha incorreta. Ao inserir a senha no campo, o módulo grava as informações em um arquivo "*.dat*", que é consultado toda vez que o módulo é energizado, realizando-se a conexão com a rede caso esta já tenha sido utilizada e esteja disponível.

FIGURA 3.12 – PÁGINA CONFIGURAÇÃO ESP32

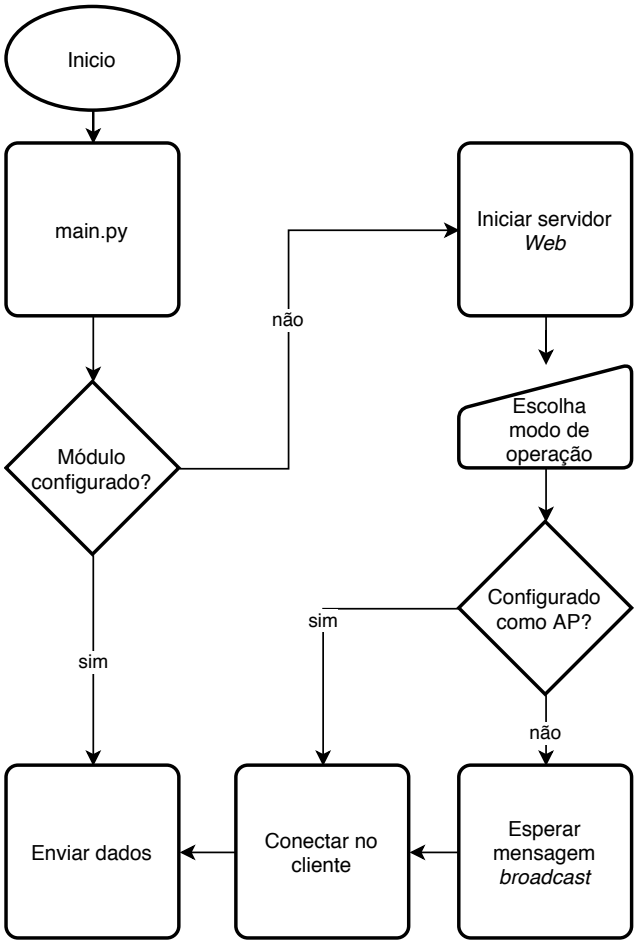
Fonte: O autor (2018)

Dado a escolha de conexão do módulo em uma rede local, é funcional que este encontre o IP do microcomputador ao qual irá enviar os dados de forma automática, não necessitando de configurações adicionais. Assim, realizou-se esta funcionalidade através do recebimento de uma mensagem *broadcast* através do protocolo UDP. A mensagem deverá ser enviada automaticamente pelo computador que o módulo deverá se comunicar. Ao utilizar o endereço *broadcast*, não há necessidade que o módulo e o

dispositivo saibam as configurações de IP um do outro para se comunicar. Ou seja, a mensagem enviada por uma aplicação cliente ao endereço *broadcast* é enviada a todos os dispositivos conectados na rede utilizada, através de uma técnica de roteamento implementada na camada de rede (KUROSE, 2013). Ao receber a mensagem *broadcast* é possível descobrir o endereço do computador e então iniciar o envio de dados tal como ilustrado na FIGURA 3.11.

Uma outra opção é o módulo continuar como *AP* e fazer a comunicação diretamente com o microcomputador, onde esta opção pode ser feita diretamente na página do servidor *Web* através de um botão. Na FIGURA 3.13 tem-se o fluxograma que apresenta a funcionalmente o fluxo das principais operações completa da configuração do módulo. Os códigos completos de configuração encontram-se em no APÊNDICE B e no APÊNDICE C.

FIGURA 3.13 – FLUXOGRAMA DA CONFIGURAÇÃO ESP32



Fonte: O autor (2018)

3.1.5 Projeto Filtro Digital

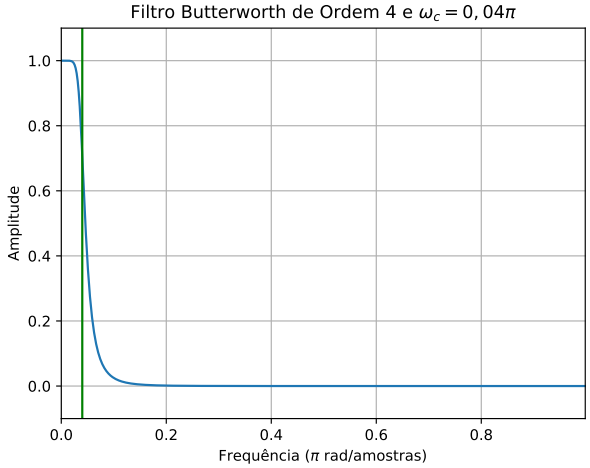
O *software* desenvolvido para o microcomputador filtrará os sinais rebebidos do módulo ESP32 utilizando um filtro digital que foi projetado dado as características do sinal respiratório. Considerando-se uma taxa respiratória máxima de 60 respirações por minuto, chega-se a $1\ Hz$. Neste sentido, o filtro digital projetado terá por objetivo atuar de forma equivalente na filtragem de sinais com espectro acima desta frequência.

Desconsiderando-se possíveis atrasos de transmissão através do *Wi-Fi* e também levando-se em consideração que o sinal é de baixíssima frequência, será utilizado $20\ ms$ como período de amostragem para o projeto do filtro. Neste sentido pode-se definir a frequência de corte discreta, em $rad/amostra$:

$$\omega_c = \Omega_c \cdot T = 2 \cdot \pi \cdot 1 \cdot 20 \cdot 10^{-3} = 0,04\pi\ rad/amostra \tag{3.4}$$

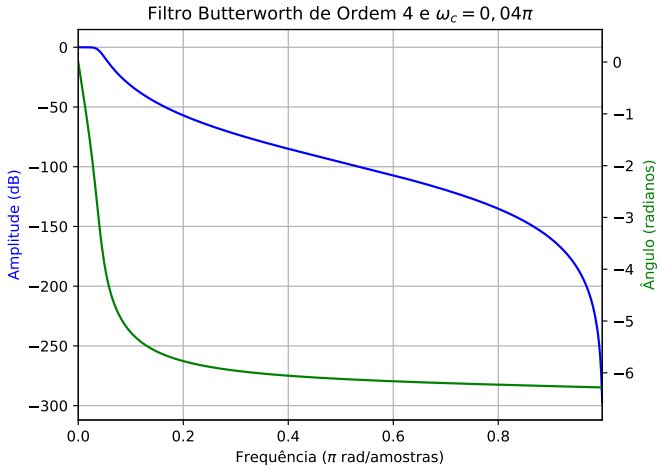
Dado a frequência de corte definida, escolheu-se pela utilização de um filtro do tipo IIR de resposta *Butterworth* devido a sua resposta sem oscilações tanto na faixa de passagem como na de rejeição, tal como evidenciado na subseção 2.4.5. Projetou-se um filtro de quarta ordem através do módulo *signal* disponível na biblioteca *scipy* da linguagem de programação Python. Através deste módulo obteve-se os coeficientes do filtro e também sua resposta em frequência. Na FIGURA 3.14, tem-se a resposta em frequência obtida, observa-se a frequência de corte em $0,04\ \pi$. Na FIGURA 3.15, apresenta-se também a resposta em frequência em dB, além de apresentar a resposta em fase do filtro.

FIGURA 3.14 – RESPOSTA EM FREQUÊNCIA DO FILTRO PROJETADO (ESCALA LINEAR)



Fonte: O autor (2018)

FIGURA 3.15 – RESPOSTA EM FREQUÊNCIA DO FILTRO PROJETADO (AMPLITUDE E FASE)



Fonte: O autor (2018)

3.1.6 Desenvolvimento do *software* microcomputador

O *software* para o dispositivo externo (microcomputador), tem por objetivo receber o sinais enviados pelo ESP32, então processá-los e disponibilizá-los como uma forma de *biofeedback* visual da respiração. Este *software* foi desenvolvido através da linguagem de programação Python 3, onde utilizou-se módulos padrão da linguagem, módulos de análise de dados e de processamento de sinais e também de desenvolvimento de interface gráfica para o usuário (*GUI - Graphical User Interface*).

Estabeleceu-se os seguintes requisitos gerais do *software* desenvolvido nesta etapa:

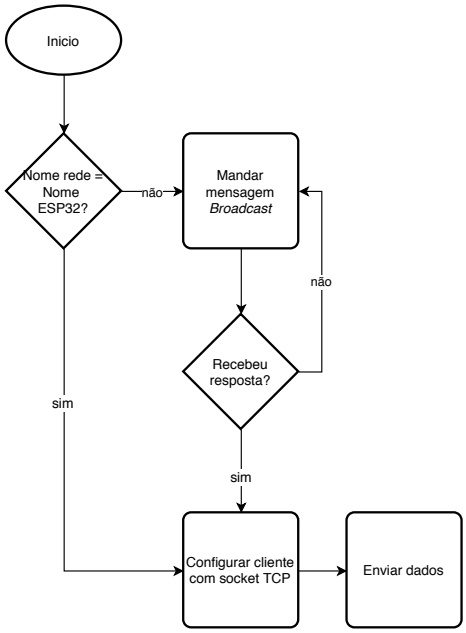
- interfacear a conexão do módulo com o microcomputador, recebendo-se os dados da atividade respiratória;
- processar os dados recebidos, filtrando-os e estimando a taxa respiratória;
- retornar as informações através de uma interface gráfica, realizando um *biofeedback* do sinal da placa de aquisições.

Como apresentado na subseção 3.1.4, o módulo ESP32 poderá ser conectado tanto através de uma rede local, como diretamente por meio da sua configuração como *AP*. Neste sentido, necessitou-se que o *software* desenvolvido, primeiramente realizasse a checagem de qual a configuração o módulo está, antes de iniciar o recebimento dos dados. Para checagem automática da conexão do módulo como modo

AP, faze-se uma chamada no sistema para que se obtenha o nome da conexão *Wi-Fi* do microcomputador, caso o nome seja aquele configurado no módulo como *AP*, então pode-se iniciar o recebimento de dados como cliente por meio de um IP fixo.

Se o módulo estiver conectado em uma rede local, então necessita-se que este envie mensagens *broadcast* através do protocolo UDP para a rede. Desta forma é possível encontrar o número IP do ESP32 de forma automática dado as configurações feitas no módulo. Recebendo uma resposta do módulo, encontra-se o seu endereço e pode-se iniciar a comunicação através de *sockets* TCP, onde o microcomputador torna-se um cliente que irá receber os dados do servidor implementado no ESP32. O código de configuração e recebimento de dados encontra-se no APENDICE D. Na FIGURA 3.16, tem-se o fluxograma desta funcionalidade do *software*.

FIGURA 3.16 – FLUXOGRAMA CONFIGURAÇÃO DE RECEBIMENTO DE DADOS



Fonte: O autor (2018)

No contexto do processamento de dados, utilizou-se o filtro projetado na subseção 3.1.5. Sendo uma etapa de pré-processamento que elimina possíveis sinais que não possuem a informação da atividade respiratória. O filtro digital foi implementado através de sua equação a diferenças. Sendo um filtro de quarta ordem, do tipo IIR, este filtro possuirá o formato da equação 3.5, onde a_k e b_k são os coeficientes calculados para a frequência especificada.

$$y[n] = \sum_{k=0}^{4} b_k x[n-k] - \sum_{k=1}^{4} a_k y[n-k] \tag{3.5}$$

A implementação da equação do filtro consistiu no desenvolvimento de um algoritmo que realiza as somas das amostras anteriores e a atuais do filtro dado a multiplicação dos coeficientes b_k. Da mesma maneira soma-se à saída as saídas anteriores dado a multiplicação dos coeficientes a_k. Na FIGURA 3.17, apresenta-se o pseudocódigo de implementação do filtro. Observa-se a utilização de dois elementos de *buffer* que armazenam as entradas e as saídas anteriores do sinal, onde os índices dos vetores são deslocados a cada nova amostra para que sempre os últimos quatro valores de entrada e saída sejam armazenados.

FIGURA 3.17 – PSEUDOCÓDIGO FILTRO DIGITAL

Algoritmo Filtro Digital

```
ak ← [a1, a2, a3, a4]
bk ← [b0, b1, b2, b3, b4]
na ← length(ak)
nb ← length(bk)
bufa ← zeros(nb)
bufb ← zeros(na)
p ← 0
v ← 0
g ← 0
sinalFiltrado ← 0
while True do
    if p > nb − 1 then
        p ← 0
    end if
    bufb(p) ← sinalLido
    sinalFiltrado ← 0
    k ← p
    p ← p + 1
    for j ← 0 to nb do
        sinalFiltrado ← sinalFiltrado + b(j) ∗ bufb(k)
        k ← k − 1
        if k < 0 then
            k ← nb − 1
        end if
    end for
    for j ← 0 to nb do
        sinalFiltrado ← sinalFiltrado − a(j) ∗ bufa(g)
        g ← g − 1r
        if g < 0 then
            g ← na − 1
        end if
    end for
    bufa[v] ← sinalFiltrado
    g ← v
    v ← v + 1
end while
```

Fonte: O autor (2018)

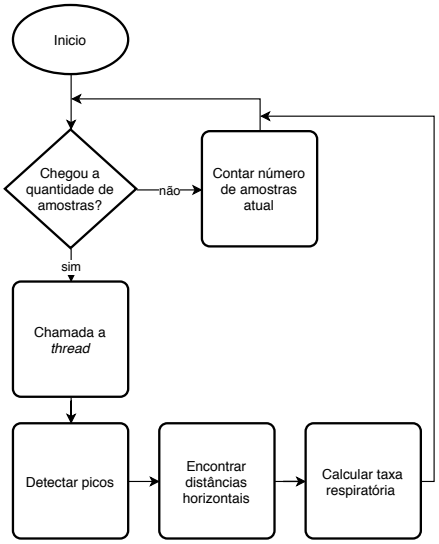
Após o sinal filtrado, é aplicado um algoritmo de calculo de taxa respiratória. Este calculo consiste inicialmente na aplicação da detecção de picos do sinal através

do algoritmo proposto por Negri (2018). Este algoritmo é codificado em um módulo em Python, que possui um método que encontra o índice numérico dos picos de um sinal temporal discreto, utilizando diferenças de primeira ordem entre seus elementos (NEGRI, 2018). Como parâmetros deste método, tem-se a distância mínima entre amostras para encontrar um pico e também o mínimo valor de amplitude que é considerado como a ocorrência de um pico.

No *software* desenvolvido, o cálculo de taxa respiratória é realizado a cada ocorrência de um número fixo de amostras. Assim, durante a execução do *software*, quando chega-se ao número de amostras estabelecido, encontra-se o índice da ocorrência de picos e então calcula-se a diferença consecutiva entre os elementos horizontais (amostra de ocorrência), tal como descrito por Jeyhani et al. (2017). Multiplica-se este vetor pelo período de amostragem ($20\ ms$) e realiza-se a média. O inverso do valor obtido multiplicado por 60 é então considerada a taxa respiratória estimada. Para ganho de desempenho, executa-se estes cálculos através da chamada de uma *thread* que é executada separadamente do programa principal. Na equação 3.6, apresenta-se o método de calculo. Na FIGURA 3.18, tem-se o fluxograma que apresenta a o fluxo de cálculos relacionados a estimação da taxa respiratória.

$$taxa = \frac{60}{media(distancias\ horizontais \cdot T)} \left(\frac{respirações}{min}\right) \tag{3.6}$$

FIGURA 3.18 – FLUXOGRAMA CONFIGURAÇÃO DE RECEBIMENTO DE DADOS



Fonte: O autor (2018)

Dentro do contexto de interface do sistema, utilizou-se a *framework PyQt4* para o desenvolvimento da interface gráfica do *software* que irá retornar as informações do *biofeedback* respiratório.

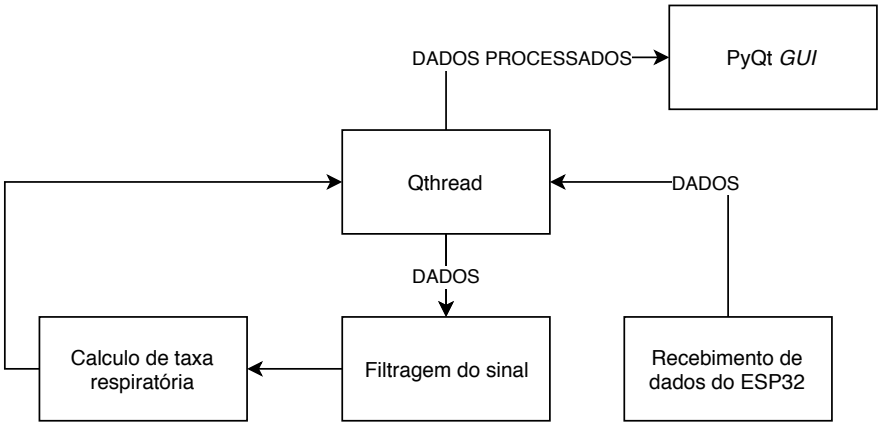
O desenvolvimento do *layout* do *GUI (Graphic User Interface)* foi feito na ferramenta *Qt Designer*, colocando-se um gráfico para acompanhamento da forma de onda em tempo real, uma barra controle da respiração, um *display* que retorna a taxa respiratória estimada e um *display* para acompanhamento do tempo de treinamento (FIGURA 3.19). Este *Layout* é exportado para uma Classe em Python, onde é possível utilizá-la para acessar os elementos gráficos e executar o *software* de interface gráfica.

FIGURA 3.19 – *LAYOUT* DA INTERFACE DO SISTEMA

Fonte: O autor (2018)

Dado o desenvolvimento da aparência do *software* do protótipo, realizou-se implementação da lógica do sistema. As estruturas de processamento são utilizadas para fornecer os dados ais elementos interativos do sistema, tal como o gráfico e a barra de treinamento. O programa principal é estruturado em dois elementos de código, o primeiro é responsável pela execução de uma interface gráfica e o segundo é responsável pelo processamento e chamada de *threads* adicionais. No segundo elemento, utilizou-se o objeto *Qthread* que permite o envio de dados para a interface gráfica através de sinais de compartilhamento que são diretamente realizados através da *framework PyQt4* no mesmo código. Na FIGURA, 3.20 evidencia-se a estrutura do fluxos dos dados no *GUI* desenvolvido. No APÊNDICE E, tem-se o código principal do *software*

FIGURA 3.20



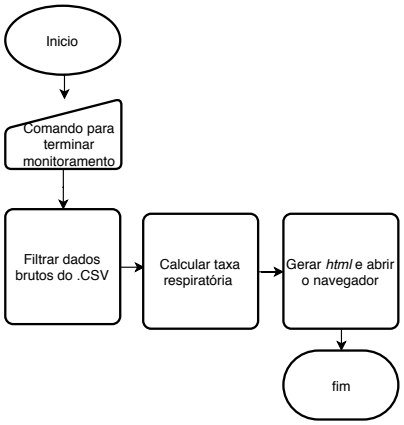
Fonte: O autor (2018)

Além do *software* de acompanhamento tempo real da atividade respiratória, desenvolveu-se a funcionalidade da geração de um relatório de treinamento após o termino da utilização do sistema. O relatório é gerado no formato *HTML* e é produzido no encerramento da execução do *software* de interface gráfica que já foi descrito. Após aberto, o relatório dispõe de uma tabela interativa que possui os valores de taxa respiratória em intervalo fixo de amostras, definido como o período de análise. Também no relatório são apresentados dois gráficos interativos, sendo o primeiro da forma de onda da atividade respiratória e o segundo um gráfico que apresenta as taxas de respiração calculadas no período de tempo de análise.

Para geração deste relatório, utilizou-se a biblioteca *Plotly*, que possui dentre outras funcionalidades a geração de gráficos interativos dos mais variados tipos com interação *online* e *offline* (PLOTLY, 2018). Os dados sem nenhum processamento, são gravados em um arquivo *CSV* (*Comma-separated values*) durante o monitoramento da atividade respiratória. Ao fechar-se a janela da interface gráfica, filtra-se os dados do arquivo como um todo e então calcula-se as taxas respiratórias em um intervalo de amostras específico (1200 amostras).

Terminado o processamento *offline*, o arquivo *HTML* é aberto em um navegador do sistema operacional automaticamente com o relatório. No APÊNDICE F, tem-se o código em Python desenvolvido para realizar esta funcionalidade, este é importado como um módulo no código principal (APÊNDICE E) e chamado através de uma função, assim que a interface é fechada. Na FIGURA 3.21, apresenta-se um fluxograma do funcionamento da geração do relatório da atividade respiratório.

FIGURA 3.21 – DIAGRAMA DE BLOCOS DO SISTEMA

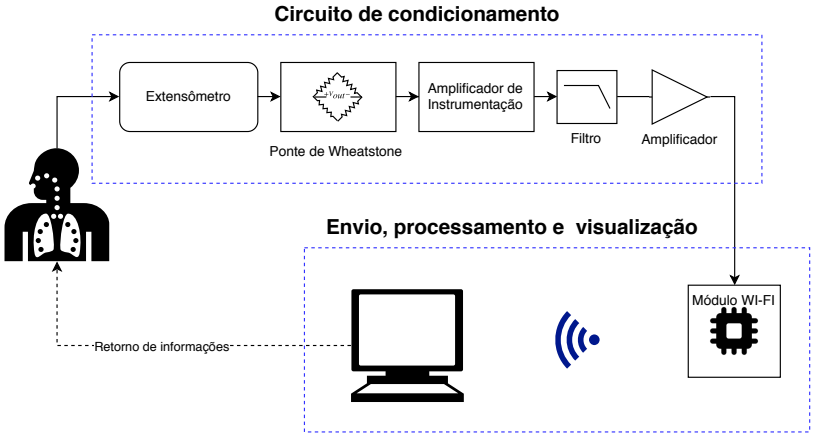


Fonte: Autor

3.1.7 Diagrama do sistema

O sinal que quantizará a respiração será proveniente de um processo de condicionamento de sinais analógicos. As deformações relativas ao abdômen serão utilizadas como parâmetros de entrada, de tal modo que serão desenvolvidos e acoplados a partes funcionais do sistema que tornarão possível quantizar a atividade respiratória através de grandezas elétricas. Após o condicionamento de sinal, este será digitalizado através de um conversor analógico digital e enviado para um dispositivo externo (microcomputador) por meio da comunicação sem-fio *Wi-Fi* (realizada pelo ESP32). No dispositivo externo o sinal é processado, retornando as informações da atividade respiratória como *biofeedback*. Levando-se em consideração as etapas que desenvolvidas, na FIGURA 3.22, tem-se o diagrama de blocos que compõe o sistema, observa-se a separação deste em blocos funcionais.

FIGURA 3.22 – DIAGRAMA DE BLOCOS DO SISTEMA



Fonte: Autor

3.1.8 Custos do protótipo

Para a implementação do protótipo, foi necessário a compra de componentes eletrônicos. Dentre eles, o transdutor elástico e o TPS6040 foram comprados no exterior, devido ao fato de não serem comercializados no Brasil. Desta forma, a TABELA 1 representa a estimativa de valores dos componentes adquiridos ao longo do projeto. Esta estimativa, levou-se em considerações apenas os valores dos componentes, desconsiderando os valores de frete e importação.

TABELA 1 – CUSTOS DO PROTÓTIPO

Componentes	VALOR [R$]
Transdutor Elástico	50
INA128	32
Resistores	3
Capacitores	4
ESP32	75
Conectores	5
LM6142	18
TPS6040	5
Placa Fenolite	4
TOTAL	196

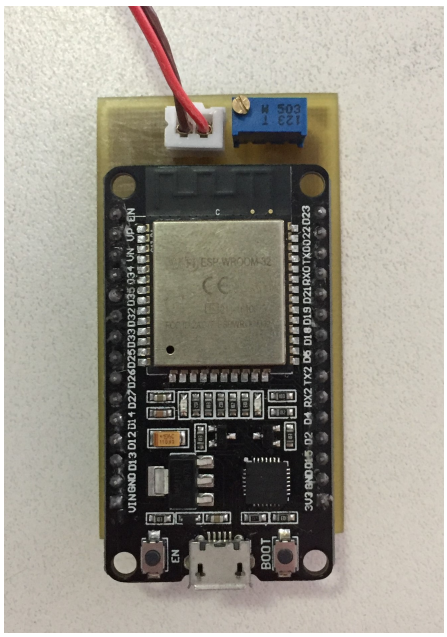
4 RESULTADOS

Neste capitulo, serão apresentados os resultados e discussões relacionados ao desenvolvimento do protótipo deste trabalho. Serão apresentados a montagem do circuito de aquisição, os testes do circuito e também o funcionamento do sistema como um todo.

4.1 PLACA DE AQUISIÇÃO

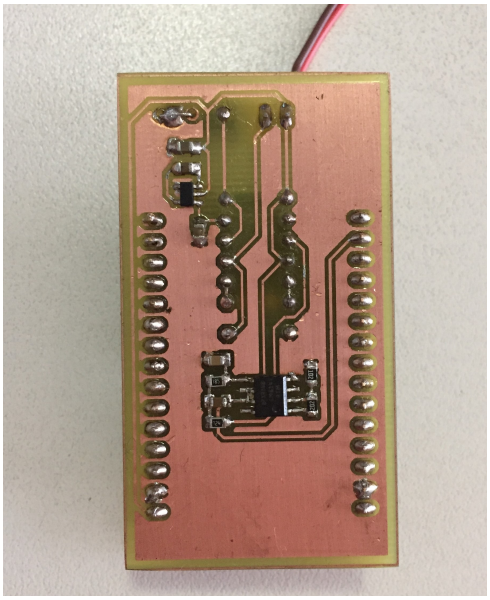
A partir do esquemático e do *layout* definido na subseção 3.1.3, fora possível confeccionar a PCI (FIGURA 4.2). Nesta observa-se que a placa de aquisição tem tamanho compatível para integrar ao cinto, juntamente com o transdutor elástico e a bateria de alimentação, com isto, a PCI tem a dimensão de aproximadamente 30x62 mm. Observa-se os conectores laterais que servem de *shield* para o ESP32, com isto a vista superior aparecerá apenas o ESP32 (FIGURA 4.1). Desta forma a PCI já vem preparada para ser integrado com a placa de desenvolvimento, bastando apenas conectá-los junto aos pinos, vista inferior (FIGURA 4.2).

FIGURA 4.1 – PLACA DE CIRCUITO IMPRESSO VISTA SUPERIOR



5 Fonte: Autor (2018)

FIGURA 4.2 – PLACA DE CIRCUITO IMPRESSO VISTA INFERIOR



Fonte: Autor (2018)

4.2 TESTES DE CIRCUITO

Como primeiro teste do desenvolvimento do sistema, testou-se a ponte de Wheatstone com dois resistores de 680 Ω e um *trimpot* para ter uma maior facilidade na calibragem da ponte. Assim, quando o transdutor estiver em repouso, ajusta-se a tensão diferencial aproximadamente para zero. Alimentou-se o circuito com uma tensão de 3,3 V, obtendo-se a tensão diferencial de saída do circuito da ponte. Realizou-se testes, esticando o condutor elástico para verificar a variação da tensão de saída da ponte de Wheatstone. Desta forma, pôde-se analisar qualitativamente os diferentes níveis de tensões dado níveis de alongamento do transdutor. Na TABELA 2, tem-se as tensões medidas na saída do circuito com o transdutor elástico. Observou-se que a variação de tensão foi substancial, podendo-se utilizar ganhos relativamente pequenos para a amplificação do sinal.

TABELA 2 – TENSÃO SAÍDA DA PONTE DE WHEATSTONE

Alongamento (condutor elástico)	Saída Ponte W. (mV)
Mínimo	180 mV
Médio	320 mV
Máximo	500 mV

Fonte: O autor (2018)

Da mesma maneira que a tensão da saída da ponte de Wheatstone, analisou-

se a tensão de saída do amplificador de instrumentação INA128 nas mesmas condições da TABELA 2. Obteve-se a TABELA 3, nesta, evidencia-se a saída do amplificador de instrumentação dado o ganho que foi ajustado.

Após analisar as saídas da ponte de Wheatstone e também do amplificador de instrumentação, ajustou-se o ganho do amplificador não inversor em $1,5$, obtendo-se uma tensão de máxima no circuito de aproximadamente $1,42\ V$.

TABELA 3 – TENSÃO DE SAÍDA INA128

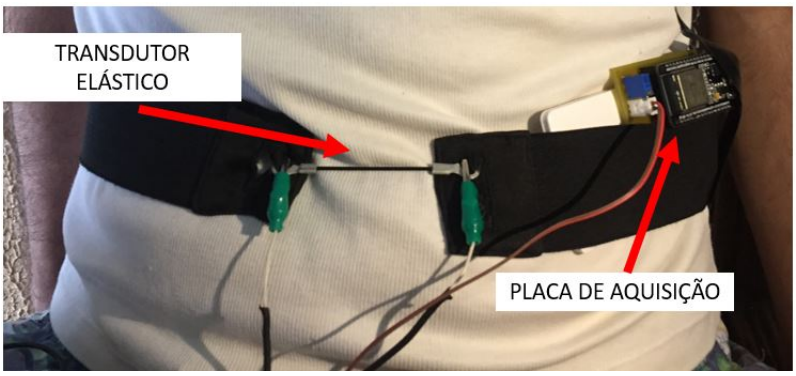
Alongamento (condutor elástico)	Saída Ponte INA 128 (mV)	Razão entre entrada e saída (V/V)
Mínimo	330 mV	1,84
Médio	609 mV	1,90
Máximo	938 mV	1,88

Fonte: O autor (2018)

4.3 TESTES DE FUNCIONAMENTO

A partir da placa de aquisição, realizou-se testes no sistema completo. Desta forma, integrou-se o condutor elástico a uma cinta para realização do monitoramento da atividade respiratória em um dos integrantes da equipe. Para os testes realizados, a cinta foi posicionada no abdômen (respiração abdominal), monitorando-se os sinais de saída do circuito durante a atividade respiratória do participante do teste.

FIGURA 4.3 – POSICIONAMENTO DA CINTA COM O SISTEMA DE AQUISIÇÃO



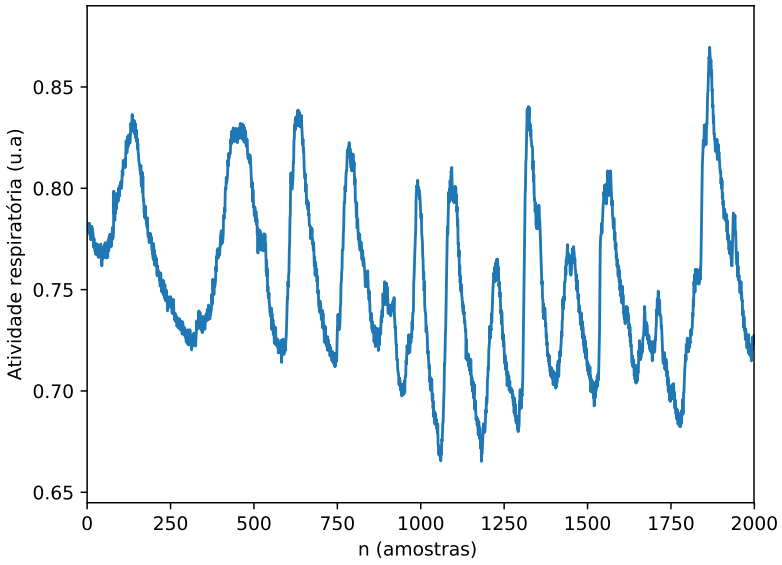
Fonte: Autor (2018)

Para testar o sistema, realizou-se a aquisição e envio do sinal durante a atividade respiratória. Os sinais foram enviados através do ESP32 e recebidos pelo *software* do microcomputador. Os dados recebidos serão analisados, evidenciando-se as caraterísticas do sinal que foi processado e que é utilizado como *biofeedback* no

sistema. Serão analisados os dados recebidos de forma bruta (sem processamento), os dados após a filtragem e também o resultado da detecção de taxas respiratórias a partir dos dados processados.

Neste sentido, como primeira análise, pode-se observar a FIGURA 4.4, nesta tem-se a atividade respiratória monitorada durante o intervalo de tempo de $40\ s$ (2000 amostras). Observa-se o sinal recebido sem a realização da filtragem digital, torna-se evidente a necessidade da filtragem após a amostragem da saída do circuito de aquisição. Como o sinal recebido possui o valor inteiro que vai de 0 a 4095 (conversor analógico digital de 12 bits), normalizou-se o sinal por 4095 para melhor visualização, apresentando-o no gráfico como um sinal adimensional.

FIGURA 4.4 – SINAL RESPIRATÓRIA RECEBIDO (DADOS BRUTOS)



Fonte: O autor (2018)

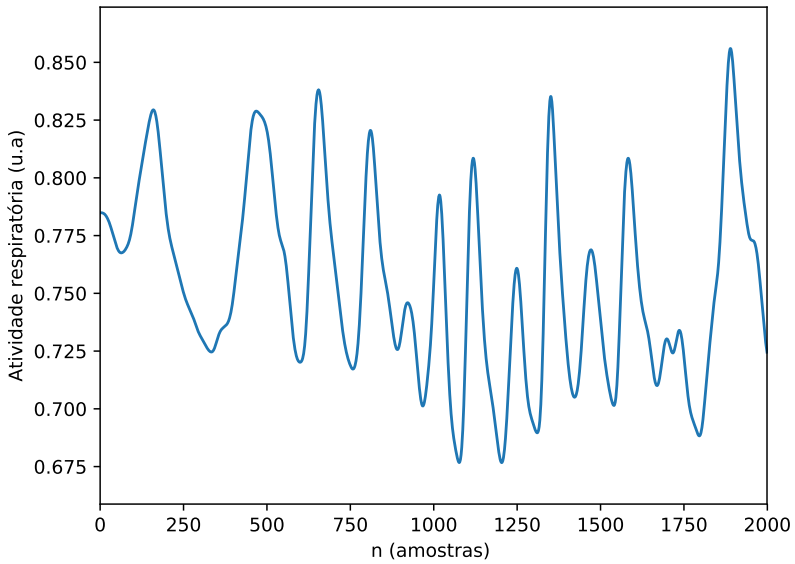
Após o recebimento dos dados brutos, estes são filtrados através do filtro passa baixa que foi projetado para filtrar frequências que não estão relacionadas a atividade respiratória. A atividade respiratória varia em torno de 4 a 60 respirações por minuto, neste sentido aplicou-se o filtro projetado na subseção 3.1.5.

Observa-se na FIGURA 4.5 o sinal filtrado, evidencia-se a suavização da curva da FIGURA 4.4, preprocessando o sinal para utilização na interface gráfica e também para o cálculo da taxa respiratória.

Na FIGURA 4.6 e na FIGURA 4.6, tem-se a *FTT* do sinal filtrado. A análise

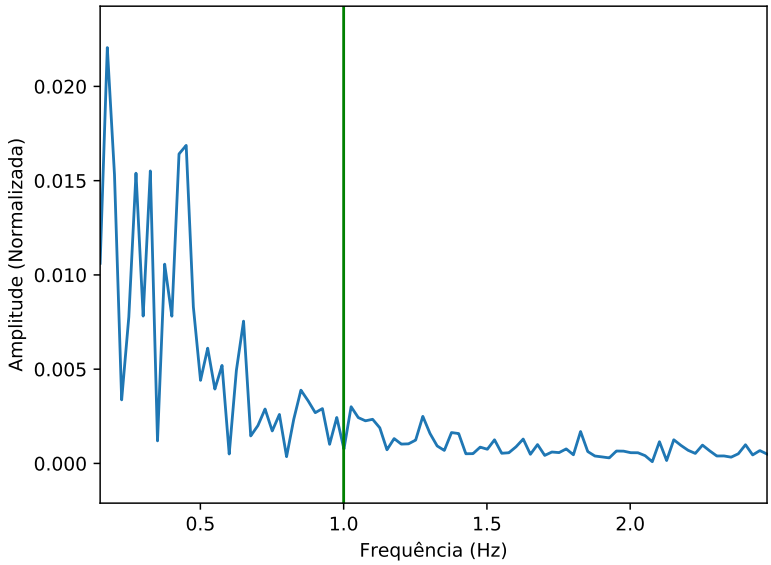
em frequência foi realizada através do módulo denominado *scipy.fftpack* do Python 3. Como esperado, observa-se que as amplitudes de frequência que estão acima de 1 Hz, são atenuadas com a aplicação do filtro projetado.

FIGURA 4.5 – SINAL RESPIRATÓRIO RECEBIDO (DADOS FILTRADOS)



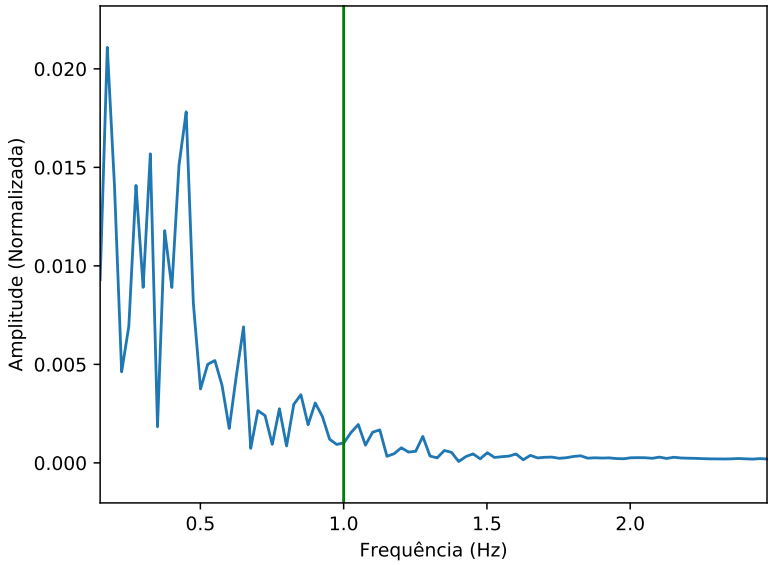
Fonte: O autor (2018)

FIGURA 4.6 – FFT SINAL RESPIRATÓRIO RECEBIDO (DADOS FILTRADOS)



Fonte: O autor (2018)

FIGURA 4.7 – FFT SINAL RESPIRATÓRIO RECEBIDO (DADOS FILTRADOS)



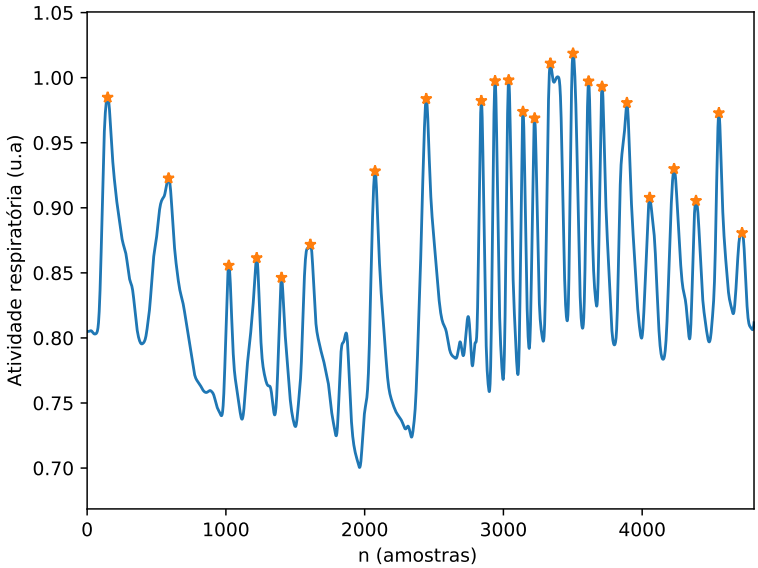
Fonte: O autor (2018)

Para evidenciar o calculo da taxa respiratória através do sistema de aquisição, foram analisadas 4800 amostras recebidas ($96\ s$), obtendo-se os valores de taxas respiratórias a cada 1200 amostras ($24\ s$). Na FIGURA 4.8, tem-se o gráfico das amostras analisadas. O detector de picos implementado por Negri (2018), tem como limiar de amplitude, a diferença entre o máximo e o mínimo valor das amostras, multiplicado por um valor de porcentagem definida na chamada do detector e subtraído novamente pelo valor mínimo, tal como definido na equação 4.1.

$$limiar = porcentagem \cdot (max(amostras) + min(amostras)) - min(amostras) \quad (4.1)$$

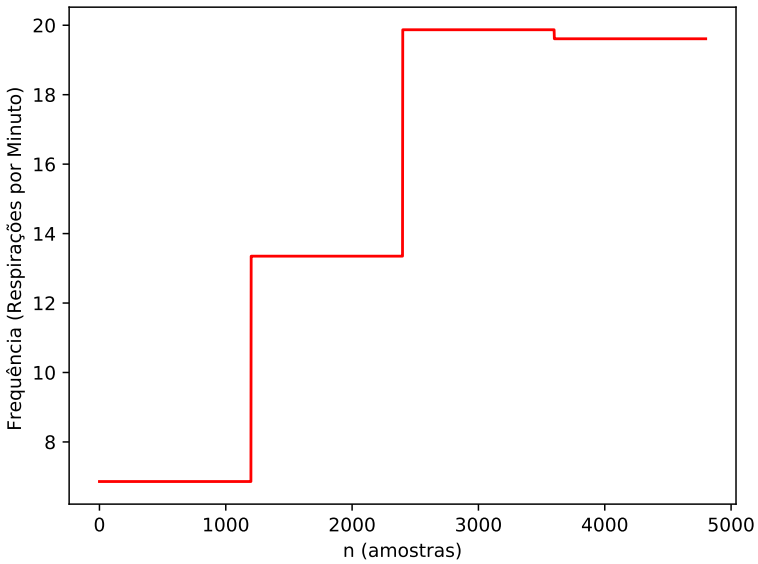
Na porcentagem da chamada da função, utilizou-se $0,4$, tendo-se a distância mínima de 60 amostras entre picos. Além disso, experimentalmente, obteve-se um resultado mais coerente, eliminando-se valores de picos encontradas que tiveram valores menores que 40% da média entre o máximo e o mínimo pico encontrado no detector. Na FIGURA 4.9, apresenta-se o gráfico das taxas respiratórias encontradas. A sequencia de taxas encontrada foram $6,87$, $14,07$, $19,85$ e 18.99 (em respirações por minuto). Dado as características da FIGURA 4.7, observa-se que a sequencia apresenta-se coerente pois acompanha o aumento e diminuição da distância temporal entre picos.

FIGURA 4.8 – SINAL RESPIRATÓRIO COM DETECÇÃO DE PICOS



Fonte: Autor (2018)

FIGURA 4.9 – TAXA RESPIRATÓRIA EM RELAÇÃO AO NUMERO DE AMOSTRAS ANALISADAS



Fonte: O autor (2018)

Finalmente, com o objetivo de evidenciar o funcionamento do protótipo desenvolvido, observa-se na FIGURA 4.10 o sistema de *biofeedback* durante a sua utilização, tem-se os elementos interativos, onde o gráfico em tempo real é retornado, a taxa

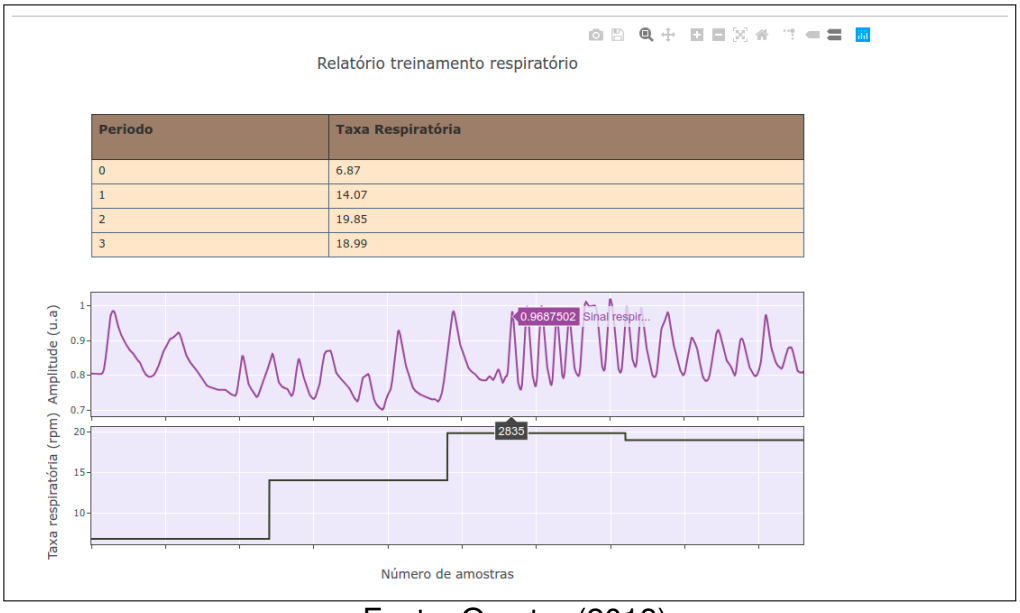
respiratória e uma barra proporcional que movimenta-se proporcionalmente a amplitude do sinal recebido.

FIGURA 4.10 – INTERFACE GRÁFICA DURANTE O TREINAMENTO

Fonte: O autor (2018)

Além da interface gráfica desenvolvida, empregou-se a funcionalidade do relatório de atividade respiratória ao fim da utilização do *software*, na FIGURA 4.11 tem-se um exemplo de relatório gerado, observa-se a presença de uma tabela a informação da taxa respiratória e dois gráficos, um com a forma de e um com a evolução temporal da taxa respiratória.

FIGURA 4.11 – RELATÓRIO ATIVIDADE RESPIRATÓRIA



Fonte: O autor (2018)

5 CRONOGRAMA

O cronograma utilizado para a realização do trabalho de conclusão do curso pode ser observado na TABELA 4.

TABELA 4 – CRONOGRAMA DO DESENCVOLVIMENTO DO TRABALHO

ATIVIDADES	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
Definição do tema/Problema	x											
Definição dos objetivos	x											
Pesquisa Bibliográfica	x	x	x									
Elaboração da metodologia		x	x									
Reconhecer as características do biofeedback		x	x									
Redação do projeto			x	x								
Desenvolvimento do protótipo				x	x	x	x	x				
Teste do protótipo						x	x	x	x	x		
Análise dos resultados				x	x	x	x	x	x	x	x	
Elaboração do software							x	x	x	x		
Integração do software e hardware									x	x	x	
Redação do TCC					x	x	x	x	x	x	x	x
Apresentação do TCC						x						x

6 CONCLUSÕES

Neste trabalho, conseguiu-se abordar os principais aspectos do conceito de *biofeedback*, enfatizando o seu uso como ferramenta para pacientes que sofrem de crise de ansiedade, transtornos e entre outros distúrbios.

Através do desenvolvimento de um sistema eletrônico e computacional, teve-se por objetivo o monitoramento da atividade respiratória, visando que estas informações possam ser utilizadas como uma resposta de *biofeedback* aos usuários do sistema, que assim, em conjunto com profissionais da área de medicina e psicologia, possam obter a habilidade de autocorreção fisiológica.

Dado o objetivo do desenvolvimento de um dispositivo de monitoramento da atividade respiratória, foram estudados os aspectos necessários para quantizar e qualificar este tipo de atividade fisiológica do corpo, colocando-se em pauta a utilização de um condutor elástico como transdutor para o monitoramento dos movimentos respiratórios do abdômen. Desta forma, foram estudados importantes aspectos da aquisição dos sinais provenientes deste tipo de transdutor resistivo, possibilitando a escolha da abordagem mais adequada por meio do estado da arte.

Com transdutor escolhido, realizou-se o levantamento qualitativo de suas características em conjunto com uma ponte de Wheatstone. A partir disso, tornou-se evidente quais os aspectos e valores de amplificação necessários para condicionar o sinal proveniente do alongamento e contração do transdutor durante a atividade respiratória. A partir destes testes, estabeleceu-se os blocos funcionais do circuito de aquisição. O circuito obtido mostrou uma excelente responsividade em relação ao monitoramento da atividade respiratória através da movimentação abdominal, tonando-se possível realizar as etapas de transmissão, processamento e visualização dos dados.

Através da modelagem dos elementos de *software* do sistema, foi possível definir quais requisitos estes deverão possuir para cumprir as funcionalidades desejadas. Por meio do desenvolvimento do *firmware* do modulo ESP32, realizou-se a comunicação e envio de dados do conversor analógico digital para um dispositivo externo (microcomputador), obtendo-se como principais resultados, a realização da amostragem adequada do sinal e também sistemas de configuração e conexão do módulo ao microcomputador para envio de dados.

No âmbito do processamento dos dados do circuito de aquisição, por meio de estudos realizados sobre filtros digitais, projetou-se um filtro IIR Butterworth que foi implementado através de um algoritmo. Desta maneira, realizou-se a filtragem e adequação do sinal respiratório para visualização e também calculo da taxa respiratória. Nesta etapa, obteve-se resultados coerentes, verificando-se que o sinal tornou-se livre de oscilações indesejadas ao monitoramento respiratório. Em conjunto com a filtragem, realizou o desenvolvimento do algoritmo de calculo de taxas respiratórias. Mediante a estudos sobre os algoritmos possíveis, escolheu-se a implementação da abordagem por detecção de picos, onde os resultados encontrados mostraram-se satisfatórios. A analise dos testes mostrou que quanto maior a ocorrência de picos em uma janela de amostras analisadas, maior é a taxa respiratória encontrada. Por fim, o último resultado do projeto, é a interface gráfica desenvolvida. Dado a realização dos testes, o sistema mostrou-se indutivo e também lúdico, proporcionando um *biofeedback* visual que poderá ser empregado em treinamentos realizados em clinicas de psicologia e também reabilitação.

Após as etapas realizadas, tanto no desenvolvimento de *software* como no desenvolvimento do circuito de aquisição de sinais, foi possível realizar o monitoramento da atividade respiratória através de testes em um dos participantes do projeto. Para isto, integrou-se o circuito de aquisição aos *softwares* já desenvolvidos, onde desta forma foi possível caracterizar a atividade respiratória através da movimentação abdominal. A partir da analise dos gráficos gerados pelo sistema e através da observação do comportamento deste, tornou-se evidente a possibilidade da utilização do sistema como uma ferramenta de *biofeedback*, tal como apresentado no objetivo principal deste trabalho de conclusão de curso.

Para trabalhos futuros, pode-se citar os seguintes tópicos de desenvolvimento. Estes poderão agregar valor acadêmico e até mesmo comercial ao protótipo desenvolvido neste trabalho:

- desenvolvimento de aplicativo para sistemas *mobile*, retornando as informações de *biofeedback* no celular;
- utilizar-se de algorimos inteligentes de *machine learning*, para analisar os dados do circuito de aquisição e retornar informações ao usuário;
- utilizar a comunicação *Bluetooth* para enviar os dados para o dispositivo externo.

REFERÊNCIAS

ADAFRUIT. *Conductive Rubber Cord Stretch Sensor + extras!* [S.l.], 2018. Disponível em: <https://www.adafruit.com/product/519>. Acesso em: 25/08/2018. Citado na página 27.

AMABIS, J. M. *Fundamentos da biologia moderna.* 2. ed. [S.l.]: São Paulo: Moderna, 1999. Citado 2 vezes nas páginas 22 e 23.

ANDOLFATO, R. P.; CAMACHO, J. S.; BRITO, G. d. *Extensometria básica*. [S.l.: s.n.], 2004. Citado 3 vezes nas páginas 25, 30 e 46.

ARIMA HIROTO ARAKI, Y. I. M. Respiration by using strain gauge. *2015 7th International Conference on Emerging Trends in Engineering Technology*, Osaka Prefecture University, 2015. Citado na página 18.

BAKER, B. C. Anti-aliasing, analog filters for data acquisition systems. *Microchip Technology Journal*, Microchip, 1999. Citado 2 vezes nas páginas 34 e 49.

BIFULCO, P. et al. A stretchable, conductive rubber sensor to detect muscle contraction for prosthetic hand control. In: IEEE. *E-Health and Bioengineering Conference (EHB), 2017*. [S.l.], 2017. p. 173–176. Citado na página 26.

CHARLTON, P. H.; VILLARROEL, M.; SALGUIERO, F. Waveform analysis to estimate respiratory rate. In: *Secondary Analysis of Electronic Health Records*. [S.l.]: Springer, 2016. p. 377–390. Citado 2 vezes nas páginas 42 e 43.

CHAVES, J. *Biofeedback: A terapia do século 21*. [S.l.], 2017. Disponível em: <http://www.cerebromente.org.br/n04/tecnologia/biofeed.htm>. Acesso em: 08/09/2017. Citado na página 21.

CLEVELANDCLINC. *What are vital signs?* [S.l.], 2014. Disponível em: <https://my.clevelandclinic.org/health/articles/vital-signs>. Acesso em: 08/09/2017. Citado 2 vezes nas páginas 33 e 49.

ESPRESSIF. *ESP32*. [S.l.], 2018. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>. Acesso em: 30/06/2016. Citado 2 vezes nas páginas 43 e 44.

FREY, J. et al. Breeze: Sharing biofeedback through wearable technologies. *eprint arXiv:1802.04995*, 2018. Citado na página 15.

GARGIULO, G. D. et al. A wearable contactless sensor suitable for continuous simultaneous monitoring of respiration and cardiac activity. *Journal of Sensors*, v. 2015, p. 6, 2015. Citado na página 15.

GOZZI, R. *Volumes e Capacidades Pulmonares*. [S.l.], 2017. Disponível em: <http://anatomiafacil.com.br/041-volumes-e-capacidades-pulmonares/>. Acesso em: 03/10/2017. Citado 2 vezes nas páginas 23 e 24.

GUIDI, G. K. *Movimentos Respiratórios*. [S.l.], 2017. Disponível em: <http://www.infoescola.com/fisiologia/movimentos-respiratorios/>. Acesso em: 10/09/2017. Citado 2 vezes nas páginas 22 e 23.

IDEUS. *POWER SYNC documentation*. [S.l.], 2018. Disponível em: <https://www.ideus.com/en/power-sync>. Acesso em: 08/09/2018. Citado na página 50.

INGLE, V. K.; PROAKIS, J. G. *Digital signal processing using MATLAB*. [S.l.]: Cengage Learning, 2010. Citado 2 vezes nas páginas 37 e 40.

JEYHANI, V. et al. Comparison of simple algorithms for estimating respiration rate from electrical impedance pneumography signals in wearable devices. *Springer Berlin Heidelberg*, v. 7, p. 21–31, 2017. ISSN 2190-7188. Citado 6 vezes nas páginas 15, 33, 41, 42, 43 e 60.

KITCHIN, C.; COUNTS, L. *A Designer's Guide to Instrumentation Amplifiers*. 3. ed. [S.l.]: Analog Devices, 2006. Citado na página 32.

KUROSE, K. W. R. J. F. *Computer Neworking - A Top-Down Approach*. 6. ed. [S.l.]: PEARSON, 2013. Citado na página 55.

MICROPYTHON. *MicroPython*. [S.l.], 2017. Disponível em: <https://micropython.org/>. Acesso em: 15/10/2016. Citado na página 52.

NEGRI, L. H. *PeakUtils 1.3.0 documentation*. [S.l.], 2018. Disponível em: <https://peakutils.readthedocs.io/en/latest/reference.html#module-peakutils.peak>. Acesso em: 08/09/2018. Citado 2 vezes nas páginas 60 e 70.

NORTHROP, R. B. *Introduction to instrumentation and measurements*. [S.l.]: CRC press, 2005. Citado 3 vezes nas páginas 24, 28 e 46.

OPPENHEIM, A. V.; SCHAFER, R. W. *Discrete-time signal processing*. [S.l.]: Pearson Education, 2014. Citado 4 vezes nas páginas 36, 37, 38 e 40.

OSMAN, O.; HAYDAR-AHMAD, I.; HAGE-DIAB, A. Thoracic kyphosis alert system. In: IEEE. *Advances in Biomedical Engineering (ICABME), 2015 International Conference on*. [S.l.], 2015. p. 182–184. Citado na página 27.

PERTENCE, A. *Amplificadores Operacionais e Filtros Ativos*. 6. ed. [S.l.]: Bookman, 2007. Citado 4 vezes nas páginas 30, 31, 34 e 35.

PLOTLY. *Plotly Python Open Source Graphing Library*. [S.l.], 2018. Disponível em: <https://plot.ly/python/>. Acesso em: 08/09/2018. Citado na página 62.

ROSSI, A. M. *Biofeedback*. [S.l.], 2017. Disponível em: <http://www.anamrossi.com.br/biofeedback.htm>. Acesso em: 08/09/2017. Citado na página 20.

SEDRA, A.; SMITH, K. C. *Microeletrônica*. 2. ed. [S.l.]: Pearson, 2005. Citado 2 vezes nas páginas 30 e 33.

SILVA, J. D. de A. O emprego do biofeedback como estratégica de manejo do estresse e da ansiedade em atletas: um ensaio clínico. *Revista Brasileira de Terapia Comportamental e Cognitiva*, Associação Brasileira de Psicoterapia e Medicina Comportamental, 1999. Citado na página 15.

SMITH, S. W. et al. *The scientist and engineer's guide to digital signal processing*. [S.l.]: California Technical Pub. San Diego, 1997. Citado 2 vezes nas páginas 37 e 40.

TAYFU, U. *Wi-Fi Manager*. [S.l.], 2017. Disponível em: <https://github.com/tayfunulu/WiFiManager>. Acesso em: 08/09/2018. Citado na página 54.

TEXAS, I. *LM6142/LM6144 17 MHz Rail-to-Rail Input-Output Operational Amplifiers*. [S.l.], 2013. Disponível em: <http://www.ti.com/lit/ds/symlink/lm6142.pdf>. Acesso em: 09/09/2018. Citado na página 48.

____. *INA12x Precision, Low Power Instrumentation Amplifiers (Rev. C)*. [S.l.], 2015. Disponível em: <http://www.ti.com/product/INA128/datasheet>. Acesso em: 08/09/2017. Citado 2 vezes nas páginas 47 e 48.

____. *TPS6040x Unregulated 60-mA Charge Pump Voltage Inverter*. [S.l.], 2015. Disponível em: <http://www.ti.com/lit/ds/symlink/tps60402.pdf>. Acesso em: 08/09/2018. Citado 2 vezes nas páginas 50 e 51.

VARAKI, E. S.; BREEN, P. P.; GARGIULO, G. D. Quantification of a low-cost stretchable conductive sensor using an expansion/contraction simulator machine: a step towards validation of a noninvasive cardiac and respiration monitoring prototype. *Machines*, Multidisciplinary Digital Publishing Institute, v. 5, n. 4, p. 22, 2017. Citado na página 27.

WEBSTER, J. G. *Medical instrumentation application and design*. [S.l.]: John Wiley & Sons, 2009. Citado 4 vezes nas páginas 24, 25, 26 e 28.

YILDIZ, S. K.; MUTLU, R.; ALICI, G. Performance quantification of strain sensors for flexible manipulators. In: IEEE. *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on*. [S.l.], 2016. p. 584–589. Citado na página 27.

ZHU, Q.; KONG, X. long; XIE, Y. ying. The influence of biofeedback on respiratory. *ICSAI*, Beijing Jiaotong University, 2012. Citado 2 vezes nas páginas 15 e 17.

APÊNDICE A – PROGRAMA AMOSTRAGEM E ENVIO ESP32

```
1  import socket
2  import network
3  import machine
4
5  # Configure network
6  ap = network.WLAN(network.AP_IF)
7  ap.config(essid='RESPIRACAO-MONITOR', authmode=network.AUTH_WPA_WPA2_PSK,
       password='respira123')
8  ap.active(True)
9
10 adc = machine.ADC(machine.Pin(36))
11 adc.atten(adc.ATTN_6DB)
12 value = adc.read()
13
14 interruptCounter = 0
15 timer = machine.Timer(-1)
16
17 def handleInterrupt(timer):
18     global interruptCounter
19     interruptCounter = interruptCounter+1
20
21 s = socket.socket()
22 addr = socket.getaddrinfo('0.0.0.0', 8080)[0][-1]
23 s.bind(addr)
24 s.listen(1)
25
26 timer.init(period=20, mode=machine.Timer.PERIODIC, callback=handleInterrupt
       )
27
28 while True:
29     print('Esperando conexao com o GUI!')
30     connection, client_address = s.accept()
31     try:
32         print('ESP32', client_address)
33         while True:
34             if interruptCounter>0:
```

```
35                  state = machine.disable_irq()
36                  interruptCounter = interruptCounter-1
37                  machine.enable_irq(state)
38                  value = str(adc.read())
39                  try:
40                      if len(value)<4:
41                          if len(value) == 3:
42                              value = '0'+ value
43                          if len(value) == 2:
44                              value = '00'+ value
45                          if len(value) == 1:
46                              value = '000'+ value
47                      msg = connection.send(value)
48                      if msg:
49                          print( value )
50                      else:
51                          print('Perdeu-se a comunicacao', client_address)
52                          break
53                  except:
54                      connection.close()
55                      break
56      finally:
57          connection.close()
```

APÊNDICE B – PROGRAMA CONFIGURAÇÃO *WI-FI* ESP32

```
1  import network
2  import socket
3  import ure
4  import time
5  import machine
6
7  ap_ssid = "RESPIRATORY-MONITOR"
8  ap_password = "respira123"
9  ap_authmode = 3  # WPA2
10
11 adc = machine.ADC(machine.Pin(36))
12 adc.atten(adc.ATTN_6DB)
13 value = adc.read()
14 interruptCounter = 0
15 timer = machine.Timer(-1)
16
17 NETWORK_PROFILES = 'wifi.dat'
18
19 wlan_ap = network.WLAN(network.AP_IF)
20 wlan_sta = network.WLAN(network.STA_IF)
21
22 server_socket = None
23
24
25 def handleInterrupt(timer):
26     global interruptCounter
27     interruptCounter = interruptCounter+1
28
29 def get_connection():
30     """return a working WLAN(STA_IF) instance or None"""
31
32     # First check if there already is any connection:
33     if wlan_sta.isconnected():
34         return wlan_sta
35
36     connected = False
```

```
37      try:
38          # ESP connecting to WiFi takes time, wait a bit and try again:
39          time.sleep(3)
40          if wlan_sta.isconnected():
41              return wlan_sta
42
43          # Read known network profiles from file
44          profiles = read_profiles()
45
46          # Search WiFis in range
47          wlan_sta.active(True)
48          networks = wlan_sta.scan()
49
50          AUTHMODE = {0: "open", 1: "WEP", 2: "WPA-PSK", 3: "WPA2-PSK", 4: "
                WPA/WPA2-PSK"}
51          for ssid, bssid, channel, rssi, authmode, hidden in sorted(networks
                , key=lambda x: x[3], reverse=True):
52              ssid = ssid.decode('utf-8')
53              encrypted = authmode > 0
54              print("ssid: %s chan: %d rssi: %d authmode: %s" % (ssid,
                    channel, rssi, AUTHMODE.get(authmode, '?')))
55              if encrypted:
56                  if ssid in profiles:
57                      password = profiles[ssid]
58                      connected = do_connect(ssid, password)
59                  else:
60                      print("skipping unknown encrypted network")
61              else:  # open
62                  connected = do_connect(ssid, None)
63              if connected:
64                  break
65
66      except OSError as e:
67          print("exception", str(e))
68
69      # start web server for connection manager:
70      if not connected:
71          connected = start()
72
```

```
73      return wlan_sta if connected else None
74
75
76  def read_profiles():
77      with open(NETWORK_PROFILES) as f:
78          lines = f.readlines()
79      profiles = {}
80      for line in lines:
81          ssid, password = line.strip("\n").split(";")
82          profiles[ssid] = password
83      return profiles
84
85
86  def write_profiles(profiles):
87      lines = []
88      for ssid, password in profiles.items():
89          lines.append("%s;%s\n" % (ssid, password))
90      with open(NETWORK_PROFILES, "w") as f:
91          f.write(''.join(lines))
92
93
94  def do_connect(ssid, password):
95      wlan_sta.active(True)
96      if wlan_sta.isconnected():
97          return None
98      print('Trying to connect to %s...' % ssid)
99      wlan_sta.connect(ssid, password)
100     for retry in range(100):
101         connected = wlan_sta.isconnected()
102         if connected:
103             break
104         time.sleep(0.1)
105         print('.', end='')
106     if connected:
107         print('\nConnected. Network config: ', wlan_sta.ifconfig())
108     else:
109         print('\nFailed. Not Connected to: ' + ssid)
110     return connected
111
```

```
112
113 def send_header(client, status_code=200, content_length=None ):
114     client.sendall("HTTP/1.0 {} OK\r\n".format(status_code))
115     client.sendall("Content-Type: text/html\r\n")
116     if content_length is not None:
117       client.sendall("Content-Length: {}\r\n".format(content_length))
118     client.sendall("\r\n")
119
120
121 def send_response(client, payload, status_code=200):
122     content_length = len(payload)
123     send_header(client, status_code, content_length)
124     if content_length > 0:
125         client.sendall(payload)
126     client.close()
127
128 def handle_root(client):
129     wlan_sta.active(True)
130     ssids = sorted(ssid.decode('utf-8') for ssid, *_ in wlan_sta.scan())
131     send_header(client)
132     client.sendall("""\
133         <html>
134             <h1 style="color: #5e9ca0; text-align: center;">
135                 <span style="color: #ff0000;">
136                    Wi-Fi Cliente Configura&#231;&#227;o ESP32
137
138                 </span>
139             </h1>
140             <form action="configure" method="post">
141                 <table style="margin-left: auto; margin-right: auto;">
142                     <tbody>
143     """)
144     while len(ssids):
145         ssid = ssids.pop(0)
146         client.sendall("""\
147                         <tr>
148                             <td colspan="2">
149                                 <input type="radio" name="ssid" value="{0}"
                                        />{0}
```

```
150                                 </td>
151                             </tr>
152             """.format(ssid))
153         client.sendall("""\
154                             <tr>
155                                 <td>Password:</td>
156                                 <td><input name="password" type="password" /></
                                        td>
157                             </tr>
158                         </tbody>
159                     </table>
160                     <p style="text-align: center;">
161                         <input type="submit" value="Submit" />
162                     </p>
163                     <p style="text-align: center;">
164                        <button name="GoAp" value="Ap" type="submit">Go Ap</
                              button>
165                     </p>
166                 <form>
167 
168 
169                 <p>&amp;nbsp;</p>
170                 <hr />
171                 <h5>
172                     <span style="color: #ff0000;">
173                         O seu ssid e senha s&amp;#227;o salvados dentro do "%(
                                filename)s" para sua conex&amp;#227;o, fique tranquilo (
                                a), &amp;#233; seguro!
174                     </span>
175                 </h5>
176                 <hr />
177                 <h2 style="color: #2e6c80;">
178                     Informa&amp;#231;&amp;#245;es importantes:
179 :
180                 </h2>
181                 <ul>
182 
183                     <li>
184                         Agradecimentos ao Tayfu pelo c&amp;oacute;digo de refer&
```

```
                                ecirc;ncia
185   <a href="https://github.com/tayfunulu/WiFiManager"
186                                 target="_blank" rel="noopener">tayfunulu/
                                        WiFiManager</a>.
187                         </li>
188                     </ul>
189
190
191                 </html>
192             """ % dict(filename=NETWORK_PROFILES))
193             client.close()
194
195
196     def handle_configure(client, request):
197         match = ure.search("ssid=([^&]*)&password=(.*)", request)
198         flagAp = str(request).find('GoAp=Ap')
199         adc = machine.ADC(machine.Pin(36))
200         adc.atten(adc.ATTN_6DB)
201         value = adc.read()
202         global interruptCounter
203         timer = machine.Timer(-1)
204
205         if match is None:
206                 if flagAp !=-1:
207
208                     send_response(client, "Continuando como Ap", status_code=400)
209                     server_socket.close()
210                     wlan_sta.active(False)
211                     s = socket.socket()
212                     addr = socket.getaddrinfo('0.0.0.0', 8080)[0][-1]
213                     print(addr)
214                     s.bind(addr)
215                     s.listen(1)
216                     timer.init(period=20, mode=machine.Timer.PERIODIC, callback=
                            handleInterrupt)
217
218                     while True:
219                         print('Esperando conex o com o GUI!')
220                         connection, client_address = s.accept()
```

```
221                     try:
222                         print('ESP32', client_address)
223                         while True:
224                             if interruptCounter>0:
225                                 state = machine.disable_irq()
226                                 interruptCounter = interruptCounter-1
227                                 machine.enable_irq(state)
228                                 value = str(adc.read())
229                                 try:
230                                     if len(value)<4:
231                                         if len(value) == 3:
232                                             value = '0'+ value
233                                         if len(value) == 2:
234                                             value = '00'+ value
235                                         if len(value) == 1:
236                                             value = '000'+ value
237                                     msg = connection.send(value)
238                                     if msg:
239                                         print( value )
240                                     else:
241                                         print('Perdeu-se a comunica  o',
                                                client_address)
242                                         break
243                                 except:
244                                     connection.close()
245                                     break
246                     finally:
247                         connection.close()
248             else:
249                 send_response(client, "Parameters not found", status_code
                        =400)
250
251             return False
252
253     # version 1.9 compatibility
254     try:
255         ssid = match.group(1).decode("utf-8").replace("%3F", "?").replace("
                %21", "!")
256         password = match.group(2).decode("utf-8").replace("%3F", "?").
```

```
            replace("%21", "!")
257     except Exception:
258         ssid = match.group(1).replace("%3F", "?").replace("%21", "!")
259         password = match.group(2).replace("%3F", "?").replace("%21", "!")
260 
261     if len(ssid) == 0:
262         send_response(client, "SSID must be provided", status_code=400)
263         return False
264 
265     if do_connect(ssid, password):
266         response = """\
267             <html>
268                 <center>
269                     <br><br>
270                     <h1 style="color: #5e9ca0; text-align: center;">
271                         <span style="color: #ff0000;">
272                             ESP32 conectado com sucesso na rede WiFi %(ssid
                                )s.
273                         </span>
274                     </h1>
275                     <br><br>
276                 </center>
277             </html>
278         """ % dict(ssid=ssid)
279         send_response(client, response)
280         try:
281             profiles = read_profiles()
282         except OSError:
283             profiles = {}
284         profiles[ssid] = password
285         write_profiles(profiles)
286 
287         time.sleep(5)
288 
289         return True
290     else:
291         response = """\
292             <html>
293                 <center>
```

```
294                     <h1 style="color: #5e9ca0; text-align: center;">
295                         <span style="color: #ff0000;">
296                             Nao foi possivel conectar em %(ssid)s.
297                         </span>
298                     </h1>
299                     <br><br>
300                     <form>
301                         <input type="button" value="Go back!" onclick="
                                history.back()"></input>
302                     </form>
303                 </center>
304             </html>
305         """ % dict(ssid=ssid)
306         send_response(client, response)
307         return False
308
309
310 def handle_not_found(client, url):
311     send_response(client, "Path not found: {}".format(url), status_code
            =404)
312
313
314 def stop():
315     global server_socket
316
317     if server_socket:
318         server_socket.close()
319         server_socket = None
320
321
322 def start(port=80):
323     global server_socket
324
325     addr = socket.getaddrinfo('0.0.0.0', port)[0][-1]
326
327     stop()
328
329     wlan_sta.active(True)
330     wlan_ap.active(True)
```

```
331
332     wlan_ap.config(essid=ap_ssid, password=ap_password, authmode=
            ap_authmode)
333
334     server_socket = socket.socket()
335     server_socket.bind(addr)
336     server_socket.listen(1)
337
338     print('Connect to WiFi ssid ' + ap_ssid + ', default password: ' +
            ap_password)
339     print('and access the ESP via your favorite web browser at 192.168.4.1.
            ')
340     print('Listening on:', addr)
341
342     while True:
343         if wlan_sta.isconnected():
344             return True
345
346         client, addr = server_socket.accept()
347         print('client connected from', addr)
348         try:
349             client.settimeout(5.0)
350
351             request = b""
352             try:
353                 while "\r\n\r\n" not in request:
354                     request += client.recv(512)
355             except OSError:
356                 pass
357
358             print("Request is: {}".format(request))
359             if "HTTP" not in request:  # skip invalid requests
360                 continue
361
362             # version 1.9 compatibility
363             try:
364                 url = ure.search("(?:GET|POST) /(.*?)(?:\\?.*?)? HTTP",
                        request).group(1).decode("utf-8").rstrip("/")
365
```

```
366             except Exception:
367                 url = ure.search("(?:GET|POST) /(.*?)(?:\\?.*?)? HTTP",
                        request).group(1).rstrip("/")
368             print("URL is {}".format(url))
369             if url == "":
370                 handle_root(client)
371             elif url == "configure":
372                 handle_configure(client, request)
373             else:
374                 handle_not_found(client, url)
375         finally:
376             client.close()
```

APÊNDICE C – PROGRAMA CONEXÃO AUTOMÁTICA ESP32

```
1  import wifimgr
2  import socket
3  import network
4  import machine
5
6  wlan = wifimgr.get_connection()
7  if wlan is None:
8      print("Nao foi possivel realizar a conexao com o Ap")
9      while True:
10         pass
11
12 def handleInterrupt(timer):
13     global interruptCounter
14     interruptCounter = interruptCounter+1
15
16 adc = machine.ADC(machine.Pin(36))
17 adc.atten(adc.ATTN_6DB)
18 value = adc.read()
19 interruptCounter = 0
20 timer = machine.Timer(-1)
21
22 sockUdp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
23 sockUdp.bind(("", 37020))
24 data, addr = sockUdp.recvfrom(1024)
25 sockUdp.sendto('Conectou-se no modulo ESP32', addr)
26 sockUdp.close()
27
28 s = socket.socket()
29 addr = socket.getaddrinfo('0.0.0.0', 8080)[0][-1]
30 s.bind(addr)
31 s.listen(1)
32
33 timer.init(period=20, mode=machine.Timer.PERIODIC, callback=handleInterrupt
       )
34
35 while True:
```

```
36      print('Esperando conexao com o GUI!')
37      connection, client_address = s.accept()
38      try:
39          print('ESP32', client_address)
40          while True:
41              if interruptCounter>0:
42                  state = machine.disable_irq()
43                  interruptCounter = interruptCounter-1
44                  machine.enable_irq(state)
45                  value = str(adc.read())
46                  try:
47                      if len(value)<4:
48                          if len(value) == 3:
49                              value = '0'+ value
50                          if len(value) == 2:
51                              value = '00'+ value
52                          if len(value) == 1:
53                              value = '000'+ value
54                      msg = connection.send(value)
55                      if msg:
56                          print( value )
57                      else:
58                          print('Perdeu-se a comunicacao', client_address)
59                          break
60                  except:
61                      connection.close()
62                      break
63      finally:
64          connection.close()
```

APÊNDICE D – *SOFTWARE* DE ENVIO E CONEXÃO

```
1  import socket
2  import sys
3  import numpy as np
4  import threading
5  from subprocess import check_output
6  import time
7
8
9  class Stream():
10     def __init__(self):
11         scanOutput = check_output("iwgetid")
12         for line in scanOutput.split():
13             if line.startswith(b"ESSID"):
14                 ssid = line.split(b'"')[1]
15
16         if ssid == b'RESPIRATORY-MONITOR':
17             self.server_address = ('192.168.4.1', 8080)
18         else:
19             server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM,
                   socket.IPPROTO_UDP)
20             server.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
21             server.bind(("", 5000))
22             message = b"Mensagem para o modulo"
23             countSender = 0
24             while countSender < 4:
25                 server.sendto(message, ('<broadcast>', 37020))
26                 print("message sent!")
27                 countSender = countSender + 1
28                 time.sleep(1)
29             data, addr = server.recvfrom(1024)
30             print(data)
31             server.close()
32             self.server_address = (addr[0], 8080)
33         self.STREAM_SIZE = 100
34         self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
35         time.sleep(1)
```

```
36          self.sock.connect(self.server_address)
37          self.data = 0
38
39      def processData(self):
40          try:
41              self.data = self.sock.recv(4)
42          except:
43              print('Erro!')
44
45      def putData(self):
46          t=threading.Thread(target = self.processData)
47          t.start()
48          t.join()
49
50      def getData(self):
51          return self.data
52
53      def close(self):
54          self.sock.close()
```

APÊNDICE E – *SOFTWARE* PRINCIPAL

```
1  from PyQt4 import QtGui,QtCore
2  import sys
3  import layout
4  import numpy as np
5  import pylab
6  import time
7  import pyqtgraph
8  import data
9  from scipy import signal
10 import peakutils
11 import threading
12 import report
13 import webbrowser
14
15 class App(QtGui.QMainWindow, layout.Ui_MainWindow):
16     def __init__(self, parent = None):
17         self.sinalFiltrado = np.zeros(10)
18         self.Y = np.zeros(1000)
19         self.mark = np.zeros(1000)
20         self.mark[500] = 1.0
21         self.read = np.zeros(1000)
22         self.cont = 0
23         self.thread = ThreadClass()
24         self.connect(self.thread,QtCore.SIGNAL('Wave and RR and Timer'),
               self.takeParameters)
25         self.thread.start()
26         self.completeBar = 0
27         self.rrRead = 0.0
28         self.lastSample = 0.0
29         self.timeTraining = 0
30         super(App, self).__init__(parent)
31         self.setupUi(self)
32         self.grPCM.plotItem.showGrid(True, True, 0.7)
33
34     def takeParameters(self,read,rrRead,lastSample,timeTraining):
35         self.read = read
```

```
36            self.rrRead = rrRead
37            self.lastSample = lastSample
38            self.timeTraining  = timeTraining
39
40        def update(self):
41            n=np.arange(1000)
42            pen=pyqtgraph.mkPen(color=1,width=1)
43            self.grPCM.plot(n,self.read[::-1],pen=pen,clear=True,antialias=True
                  )
44            self.completeBar = int(self.lastSample*1250 - 250)
45            self.pbLevel.setValue(self.completeBar)
46            self.cont = self.cont + 1
47            self.lcdNumber.display(self.rrRead)
48            self.lcdNumber_3.display(self.timeTraining)
49            QtCore.QTimer.singleShot(20, self.update)
50
51    def main():
52        app = QtGui.QApplication(sys.argv)
53        form = App()
54        form.show()
55        form.update()
56        app.exec_()
57        print("DONE")
58
59    class ThreadClass(QtCore.QThread):
60        def __init__(self):
61            QtCore.QThread.__init__(self)
62            self.samplePeriod = 0.02
63            self.x = data.Stream()
64            self.timeWindowRR = 1200
65            self.b, self.a = signal.butter(3, 0.04)
66            self.zi = signal.lfilter_zi(self.b, self.a)
67            self.bufferRR = np.zeros(self.timeWindowRR)
68            self.respiratoryRate = 0.0
69            self.samplesRR = np.arange(0,self.timeWindowRR)
70            self.file = open('respiratorydata.csv', 'r+')
71            self.file.truncate(0)
72            self.file = open('respiratorydata.csv','w')
73
```

```
74
75      def calculateRR(self):
76          indexesInvalide = []
77          countInvalide = 0
78          breathingFiltred = self.bufferRR
79          #base = peakutils.baseline(breathingFiltred, 2)
80          #breathingFiltred = breathingFiltred-base
81
82          print(np.max(breathingFiltred))
83          indexes = peakutils.indexes(breathingFiltred, thres=0.4, min_dist
                =60)
84          peakValues = breathingFiltred[indexes]
85          thre = 0.4*0.5*(np.max(breathingFiltred)+np.min(breathingFiltred))
86          print(thre)
87          print(peakValues)
88          for i in range(len(peakValues)):
89              if peakValues[i] < thre:
90                  indexesInvalide.insert(countInvalide,i)
91                  countInvalide = countInvalide + 1
92
93          if countInvalide-1 == len(indexes):
94              print("Non breathing detect")
95              pass
96          else:
97              timesRR = self.samplesRR[indexes]*self.samplePeriod
98              timesRR = np.delete(timesRR,indexesInvalide)
99
100             if len(timesRR) == 1:
101                 self.respiratoryRate = float("{0:.2f}".format((1/(self.
                        timeWindowRR*self.samplePeriod))*60))
102             else:
103                 distanceTimesRR = [x - timesRR[i - 1] for i, x in enumerate
                        (timesRR) if i > 0]
104                 self.respiratoryRate = float("{0:.2f}".format((1/np.mean(
                        distanceTimesRR))*60))
105
106     def run(self):
107         read = np.zeros(1000)
108         sinal = np.zeros(1000)
```

```
109         contSamplesRR = 0
110         rrRead = 0.0
111         start = time.time()
112         time.clock()
113         elapsed = 0
114         timeTraining = 0
115         z = np.zeros(len(self.a))
116         h = np.zeros(len(self.b))
117         nb = len(self.b)
118         na = len(self.a)
119         p = 0
120         v = 0
121         g = 0
122         acc = 0
123         lastSample = 0
124         contData = 0
125         while True:
126             self.x.putData()
127             breathingWave = float(self.x.getData())/4096
128             lineCsv = str(contData)+';'+str(breathingWave)+"\n"
129             contData =  contData + 1
130             self.file.write(lineCsv)
131             if p > nb-1:
132                 p=0
133 
134             z[p] = breathingWave
135             acc = 0
136             k = p
137             p = p + 1
138 
139             for j in range(1,nb):
140                 acc = acc + self.b[j]*z[k]
141                 k = k-1
142                 if k < 0:
143                     k = nb-1
144             for i in range(1,na):
145                 acc = acc - self.a[i]*h[g]
146                 g = g-1
147                 if g < 0:
```

```
148                     g = na-1
149
150             if v > na-1:
151                 v =0
152
153             h[v] = acc
154             g = v
155             v = v + 1
156
157             lastSample = acc
158             sinal = np.insert(sinal,0,acc)
159             sinal = np.delete(sinal,-1)
160             read =  sinal
161             self.bufferRR[contSamplesRR] = breathingWave
162             contSamplesRR =  contSamplesRR + 1
163             rrRead = self.respiratoryRate
164             self.emit(QtCore.SIGNAL('Wave and RR and Timer'),read,
165             rrRead,lastSample, timeTraining)
166
167             elapsed = int(time.time() - start)
168
169             if elapsed == 60:
170                 timeTraining = timeTraining + 1
171                 start = time.time()
172             if contSamplesRR ==self.timeWindowRR:
173                 t=threading.Thread(target = self.calculateRR)
174                 t.start()
175                 t.join()
176                 contSamplesRR = 0
177 if __name__ == '__main__':
178     print('Inicio do sw!')
179     main()
180     report.generateReport()
```

APÊNDICE F – *SOFTWARE* GERAÇÃO DE RELATÓRIO

```
1  import plotly.plotly as py
2  import plotly.graph_objs as go
3  import plotly.offline as ply
4  import numpy as np
5  import peakutils
6  from peakutils.plot import plot as pplot
7  import matplotlib.pyplot as plt
8  from scipy import signal
9  import webbrowser
10
11 def generateReport():
12     indexesInvalide = []
13     countInvalide = 0
14     n, breathing = np.loadtxt('respiratorydata.csv', delimiter=';', unpack=
           True)# use the peaktutils to detect sine peaks
15     b, a = signal.butter(4, 0.04, 'low')
16     samplePeriod = 0.02
17     zi = signal.lfilter_zi(b, a)
18     breathingFiltred, _ = signal.lfilter(b, a, breathing, zi=zi*breathing
           [0])
19     nb = len(breathing)
20     timeWindow = 1200
21     numCicle = int(nb/timeWindow)
22     countCicle = timeWindow
23     rrWave = np.array([])
24     respiratoryRate = 0.0
25     samplesRR = np.arange(0,timeWindow)
26     samplePeriod = 0.02
27     rrVector = np.array([])
28     periods = np.arange(0, numCicle)
29     for i in range(0, numCicle):
30         analisesWave = breathingFiltred[countCicle-timeWindow:countCicle]
31         indexes = peakutils.indexes(analisesWave, thres=0.4, min_dist=60)
32         peakValues = analisesWave[indexes]
33         thre = (np.max(analisesWave)+np.min(analisesWave))*0.5*0.4
34
```

```
35          countCicle = countCicle +  timeWindow
36          for i in range(len(peakValues)):
37              if peakValues[i] < thre:
38                  indexesInvalide.insert(countInvalide,i)
39                  countInvalide = countInvalide + 1
40          if countInvalide-1 == len(indexes):
41              print("Non breathing detect")
42              respiratoryRate = 0.0
43          else:
44              timesRR = samplesRR[indexes]*samplePeriod
45              timesRR = np.delete(timesRR,indexesInvalide)
46
47          if len(timesRR) == 1:
48              respiratoryRate = (1/(timeWindow*samplePeriod))*60
49          else:
50              distanceTimesRR = [x - timesRR[i - 1] for i, x in enumerate(
                    timesRR) if i > 0]
51              respiratoryRate = float("{0:.2f}".format((1/np.mean(
                    distanceTimesRR))*60))
52
53          rr = np.ones(timeWindow)*respiratoryRate
54          rrWave = np.concatenate((rrWave,rr))
55          rrVector = np.append(rrVector,respiratoryRate)
56
57      trace1 = go.Table(
58          domain=dict(x=[0, 1],
59                      y=[0.7, 1.0]),
60          columnwidth=[1, 2, 2, 2],
61          columnorder=[0, 1, 2, 3, 4],
62          header = dict(height = 50,
63                      values = [['<b>Periodo</b>'],['<b>Taxa Respirat ria</b
                            >']],
64                      line = dict(color='rgb(50, 50, 50)'),
65                      align = ['left'] * 5,
66                      font = dict(color=['rgb(45, 45, 45)'] * 5, size=14),
67                      fill = dict(color='#998067')),
68          cells = dict(values = [periods,rrVector],
69                      line = dict(color='#506784'),
70                      align = ['left'] * 5,
```

```
71                      font = dict(color=['rgb(40, 40, 40)'] * 5, size=12),
72                      format = [None] + [", .2f"] * 2 + [',.4f'],
73                      prefix = [None] * 2 + ['$', u'\u20BF'],
74                      suffix=[None] * 4,
75                      height = 27,
76                      fill = dict(color=['rgb(255,231,200)', 'rgb
                            (255,231,200)']))
77      )
78
79      trace4=go.Scatter(
80          x=n,
81          y=breathingFiltred,
82          xaxis='x1',
83          yaxis='y1',
84          mode='lines',
85          line=dict(width=2, color='#9748a1'),
86          name='Sinal respirat rio'
87      )
88
89      trace5=go.Scatter(
90          x=n[0:numCicle*timeWindow],
91          y=rrWave,
92          xaxis='x2',
93          yaxis='y2',
94          mode='lines',
95          line=dict(width=2, color='#353D21'),
96          name='taxa respirat ria'
97      )
98
99
100     axis=dict(
101         showline=True,
102         zeroline=False,
103         showgrid=True,
104         mirror=True,
105         ticklen=4,
106         gridcolor='#ffffff',
107         tickfont=dict(size=10)
108     )
```

```
109
110     layout2 = dict(
111         width=950,
112         height=800,
113         autosize=False,
114         title='Relat rio treinamento respirat rio',
115         margin = dict(t=100),
116         showlegend=False,
117         xaxis1=dict(axis, **dict(anchor='y1', showticklabels=False)),
118         xaxis2=dict(axis, **dict(title = ' N mero de amostras', anchor='y2
                ', showticklabels=False)),
119         yaxis1=dict(axis, **dict(title = 'Amplitude (u.a)',domain=[2 * 0.21
                 + 0.02 + 0.02, 0.68], anchor='x1')),
120         yaxis2=dict(axis, **dict(title = 'Taxa respirat ria (rpm)', domain
                =[0.21 + 0.02, 2 * 0.21 + 0.02], anchor='x2')),
121         plot_bgcolor='rgba(228, 222, 249, 0.65)'
122     )
123
124     fig2 = dict(data=[trace1, trace4, trace5], layout=layout2)
125     ply.plot(fig2, filename='report.html', auto_open=False)
126     chrome_path = '/usr/bin/google-chrome %s'
127     webbrowser.get(chrome_path).open('report.html')
```