UNIVERSIDADE FEDERAL DO PARANÁ

ANDRÉ ALLAN HRECEK MACHOSKI
MARCELO SILVA DA CRUZ

SISTEMA DE MONITORAMENTO E DETECÇÃO DE ANOMALIAS CARDIORRESPIRATÓRIAS COM AUXÍLIO DE UM SENSOR DE OXIMETRIA

CURITIBA
2018

ANDRÉ ALLAN HRECEK MACHOSKI
MARCELO SILVA DA CRUZ

SISTEMA DE MONITORAMENTO E DETECÇÃO DE ANOMALIAS CARDIORRESPIRATÓRIAS COM AUXÍLIO DE UM SENSOR DE OXIMETRIA

Monografia apresentada à disciplina de Trabalho de Conclusão de Curso B como requisito parcial à conclusão do Curso de Engenharia Elétrica (Ênfase em Sistemas Eletrônicos Embarcados), Setor de Tecnologia, da Universidade Federal do Paraná.
Orientador: Prof. Ph.D. João da Silva Dias

CURITIBA
2018

TERMO DE APROVAÇÃO

ANDRÉ ALLAN HRECEK MACHOSKI
MARCELO SILVA DA CRUZ

SISTEMA DE MONITORAMENTO E DETECÇÃO DE ANOMALIAS CARDIORRESPIRATÓRIAS COM AUXÍLIO DE UM SENSOR DE OXIMETRIA

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica com Ênfase em Sistemas Eletrônicos Embarcados da Universidade Federal do Paraná como requisito à obtenção do título de Engenheiro Eletricista, pela seguinte banca avaliadora:

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

_____ Prof. Dr. José Carlos da Cunha - Departamento de Engenharia Elétrica, UFPR

_____ Prof. M.Sc. Bruno Pohlot Ricobom Departamento de Engenharia Elétrica, UFPR

Curitiba, 04 de dezembro de 2018

RESUMO

Para analisar o estado de saúde de um ser humano é importante realizar medições de sinais vitais que são excelentes indicadores do funcionamento do sistema circulatório e respiratório. A medição desses parâmetros colaboram significantemente com o serviço da equipe médica proporcionando um diagnóstico mais assertivo. Em vista disso, durante o período compreendido pelo trabalho de conclusão de curso, os integrantes do presente projeto desenvolveram um sistema de comunicação entre profissionais da saúde e pacientes. Em virtude de que esse sistema é baseado na aplicação de um sensor oximétrico para mediçao de sinais, o qual possui uma unidade de tratamento para filtragem, amplificação, amostragem e processamento de dados. Por conseguinte, os resultados esperados são de caráter computacional composto por simulações de circuitos, desenvolvimento de *software* para processamento de dados e programação em Java para o aplicativo do ambiente *Android*. Por fim, no decorrer do trabalho foi construído um prótipo para validar resultados provenientes de simulações e pode ser utilizado em aplicações biomédicas para monitoramento de frequência cardíaca e oxigenação sanguínea.

Palavras-chaves:Oximetria, engenharia biomédica, oxigenação sanguínea.

ABSTRACT

To analyze the health status of a human is important to collect measures of vital signs which are excellent indicators of the blood circulation system operation. These parameters measurement collaborate significantly with the medical team service to provide a diagnosis more assertive. Thus, during the period by undergraduate thesis, the project members developed a communication system between health professionals and patients. Because this system is based on the application of a oximetry sensor for signal measurement, which has a processing unit for filtering, amplification, sampling and data processing. Therefore, the expected results are of computational character composed by circuit simulations, development of software for data processing and programming in Java for the environment application Android. Finally, throughout project it was created a prototype to validate results from simulations which can be used at biomedical applications to monitor heart rate and blood oxygenation.

Key-words: Oximetry, biomedical engineering, blood oxygenation.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A engenharia contribui fortemente para a evolução da ciência ao aplicar principalmente conhecimentos físicos e matemáticos com a finalidade de criar e aperfeiçoar máquinas e equipamentos, desenvolvendo sistemas e soluções com o propósito de solucionar problemas da sociedade. Desde as últimas décadas o mundo vem se tornando cada vez mais industrial com *softwares* embarcados e dispositivos que aumentam consideravelmente a velocidade de execução de processos e procedimentos na área biomédica gerando maior segurança e confiabilidade. No entanto, quando se ingressa no domínio de aplicações de monitoramento e detecção de anomalias cardiorrespiratórias, surge um dos grandes desafios da atualidade que é a rigidez para assegurar confiabilidade e qualidade sem gerar desconforto ao paciente.

De acordo com a tese de mestrado de Moura (2012) a comunicação médico-paciente é fundamental para a concretização de tratamentos e a efetiva prática da medicina, a qual se sustenta em três pilares da racionalidade científica: o caráter generalizante para a utilização de ideias universais, o mecanicista que examina o universo como uma máquina causal linear, e o analítico que aborda os métodos práticos e teóricos para validação dos conceitos. Os pilares supracitados possibilitaram a evolução de técnicas medicinais, todavia devido ao generalismo os procedimentos alopáticos podem ser limitados para casos especiais. Além disso, o avanço da personalização de serviços se tornou uma tendência para atender usuários dos mais amplos segmentos, e aplicando o mesmo conceito na área da saúde se obtém a medicina adaptativa.

Analisando a tendência de serviços personalizados, foi construído um sistema de monitoramento com a utilização da técnica de oximetria para aprimorar o tratamento de anomalias cardiorrespiratórias visando facilitar a comunicação entre profissionais da área da saúde e os pacientes. Consequentemente, com um rápido acesso a informação é possível ter um melhor planejamento para casos inesperados em que o caráter generalista não apresente a eficácia adequada. Além da personalização do sistema, a comunicação rápida e eficaz de baixo custo simplifica a transmissão de informação e pode ser empregada sem nenhuma restrição em qualquer tipo de paciente. Contudo, devido o caráter de urgência, o projeto é focado em enfermos que apresentam dificuldades de comunicação.

Combinando sensor de oximetria o qual possui circuitos para tratamento de sinal como filtros passa-baixa, amplificadores de transimpedância e de estágio final, e interface rápida, é possível alcançar uma integração prática entre *hardware* e *software* para realizar a aquisição de sinais com o método oximétrico. Portanto, os resultados que podem ser alcançados ao se explorar o projeto possuem grande relevância para ajudar a salvar vidas, visto que atualmente as doenças cardiorrespiratórias são responsáveis por um grande número de óbitos no mundo, e ter acesso a dados clínicos rapidamente contribui para diagnósticos médicos mais assertivos em situações de emergência.

1.1 OBJETIVOS

A seguir são descritos os objetivos do projeto. Para melhor planejamento e entendimento o item foi dividido em geral e específico.

O objetivo geral coloca a visão a longo prazo do projeto, explicando o que se espera de sua evolução para o trabalho de conclusão de curso. Os específicos são referentes ao que a equipe se compromete à apresentar no final da disciplina.

1.1.1 Objetivo Geral

Desenvolver um sistema de comunicação entre profissionais da saúde e pacientes no ambiente *Android* com a ferramenta *Amazon Web Service* utilizando como *hardware* o sensor oximétrico, auxiliando no diagnóstico e tratamento de enfermidades.

1.1.2 Objetivos Específicos

Dentre os objetivos específicos, destacam-se:

- levantar os requisitos e projetar os amplificadores e filtros que farão o tratamento do sinal de oxigenação sanguínea com o auxílio de *softwares* de simulação, tal como o *Quite Universal Circuit Simulator* (Qucs) e de desenvolvimento, como a ferramenta *CAD Eagle*;
- concepção de uma ferramenta de comunicação com plataforma *mobile* para servir de alerta em situações adversas proporcionando maior confiabilidade e velocidade de acesso à informação;

- criação de um sistema de armazenamento de dados utilizando a plataforma *Amazon Web Service*.

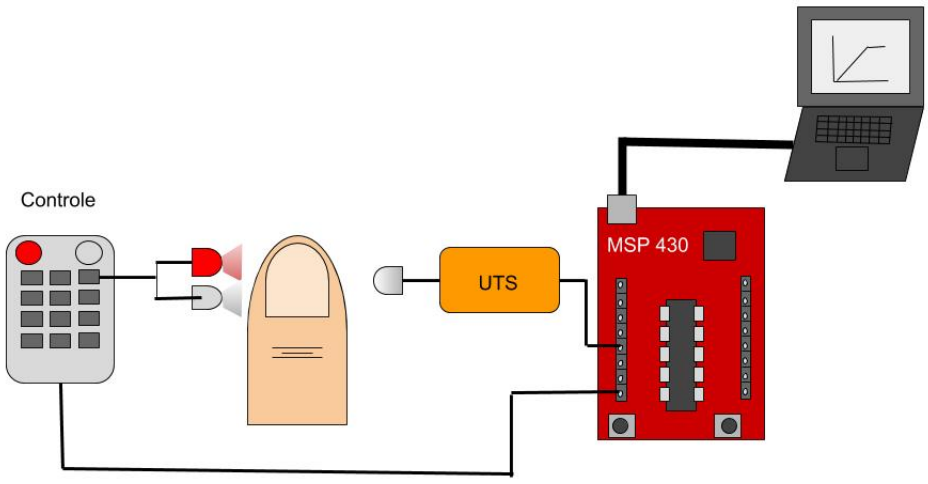
1.2 ESTRUTURA DO TRABALHO

Os temas abordados na introdução são aprofundados nos capítulos seguintes, passando por uma revisão bibliográfica levantada na fundamentação teórica, a qual aborda desde a área biológica, eletrônica, computação e comunicação. O desenvolvimento relata como foi todo o desdobramento do projeto, apresentando detalhadamente as etapas e decisões efetuadas. O capítulo de resultados são relacionados os experimentos realizados juntamente com as análises e implicações obtidas e validação. Por fim, na conclusão é apresentado o desfecho do projeto relacionando os objetivos com os resultados obtidos.

2 FUNDAMENTAÇÃO TEÓRICA

Os assuntos abordados pela fundamentação téorica estão relacionados a área médica, com o levantamento de informações relevantes que enfatizam a importância do cuidado do sistema cardiorrespiratório. Há também o aprofundamento dos conceitos técnicos que abrangem a construção do protótipo, como a unidade de tratamento de sinal (UTS), composta por: amplificação, filtros e conversão analógico-digital. Também são abordados os seguintes itens: sistema de aquisição de dados através do microcontrolador, envio de dados ao computador, utilização d*Amazon Web Service* e desenvolvimento da plataforma *mobile*. A figura 2.1 representa o projeto desenvolvido neste trabalho, o qual não visa desenvolver um oxímetro, mas sim trabalhar com o processamento do nível de oxigenação e frequência cardíaca.

FIGURA 2.1 – Esquemático do projeto a ser desenvolvido.



Fonte: Autoria própria, 2018

O sensoriamento do nível de oxigênio é feito através de dois *light emitting diode* (LED) de diferentes comprimentos de onda, um no espectro do vermelho e o outro localizado na faixa do infravermelho, o controle de potência é relizado pelo *driver* de controle embarcado no sensor, assim é possível garantir que a luminosidade emitida seja adequada. A luz é recebida pelo fotodiodo que converte o sinal luminoso em corrente elétrica, que se encaminha à UTS do sensor para conversão em tensão, filtragem e amostragem. O microcontrolador é responsável pelo tratamento e envio

das informações ao sistema embarcado que utilizará sistemas de armazenamento em nuvem. O primeiro tema abordado pela fundamentação teórica está relacionado a oximetria, seguido pelas doenças respiratórias e cardiovasculares e os impactos que tais problemas causam na sociedade, com tais tópicos é possível sustentar a justificativa do estudo como forma de auxiliar no combate a tais enfermidades.

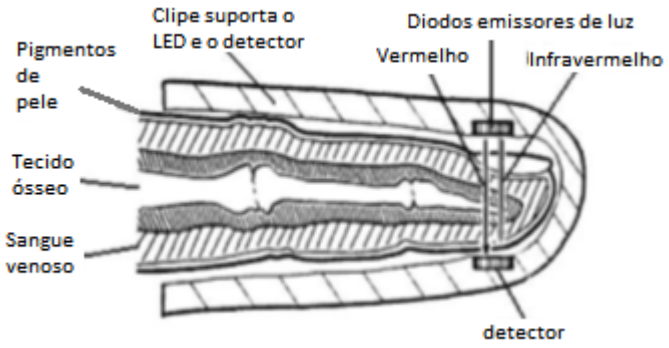
2.1 OXIMETRIA

A oximetria de pulso foi apresentada em 1983 como um método não invasivo para monitorar a saturação de oxigênio (SO2) presente no sangue (CLARK et al., 1997). Embora a leitura da saturação periférica de oxigênio (SpO2) nem sempre seja idêntica à leitura mais desejável de saturação arterial de oxigênio (SaO2) da gasometria arterial, os dois estão correlacionados para que o método seguro, e barato da oximetria de pulso seja utilizado para medir a saturação de oxigênio em uso clínico (HUCH, 1994).

Em seu modo de aplicação mais comum (transmissor), um dispositivo sensor é colocado em uma parte fina do corpo do paciente, geralmente uma ponta do dedo ou lóbulo da orelha, ou no caso de uma criança, através de um pé. O dispositivo emite luz em dois comprimentos de onda, o feixe luminoso percorrerá os diferentes tecidos que compõem o membro sensoriado até ser recebido pelo fotodetector, comumente um fotodiodo (CLARK et al., 1997).

O parâmetro adquirido com a técnica da oximetria é a variação da absorbância em cada um dos comprimentos de onda, permitindo determinar a absorbância devido apenas ao sangue arterial pulsante, excluindo sangue venoso, pele, osso, músculo, gordura, como ilustrado pela figura 2.2 (HUCH, 1994).

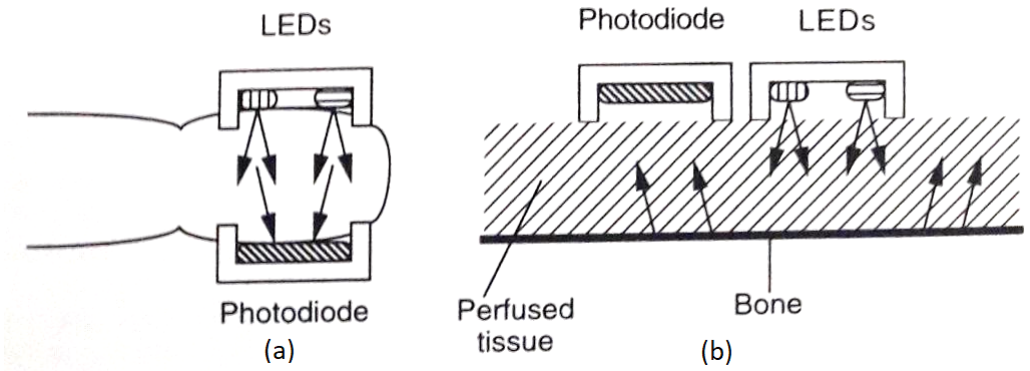
FIGURA 2.2 – Sensoriamento de oximetria.



Fonte: Alterado de (ANDRADE, 2009)

Menos comumente, a oximetria de pulso de reflectância é usada como uma alternativa ao oxímetro de pulso transmissivo supracitado. Este método não requer uma seção delgada do corpo da pessoa e, portanto, é adequado para uma aplicação universal, como pés, testa e peito, mas também tem algumas limitações. A vasodilatação e o acúmulo de sangue venoso na cabeça devido ao comprometimento do retorno venoso ao coração podem causar uma combinação de pulsações arteriais e venosas na região da testa e levar a resultados espúrios de SpO2. Tais condições ocorrem durante a anestesia com intubação endotraqueal e ventilação mecânica ou em pacientes em posição de Trendelenburg (HUCH, 1994). A figura 2.3 ilustra o método da refletância e o transmissivo.

FIGURA 2.3 – Sensoriamento de oximetria.



Fonte: (CLARK et al., 1997)

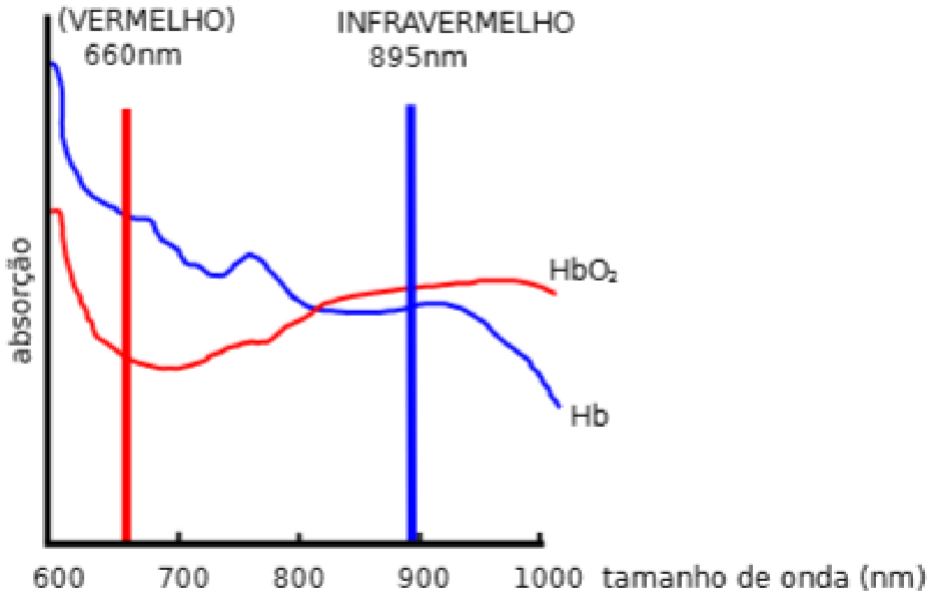
Na figura 2.3 são apresentados os dois métodos básicos da distribuição dos sensores para aquisição de dados relacionados a oximetria. O item (a) da figura 2.3 representa a oximetria transmissiva enquanto o (b) relaciona a refletiva.

2.1.1 Funcionamento

Um monitor de oxigênio no sangue exibe a porcentagem de sangue carregada de oxigênio, mais especificamente mede a porcentagem entre as hemoglobinas que não carregam oxigênio, também chamadas de hemoglobina reduzida parar a hemoglobina portadora de oxigênio, chamada de oxihemoglobina. Intervalos normais aceitáveis para pacientes sem patologia pulmonar são de 95 a 99 por cento, de forma que para um paciente respirando em ambiente no nível do mar ou próximo, há uma estimativa de que a pressão arterial pode ser feita a partir da leitura da saturação de oxigênio periférico (SpO2) do monitor de oxigênio no sangue (UK, 2004).

A medição de saturação é feita utilizando a luminosidade recebida pelo fotodiodo em cada comprimento de onda, 660nm do vermelho e 895nm do infravermelho, a oxihemoglobina absove mais a luz infravermelha do que a vermelha e hemoglobina reduzida tem comportamento contrário, assim é possível notar uma relação entre a concentração de oxigênio e absorbância da luz, conforme indicado pela figura 2.4.

FIGURA 2.4 – Curva de absorbância.



Fonte: (ANDRADE, 2009)

O fenômeno indicado pelo gráfico da figura 2.4 juntamente com a lei da absorbância de Beer-Lambert que relaciona o comprimento de onda de um feixe luminoso, a tamanho do dedo e a concentração de partículas que atenuam a luz é possível adquirir a concentração de oxigênio no sangue Os LEDs passam por um ciclo de operação de aproximadamente 30 Hz, o que permite ao fotodiodo responder à luz vermelha e infravermelha separadamente e também ajustar a linha de base da luz ambiente.

A quantidade de luz que é transmitida (em outras palavras, que não é absorvida) é medida, e sinais normalizados separados são produzidos para cada comprimento de onda. Esses sinais flutuam no tempo porque a quantidade de sangue arterial presente aumenta (literalmente pulsos) a cada batimento cardíaco. Ao subtrair a luz mínima transmitida da luz de pico transmitida em cada comprimento de onda, os efeitos de outros tecidos são corrigidos. A razão entre a medição da luz vermelha e a da luz infravermelha é então calculada pelo processador (que representa a razão entre a

hemoglobina oxigenada e a hemoglobina desoxigenada) e essa relação é convertida em SpO2 pelo processador por meio de uma tabela de consulta na lei Beer-Lambert (OXIMETRY, 2002).

2.1.2 Importância

A utilização da oximetria indica as taxas de saturação de oxigênio no sangue, o nível recomendado é de no mínimo 89% para a maioria das pessoas, baixas taxas ocasionam sobrecarregamento do coração e cérebro, assim o oximetro pode ser um inidcador de problemas cardiorespiratórios auxiliando na combate precoce às doenças cardiorespiratórias (AUGUSTO; MACHADO, 2018). Apesar de o modelo Van Vaugt ser o mais utilizado como ferramenta para efetuar diagnóstico de pneumonia, há evidências em estudo feito no Reino Unido que os dados oximétricos podem aumentar a eficiência de diagnósticos primário de pneumonia (GONDOLFO, 2018).

2.2 SAÚDE

2.2.1 Doenças respiratórias

As doenças respiratórias são enfermidades que estão no grupo de doenças que mais matam pessoa no mundo devido à asma, doença pulmonar obstrutiva crônica, infecções respiratórias agudas, tuberculose e câncer de pulmão. Os níveis de incidência dessa categoria de patologia são altos e há cetenas de milhões de pessoas que sofrem dessa enfermidade em escala mundial, bem como cerca de 4 milhões de mortes prematuras a cada ano. São números expressivos mas que podem ser menores se ocorrer medidas preventivas como reduação a exposição à poluição do ar no ambiente, e mudança de hábitos com o desuso do tabaco (SOCIETY, 2013).

A tabela 1 evidencia centenas de milhões de pessoas que estão sofrendo de alguma enfermidade respiratória. Cerca de 4 milhões de pessoas morrem anualmente devido infecção respiratória, mais comuns em crianças. Não menos importante, há a doença respiratória crônica que é a quarta principal causa de morte no mundo, devido a diversos fatores, inclusive em consideração ao nível de poluição mundial. A tuberculose também é uma patologia que afeta gravemente o mundo principalmente no continente africano, com as maiores taxas de mortalidade (SOCIETY, 2013).

TABELA 1 – Principais doenças respiratórias em escala global

Doença	Principais fatores de risco	Frequência global estimada de doenças (milhões)	Mortes globais por ano	Comentários
Doença respiratória crônica	Tabagismo, poluentes ao ar livre, asma	200 (prevalência)	Indisponível	Quarta causa principal das mortes no mundo
Asma	Predisposição genética, poluentes atmosféricos, fatores dietéticos, respostas imunológicas anormais	235 (prevalência)	0,18	Aumento da prevalência em todo o mundo
Infecções respiratórias	Baixas taxas de imunização, nutrição pobre, infecção por HIV	Insdisponível	4	A doença crônica mais comum das crianças
Tuberculose	Infecção por HIV	8,7	1,4	Cerca de 80% dos casos globais de tuberculose e HIV são provenientes da África
Câncer de pulmão	Tabagismo, exposição passiva a fumaça de cigarro, poluentes provenientes da queima de combustível	1,6	1,37	Tipo de câncer mais comum

Fonte: American Thoracic Society

Os números de mortalidade por doença respiratória podem ser reduzidos com novas ações, porém milhões de pessoas morrem devido à falta de acesso a imunizações, medicamentos ou a incapacidade do sistema de saúde para fornecer cuidados. Desse modo, cada vez mais há programas de prevenção de doenças contagiosas, os quais tem reduzido o número de transmissões. Assim com o aumento do controle de doenças respiratórias infecciosas, os esforços mudaram para doenças não transmissíveis como por exemplo a asma. A asma atinge cerca de 235 milhões de pessoas em todo o mundo, sendo uma doença que afeta todas a todos (SOCIETY, 2013).

2.2.2 Doenças cardiovasculares

Estima-se que as doenças cardiovasculares (DCV) são responsáveis por aproximadamente 17,7 milhões de mortes no ano de 2015, de forma a representar 31% de todas as mortes globais. Desses óbitos, por volta de 6,7 milhões ocorreram devido a acidentes vasculares cerebrais (AVCs). Fatores socioeconômicos estão diretamente ligados as altas taxas de vitimidade causada pela doença, visto que mais de três quartos das mortes relatadas são apontadas em países de baixa renda (OPAS, 2017).

As doenças supracitadas são manifestadas através de ataques cardíacos, derrames, uso do tabaco, sedentarismo e o uso nocivo de álcool (OMS, 2018).

A hereditariedade e má qualidade de vida combinadas com o excesso do uso de álcool e tabaco são grandes indicadores para aumentar o nível da pressão arterial, que por sua vez, gera uma reação em cadeia para propiciar doenças cardiovasculares. No momento em que a pressão sanguínea se eleva ocorre o aumento da glicose no sangue e o sobrepeso que são prejudiciais à boa saúde do coração (OMS, 2018).

No Brasil, a doença cardiovascular tem sido a principal causa de mortalidade desde a década de 1960 e maior responsável por uma porcentagem substancial de todas as hospitalizações. Em 2011, as DCV foram responsáveis por 31% de todas as mortes, sendo as doenças isquêmicas do coração (31%) e as doenças cerebrovasculares (30%) as principais causas de tal enfermidade.

Apesar do aumento no número total de mortes por DCV, as taxas de mortalidade ajustadas por idade diminuíram 24% entre 2000 e 2011. Os cuidados de saúde prestados pelo sistema de saúde pública do Brasil, que se concentra na prevenção primária, contribuíram para essa conquista, no entanto, a mortalidade difere de acordo com raça, sexo e status socioeconômico conforme a tabela 2, sendo os indivíduos negros e populações de baixa renda os maiores impactados. fato que se deve ao Brasil ter desigualdades sociais marcantes. A população é formada por uma mistura de brasileiros nativos, europeus e africanos, que enfrenta o envelhecimento rápido. As tendências temporais relacionadas aos comportamentos de saúde mostram uma redução substancial nas taxas de tabagismo, mas uma prevalência crescente de sobrepeso e obesidade, dieta pobre e sedentarismo. A alta prevalência de hipertensão e o aumento de diabetes também são motivo de preocupação. (CIRCULATION, 2016).

TABELA 2 – Doenças cardiovasculares.

	Todas as doenças cardiovasculares		Derrame e hipertensão		DAC	
	N	%	N	%	N	%
Todos indivíduos	135,893	40,8	52,558	36,1	49,625	47,5
Sexo						
Homens	82,434	47,3	29,477	41,1	33,087	54,5
Mulheres	53,449	33,7	23,074	31,3	16,535	37,9
Raça						
Brancos	64,102	36	22,496	31,2	25,523	43,3
Miscigenados	50,483	46,6	20,55	40,6	17,531	53,7
Negros	14,184	50,1	6,414	45,8	4,307	56,9
Asiáticos	511	26,1	207	24,5	190	31,2
Ameríndios	198	38,2	91	33,7	59	44,4

Fonte: NCBI Pubmed, 2013

2.2.3 Impacto econômico das doenças cardiovasculares

As doenças cardiovasculares são responsáveis por grande parte dos casos de morte no planeta na atualidade, em contrapartida também possui um alto indicador no número de ocorrências para tratamentos e medidas preventivas. Dessa forma, para entender o tamanho do problema é necessário saber suas dimensões e custos, assim é imprescindível analisar os custos inerentes que o estado brasileiro deve arcar todos os anos para o combate desses problemas. Gastos com hospitalizações, atendimentos ambulatoriais, medicamentos e serviços realacionados ao tratamento são formas de estipular o tamanho do impacto socieconômico causados no Brasil.

No ano de 2015 foi estimado que o governo brasileiro destinou cerca de 9,5% do PIB para o setor da saúde, sendo que desses 9,5%, cerca de 0,7% foi orientado para doenças cardiovasculares. Consequentemente, os custos estimados relacionados a doenças cardiovasculares para esse ano foram de R$31,7 bilhões de reais para o Brasil, apontando um aumento de aproximadente 17% do período de 2010 a 2015. Desses 31,7 bilhões, 61% são custos relacionados a morte prematura do indivíduo, situação onde ocorre o óbito devido a acidentes cardiovasculares em uma idade inferior a 70 anos. Não menos importante, foi situado que 22% são custos relacionados a internações e consultas médicas, 15% por custos pela perda de produtividade devido a doenças e 2% para gastos diversos (SIQUEIRA; SIQUEIRA-FILHO; LAND, 2017).

2.2.4 Importância de medição de parâmetros vitais

Para avaliar o estado de saúde de uma pessoa é importante realizar as medições de sinais vitais do paciente, que são parâmetros que indicam como está o funcionamento do sistema circulatório e respiratório. Como esses sinais indicam se a pessoa apresenta níveis normais ou se há alguma anomalia com o paciente, a medição desses parâmetros contribui efetivamente com o trabalho da equipe médica auxiliando em um diagnóstico mais assertivo na tomada de decisão para realizar um tratamento que seja o mais adequado ao paciente(TEIXEIRA, 2015).

Quando a medição de parâmetros vitais é focada para pacientes idosos, é necessário ter um maior grau de atenção para essa faixa etária, visto que há uma elevada sensibilidade na alteração desses parâmetros devido ao envelhecimento que deixa a saúde mais precária. Assim, é de fundamental importância estar sempre

monitorando parâmetros como pressão arterial, frequência respiratória e temperatura para evitar surpresas desagradáveis (TEIXEIRA, 2015).

A medição da pressão arterial pode ser realizada com a utilização de um aparelho medidor de pressão digital ou com um aferidor de pressão analógico esfigno-manometro combinado com a utilização de e um estetoscópio para amplificar o som durante a medição. A pressão arterial é considerada relativamente normal quando a pressão sistólica ou máxima está abaixo de 130 mmHg e a pressão diastólica ou mínima está abaixo de 85 mmHg. No entanto, para ter uma medição confiável de pressão arterial, outros parâmetros como temperatura e frequência cardíaca devem estar normalizados. A frequência cardíaca é responsável por indicar o número de vezes que o coração bate por minuto, sendo que geralmente o valor normal para um adulto é de 60 a 100 batimentos por minutos, enquanto que a temperatura é considerada normal quando se encontra na faixa de 36º a 37ºC (TEIXEIRA, 2015).

Durante o envelhecimento os parâmetros vitais tendem a ficar mais suscetível a variações, o que torna imprescindível ter um constante monitoramento desses fatores para evitar surpresas e complicações a saúde. Assim, a identificação antecipada de anomalias dos parâmetros vitais é fundamental para aumentar consideravelmente as chances de cura caso seja identificado algum problema e impactando diretamente na redução de mortes precoces(TEIXEIRA, 2015).

Outro grande problema para saúde com relação aos sinais vitais é a falha humana da equipe de enfermagem durante o registro dos valores medidos, pois os enfermeiros estão passíveis a se equivocar ao anotar os valores de medição no prontuário dos pacientes. A negligência na triagem de pacientes pode interferir efetivamente na assertividade do diagnóstico, visto que as demais pessoas da equipe médica utilizam os dados coletados para embasar seu trabalho. Por isso é necessário sempre conferir se todos os dados de um prontuário estão preenchidos corretamente a fim de evitar possíveis danos aos pacientes (TEIXEIRA, 2015).

Portanto, compreender os parâmetros relacionados à saúde e os fatores que influenciam esses parâmetros é de fundamental importância para identificar e solucionar rapidamente possíveis complicações em decorrência das alterações na saúde das pessoas em virtude do envelhecimento (TEIXEIRA, 2015).

2.2.5 Sinais vitais do organismo

Um excelente método para verificar como está à saúde do organismo é por meio da obtenção dos sinais vitais de uma pessoa, visto que são importantes parâmetros que auxiliam na avaliação da saúde. Como método preventivo de doenças, são verificados os sinais de pressão arterial, temperatura corporal e frequência cardíaca.

2.2.5.1 Pressão Arterial

A pressão arterial (PA) é a pressão que o sangue exerce contra a superfície das veias e artérias devido ao bombeamento sanguíneo exercido pelo funcionamento do coração sendo expressa em uma escala de valores mínimos e máximos. O valor máximo é denominado como pressão sistólica, enquanto que o mínimo como pressão diastólica sendo expressa pela unidade de medida de milímetros de mercúrio. O seu valor aceitável de condições normais para um adulto gira em torno de 120 mmHg para pressão sistólica ou máxima, enquanto que para a pressão diastólica ou mínima deve ser em torno de 85mmHg (OGEDEGBE; PICKERING, 2013).

Quando se iniciou a aferição de pressão arterial, costumava-se se utilizar um aparelho analógico esfigmomanômetro combinada com a utilização de um estetoscópio, porém, com a evolução da tecnologia há aparelhos digitais mais precisos e menos invasivos o que proporciona um maior grau de conforto ao paciente. No entanto, como a pressão arterial influencia outros indicadores, ela também é influencia por situações de estado emocional do paciente, condição de estado físico e também a saúde de outras partes do organismo como o coração (OGEDEGBE; PICKERING, 2013).

2.2.5.2 Frequência Cardíaca

A frequência cardíaca é medida pelo número de contrações do coração por minuto, ou batimentos por minuto (bpm) podendo variar de acordo com as necessidades físicas do corpo, incluindo a necessidade de absorver oxigênio e excretar dióxido de carbono. Geralmente é igual ou próximo do pulso medido em qualquer ponto periférico. Atividades que podem provocar mudanças incluem exercícios físicos, sono, ansiedade, estresse, doenças e consumo de drogas (ASSOCIATION, 2018).

Indica-se que a frequência cardíaca normal em adultos humanos em repouso é entre 50 e 90 bpm, embora a *American Heart Association* (AHA) afirme que um adulto

em repouso normal possui a frequência cardíaca de 60 a 100 bpm. Sendo que durante o sono, um batimento cardíaco lento com taxas em torno de 40 a 50 bpm é comum e é considerado normal. No entanto, anomalias da frequência cardíaca, por vezes, indicam doenças como taquicardia, bradicardia e arritimia (ASSOCIATION, 2018).

Há muitas anomalias relacionados a frequência cardíaca e uma delas é o fenômeno taquicardia que se trata de uma frequência cardíaca acelerada, sendo definida como acima de 100 bpm em repouso. Por outro lado, há outro fenômeno chamado de bradicardia, que é uma frequência cardíaca baixa, sendo definida como abaixo de 60 bpm em repouso. Já a arritimia, trata-se de uma alteração em que o o coração não está batendo em um padrão regular (ASSOCIATION, 2018).

2.2.5.3 Temperatura corporal

A temperatura corporal média que é considerada normal para um ser humano é de aproximadamente 37 ℃ mas pode ter variações de acordo com o indivíduo. Porém, durante o dia a temperatura de cada pessoa sofre variação de até 0,5 ℃ entre o mínimo e máximo (WALKER; HALL; HURST, 1990).

Estima-se que a o pico de temperatura para o indivíduo normal seja por volta das 18 horas enquanto que o mínimo seja por volta das 04 horas. No entanto, este padrão de temperatura pode ser alterado se o indivíduo está submetido a condições de troca de período do sono, ou seja, dormir de dia e trabalhar a noite, visto que afetará uma correção adaptativa no rítmo da temperatura.

Para fins clínicos práticos, um paciente é considerado febril se a temperatura exceder 37,5 ℃, sendo que os extremos de temperatura como quando a temperatura excede 41,1 ℃ é chamado de hiperpirexia e para temperaturas inferiores a 35 ℃ se chamada hipotermia (WALKER; HALL; HURST, 1990).

2.3 ELETRÔNICA

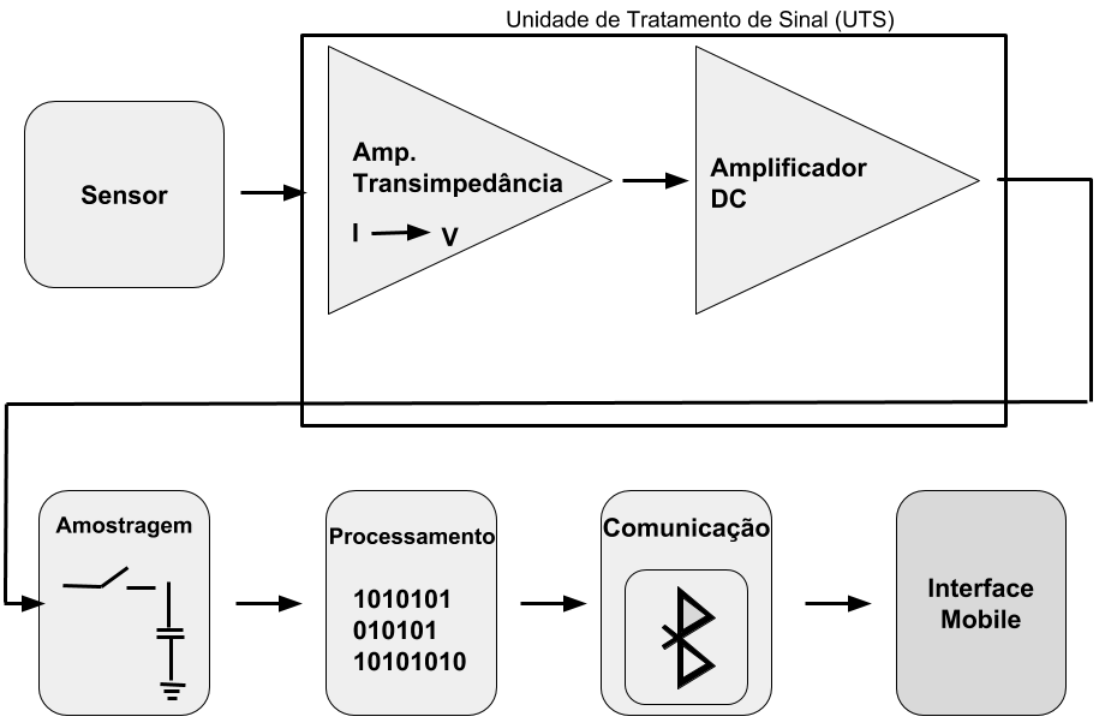
2.3.1 Diagrama de blocos

A figura 2.5 mostra o diagrama de blocos elaborado para organização e execução das tarefas. A primeira etapa do projeto está relacionada a aquisição do sensor oximétrico, devido a dificuldade de construção os integrantes da equipe utilizarão um sensor pronto modelo SD01-BC do fabricante RDM. O segundo bloco representa a

UTS formada por 2 estágios de amplicação e um de filtragem, o principal objetivo de tal bloco funcional é permitir que o sinal adquirido através do sensor receba o tratamento adequado de forma que possa ser manipulado. Os dados se encaminham para o microcontrolador que será capaz de realizar a conversão analógico-digital, realizar um simples tratamento digital do sinal e encaminha-lo ao computador.

Após o devido tratamento dos sinais e aquisição será possível desenvolver a interface *mobile* que criará a interação com o usuário final. A interface será responsável por apresentar gráficos relacionados ao batimento cardiáco e oxigenação sanguínea para que possam ser visualizados por cuidadores e até mesmo por médicos que poderão utilizar como ferramenta de consulta para avaliar o desenvolvimento da enfermidade e como o paciente está reagindo ao tratamento.

FIGURA 2.5 – Diagrama de blocos do projeto a ser desenvolvido.



Fonte: Autoria própria, 2018

2.3.2 Sensoriamento

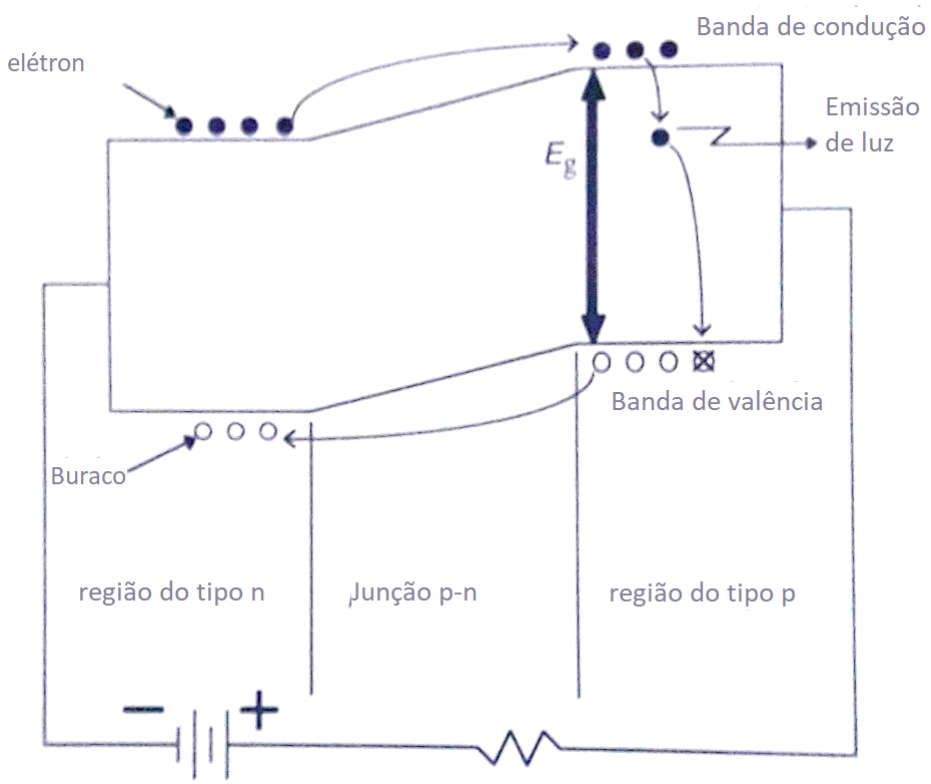
O primeiro bloco funcional é composto do sensoriamento, e apresenta a função de adquirir os dados do mundo externo. Para o caso da oximetria o sensoriamento é

composto por dois componentes básicos, o *light emitting diode* (LED) e o fotodiodo.

Os LEDs são dispositivos optoeletrônicos semicondutores capazes de emitirem luz (CLARK et al., 1997). O funcionamento dos LEDs pode ser descrito pela física de semicondutores, pois no momento em que um elétron deixa sua banda de valência e ocupa a banda de transição há o consumo de energia para que tal tarefa ocorra, nesse momento o semicondutor é capaz de conduzir corrente elétrica (CLARK et al., 1997).

Ao retornar a sua banda de valência, a energia que foi recebida para se locomover a banda de condução é dissipada através de fótons (CLARK et al., 1997) como ilustrado pela figura 2.6.

FIGURA 2.6 – Efeito fotoelétrico



Fonte: (CLARK et al., 1997)

O comprimento de luz emitido pelos LEDs é dependente do material que o compõe. Diferentes materiais apresentam diferentes *gaps* de energia entre os estados de valência e condução. A equação 2.1 apresenta a energia necessária para que ocorra o movimento entre as bandas.

$$Eg = hc/\lambda \tag{2.1}$$

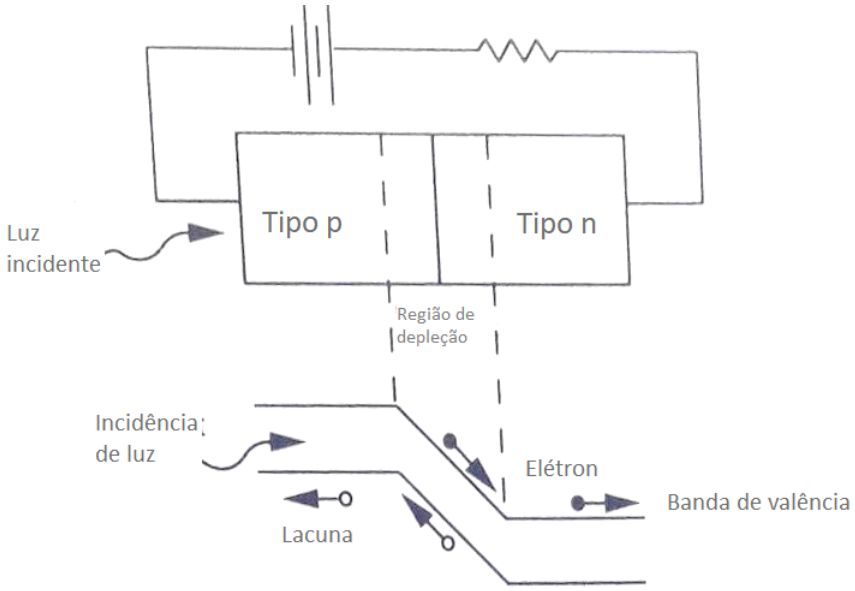
A constante h é conhecida como constante de Planck e seu valor é universal, c é a velocidade da luz no vacúo e o comprimento de onda da luz é dado por λ.

Conhecendo a energia entre as bandas do material é possível selecionar qual o comprimento de onda será emitido pelo LED. Logo isolando o λ da equação 2.1 obtém-se o comprimento de onda, como mostrado na equação 2.2.

$$\lambda = hc/Eg \tag{2.2}$$

O fotodiodo utiliza um conceito similar formulado pelo físico alemão Albert Eintein, o efeito fotoelétrico. Tal efeito é descrito pela incidência de fótons em superfícies metálicas o que resulta no desprendimento de elétrons do átomo, gerando corrente elétrica. A figura 2.7 mostra a dinâmica do efeito.

FIGURA 2.7 – Funcionamento do fotodiodo.



Fonte: (CLARK et al., 1997)

2.3.3 Unidade de tratamento de sinal

Os itens 2.8.1, 2.8.2 e 2.8.3 abordão respectivamente o amplificador de transimpedância, o filtro passa baixa e o amplificador do estágio final.

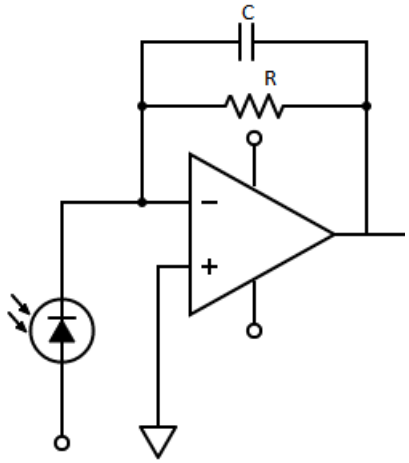
Todos os três itens supracitados compõem a unidade de tratamento de sinal.

2.3.3.1 Amplificador de transimpedância

O circuito de transimpedância, realiza a operação de derivada na corrente do fotodiodo, tornando a tensão de saída uma função da corrente de entrada. A figura 2.8 expressa a topologia do amplificador diferenciador.

Algumas boas práticas de desenvolvimento do circuito de transimpedância foram reunidas por Schowalter (1997). A primeira afirma que a capacitância de junção do fotodiodo deve ser a menor possível. A segunda alega que a área sensível a luminosidade do fotodiodo deve ser a menor possível, para aumentar a relação sinal-ruído. Já a terceira afirma que o resistor utilizado deve apresentar o maior valor possível para minimizar o ruído.

FIGURA 2.8 – Amplificador de transimpedância.



Fonte: (SEDRA ADEL; SMITH, 1991)

A função de transferência do amplificador de transimpedância é dado por:

$$V_{out} = \frac{RI_{in}}{sC} \tag{2.3}$$

E por último a escolha do amplificador deve priorizar o tipo FET pois as correntes de polarização são baixas o que resuta no aumento da sensibilidade do sistema.

2.3.3.2 Filtros Passa-baixa

Os filtros são circuitos capazes de selecionar os sinais de acordo com suas frequências, sendo que utilizam componentes reativos como capacitores e indutores ou ativos como os amplificadores.

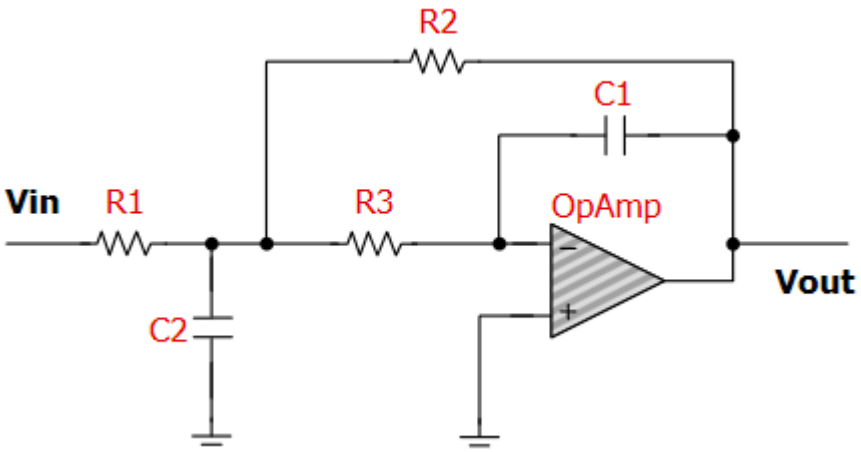
Os filtros podem ser classificados quanto a sua reposta em frequência, os mais simples são o passa-alta, passa-faixa, rejeita-faixa e o passa-baixa, o qual possui característica seletoras de baixa frequência (BONFIM, 2010).

Cada filtro possui uma funcão de transferência caracteristica, com a qual ocorre a análise matemática de atenuação, defasamento, pólos e zeros. O sistema é classificado de acordo com sua ordem e sua topologia.

Os projetos de filtros são baseados em estruturas predefinidas para implementar uma determinada função de resposta. Deste modo, uma estrutura comum e eficiente é a topologia *Multiple-Feedback*. Essa topologia apresenta um ganho negativo porque o sinal de saída será invertido em relação à entrada.

Os resistores que dão ganho ao circuito não são isolados, de forma que a alteração no ganho altera a frequência de corte do filtro projetado. Consequentemente, utilizando essa topologia há uma redução no número de componentes utilizados conforme a figura 2.9 que ilustra a toplogia MFB (PERTENCE, 2003).

FIGURA 2.9 – Topologia MFB



Fonte: (PERTENCE, 2003)

Para esta topologia, o cálculo da frequência de corte é dado pela equação 2.4, enquanto que o ganho é dado pela 2.5, e 'R2' pela 2.6 onde 'a' e 'b' são fatores de correção obtidos por tabela. E por recomendação empírica, 'C2' é dado pela equação 2.7 e 'C1' pela equação 2.10 e 'R3' pela equação 2.9 (PERTENCE, 2003).

$$fc = \frac{1}{2 * pi\sqrt{R2 * R3 * C1 * C2}} \tag{2.4}$$

$$K = -\frac{R2}{R1} \tag{2.5}$$

$$R2 = \frac{2 * (K+1)}{wc * [aC2 + \sqrt{a^2 * C2^2 - 4 * b * C1 * C2 * (K+1)}]} \tag{2.6}$$

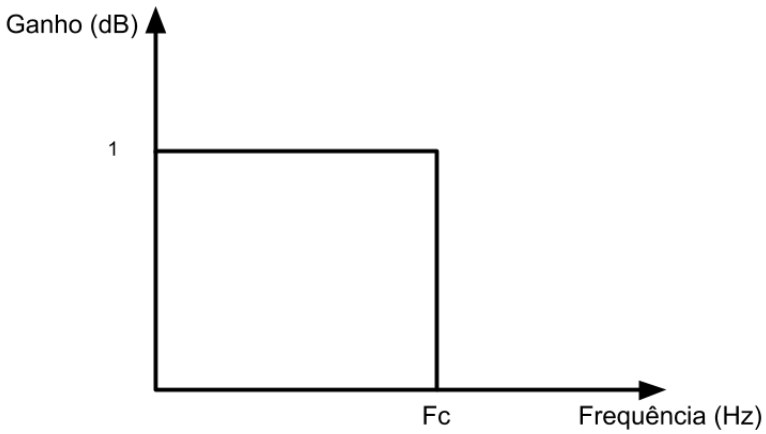
$$C2 = \frac{10}{fc} \tag{2.7}$$

$$C1 \leq \frac{a^2 * C2}{4 * b * (K+1)} \tag{2.8}$$

$$R3 = 1/(b * C1 * C2 * wc^2 * R2) \tag{2.9}$$

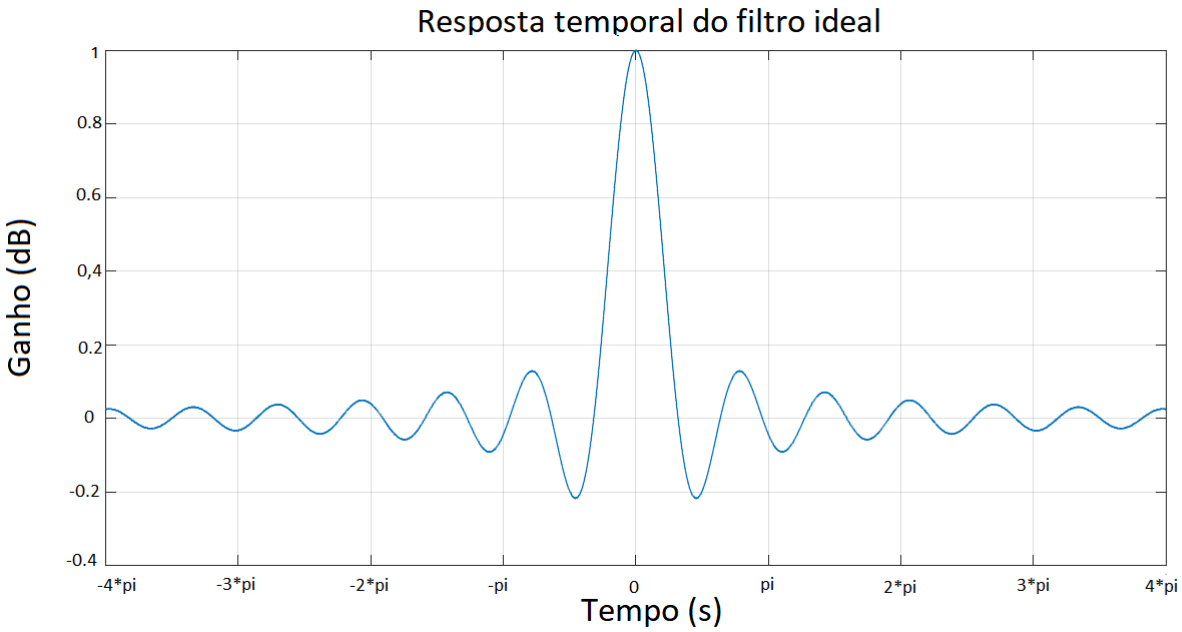
Sintetizar um filtro está relacionado com o desenvolvimento de um sistema capaz de produzir uma dada reposta conforme a excitação que recebe (FILHO, 2010). O filtro ideal pode ser representado na frequência pela forma retangular, como ilustrado pela figura 2.10. Ao aplicar a transformada inversa de Fourier a representação da resposta temporal do filtro é gerada, o sinc representado pela figura 2.11. Para que seja possível o filtro ideal seria necessário que a resposta temporal se extendesse ao tempo infinito, situação que não acontece na realidade. Assim, para que o projeto se torne viável é necessário limitar a resposta temporal, tornando a construção do filtro factível. Ao limitar no tempo, a resposta em frequência é alterada passando por distúrbios que necessitam ser levados em conta no projeto do sistema eletrônico.

FIGURA 2.10 – Resposta em frequência do filtro ideal.



Fonte: Autoria própria, 2018

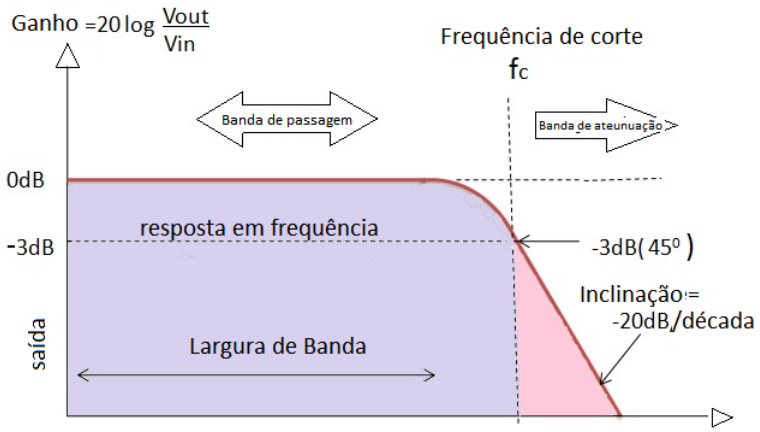
FIGURA 2.11 – Resposta temporal do filtro ideal.



Fonte: Autoria própria, 2018

Como pode ser visto na figura 2.12 é possível identificar as imperfeições que são inseridas a resposta espectral. O decaimento deixa de ser representado por uma reta vertical e dá lugar ao caimento de -20db por década, para cada ordem de grandeza do filtro o decaimento se torna 20 dB mais acidentado.

FIGURA 2.12 – Resposta espectral do filtro não ideal.



Fonte: https://www.electronicsrepairhouston.com/high-pass-filter-zero.html, 2018

A frequência fc, representada na figura 2.12 é utilizado como parâmetro para o desenvolvimento do filtro passa baixa, em tal frequência o filtro é caracterizado por atenuar o sinal de entrada em três decibeis. A cada década (multiplos de 10 da

frequêcia) a atenuação aumenta em 20 dB, tornando quase nula as componentes espectrais de alta frequência. Há outros parâmetros que de acordo com a aplicação do projeto são necessários levar em conta, como a defasagem causado pelo filtro e o atraso de grupo, defasagem variável conforme a frequência. Filtros digitais para aplicações em tempo real aproximam-se do filtro ideal, truncando e dando janelamento à resposta infinita do impulso para produzir uma resposta de impulso finita; aplicar esse filtro exige atrasar o sinal por um período de tempo moderado, permitindo que o cálculo preditivo se antecipe. Este atraso se manifesta como mudança de fase.

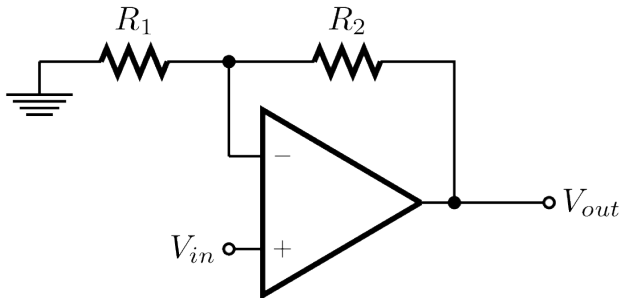
2.3.3.3 Amplificador de estágio final

Um amplificador operacional é um circuito com alta impedância de entrada e baixa impedância de saída, e que possui diversas aplicações.

O objetivo no uso do amplificador de estágio final é providenciar ajustes finais ao sinal já filtrado para os níveis de tensão que o sistema de conversão analógico digital opera. A figura 2.13 mostra o amplificador de estágio final, no caso está sendo utilizado um amplificador não-inversor. O amplificador não inversor tem como característica um ganho que é dado pela equação 2.10.

$$Av = \frac{Vo}{Vi} = \frac{R2}{R1} + 1 \tag{2.10}$$

FIGURA 2.13 – Amplificador não inversor.



Fonte: (SEDRA ADEL; SMITH, 1991)

2.4 SISTEMA COMUNICAÇÃO *MOBILE*

2.4.1 *Bluetooth*

O *Bluetooth* é um padrão de tecnologia sem fio para troca de dados em distâncias curtas (usando ondas de rádio UHF de comprimento de onda curto na faixa ISM de 2,4 a 2,485 GHz) de dispositivos fixos e móveis e construção de redes de área pessoal, ou *networking area personal* (PAN).

A tecnologia inventada pelo fornecedor de telecomunicações Ericsson em 1994, foi originalmente concebida como uma alternativa sem fio para cabos de dados RS-232. Em outras palavras, o *Bluetooth* é uma maneira de os dispositivos se comunicarem uns com os outros, sem fio e em curtas distâncias (HUANG, 2005).

O *Bluetooth* é gerenciado pelo Grupo de Interesse Especial *Bluetooth* (SIG), que tem mais de 30.000 empresas associadas nas áreas de telecomunicações, computação, redes e eletroeletrônicos. O IEEE o padronizou como IEEE 802.15.1, mas não mantém mais o padrão. O *Bluetooth* SIG supervisiona o desenvolvimento da especificação, gerencia o programa de qualificação e protege as marcas registradas. Um fabricante deve atender aos padrões SIG para comercializá-lo como um dispositivo *Bluetooth*. Uma rede de patentes se aplica à tecnologia, que é licenciada para dispositivos qualificados individuais (HUANG, 2005).

O desenvolvimento de aplicativos que fazem uso da comunicação é simples e fácil, embora possa parecer difícil devido ao seu alcance incomumente amplo (HUANG, 2005). Para trabalhar com a funcionalidade o programador necessita conhecer uma pequena fração do que está previsto nas especificações. Não obstante, o conhecimento mais profundo da plataforma auxilia em sua implementação, mas não é necessário entrar nas especificidades dos algoritmos dos dispositivos *Bluetooth* (HUANG, 2005).

2.4.2 Aplicativo *Android*

A plataforma *Android* é baseada no sistema Linux e atualmente é mantida pela google. O sistema é empregado principalmente em *smartphones* e devido ao alcance mundial se tornou um dos maiores ecossistemas vigentes para desenvolvimento de aplicações. Cada aplicativo é reconhecido como um usuário, sendo assim múltiplos usuários estão utilizando recursos do sistema. Os processos são identificados por números de identificação (ID) e a execução ocorre sobre máquinas virtuais, isolando

eventuais falhas generalizadas (DEVELOPERS, 2018).

A fim de minimizar a interferência entre os processos executados pelo *hardware* o *Android* aplica o princípio do privilégio mínimo, em que os recursos alocados aos processos são os mínimos necessários, assim a disputa entre processos é mitigada. Os blocos básicos para a construção de aplicativos no ambiente são: *activity*, *services*, *content providers* e *transmission receptors*. As *activities* representam a telas de execução dos aplicativos, são basicamente as interfaces de usuário, seu ciclo de vida é representado pela Figura 2.14 e apresentam similaridade com as funções da linguagem C, pois são armazenadas em uma pilha do tipo FIFO (*First in First out*) à medida que são fechadas, ao tocar no botão *return* do *smartphone*, ocorre a saída da pilha e a destruição do processo pelo sistema operacional (DEVELOPERS, 2018).

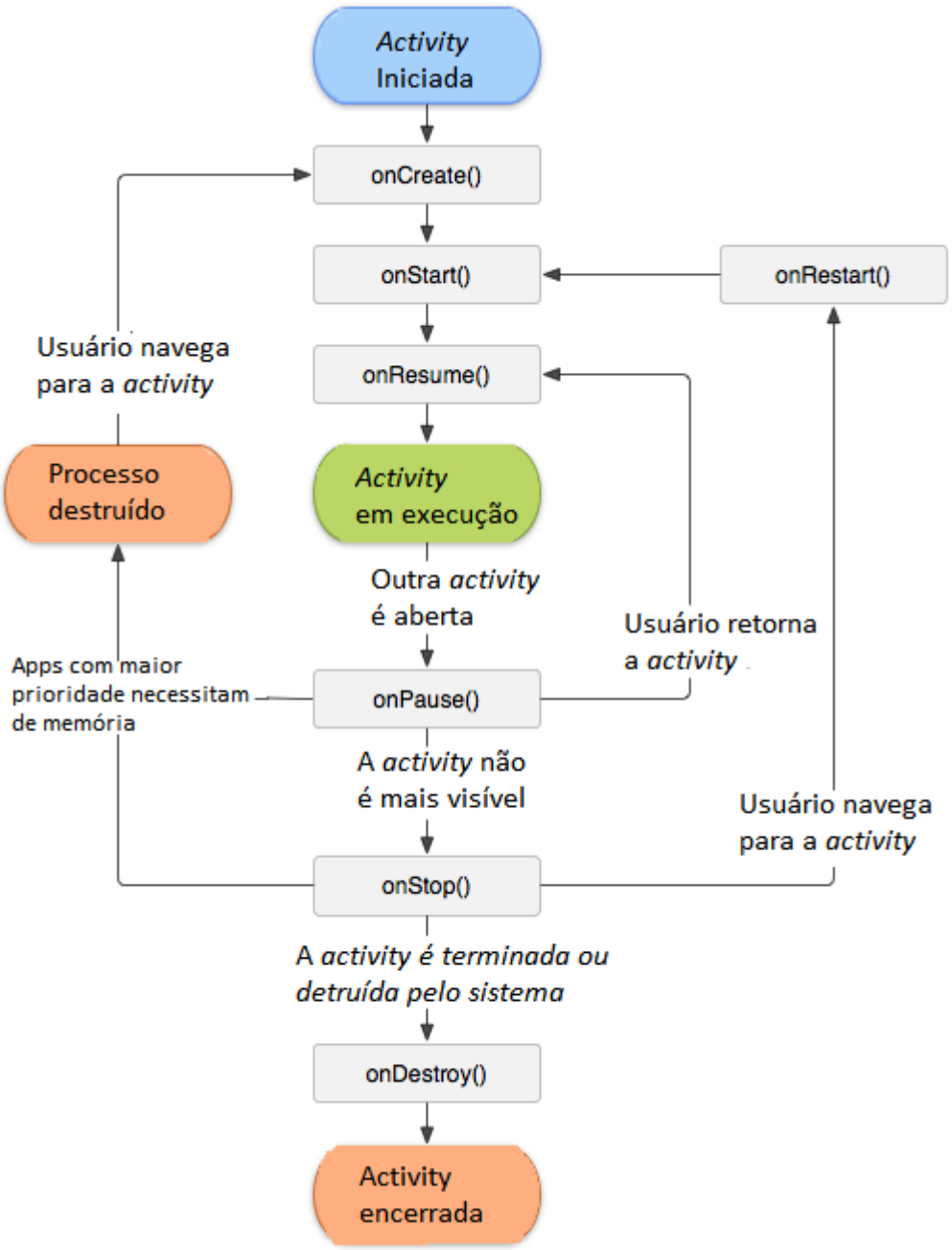
A criação é realizada através do método *onCreate()* etapa que deve ser realizada as configurações básicas da *activity*, seguido pelo *onStart()* que realiza a preparação para o próximo método. O *onResume()* é utilizado em casos que a execução foi para o *background*, o campo verde da Figura 2.14 é o momento em que a *activity* está em execução no plano principal do usuário, ao deixar o plano principal o *onPause()* é executado, caso a *activity* saia da tela irá para o *onStop()* e se o usuário encerrar a execução a *activity* será destruída pelo *onDestroy()* (DEVELOPERS, 2018).

Os *services* são caracterizados por processos que rodam ao fundo sem que o usuário o veja. Os *content providers* são responsáveis pelo gerenciamento de conteúdo compartilhado, enquanto que os *transmission receptors* lidam com as notificações do sistema (DEVELOPERS, 2018).

A vantagem do sistema operacional é a reutilização de ferramentas já criadas, como aplicativos que utilizam a câmera não necessitam escrever seu próprio algoritmo de gerenciamento da câmera basta acessar os recursos do sistema, tal característica é possível devido ao uso da tecnologia de *intents* que se resumem a chamadas assíncronas ao sistema indicando a necessidade do uso de recursos, caso o aplicativo tenha a permissão necessária o sistema fará a ponte de comunicação entre os processos de diferentes apps (DEVELOPERS, 2018).

Dos quatro blocos de construção básicos a *intent* realiza a chamada de três deles: a *activity*, o *service* e o *transmission receptor*. O *content provider* só é chamado através de *querys* em objetos do tipo *content resolver* (DEVELOPERS, 2018).

FIGURA 2.14 – Ciclo de vida da *activity*



Fonte: Alterado de Android developers.

As formas de ocorrerem a iniciação dos blocos são representadas pelos tópicos abaixo:

• *Activity*: Passar uma *intent* ao método *startActivity()* ou *startActivityForResult()*;

• *service*: Passar uma *intent* ao método *startService()* ou vincula-lo através da intente no método *bindService()*

• *transmission*: Passar uma *intent* aos métodos *sendBroadcast()*, *sendOrde-*

redBroadcast() ou *sendStickyBroadcast()*;

• *provider*: Consulta feita através da *query()* em um *content solver*.

O uso das *intents* é classificado como implícito ou explícito, caso o componente seja conhecido é passado como um dos parâmetros, tornando-a explícita, de forma contrária, o componente que executará a *intent* será definido pelo sistema operacional, tornando-a implícita, é comum que mais de um aplicativo seja capaz de lidar com a tarefa nesta ocasião a escolha será definida pelo usuário. Todo recurso, permissões, bibliotecas e APIs devem ser definidas no arquivo *manifest*, presente no diretório raiz do aplicativo, com exceção do *transmission receptor* que é declarado dinamicamente em tempo de execução (DEVELOPERS, 2018).

Além de codificação e componentes do ambiente, os aplicativos do sistema *Android* possibilitam a utilização de recursos exteriores, como imagens, áudios, leiautes, menus e animações o que tornam a experiência de usuário mais rica.

2.4.3 JAVA

Java é uma linguagem de programação e plataforma de computação lançada pela *Sun Microsystems* em 1995. Há muitos aplicativos e sites que não funcionarão a menos que se tenha o Java instalado. A plataforma é rápida, segura e confiável, desde de *laptops* até *datacenters*, consoles de jogos, supercomputadores científicos, celulares à Internet, ou sejam, o Java está em toda parte. Hoje, com a tecnologia como parte da vida cotidiana, a população pode estar conectada e acessar aplicativos e conteúdo em qualquer lugar, a qualquer hora. Por causa do Java, espera-se que os dispositivos digitais sejam mais inteligentes e mais funcionais (JAVA, 2000).

No início dos anos 90, estender o poder da computação em rede às atividades da vida cotidiana era uma visão radical. Em 1991, um pequeno grupo de engenheiros da *Sun*, chamado de "Equipe Verde", acreditava que a próxima onda da computação seria a união de dispositivos digitais e computadores. A equipe liderada por James Gosling criou a linguagem de programação Java. A Equipe Verde demonstrou sua nova linguagem com um controlador de entretenimento doméstico interativo e portátil, originalmente voltado para a indústria de televisão digital a cabo. Infelizmente, o conceito estava muito avançado para a época. Mas foi o ideal para a Internet, que estava apenas começando a decolar. Em 1995, a equipe anunciou que o navegador de Internet

Netscape Navigator incorporaria a tecnologia Java (JAVA, 2000).

Hoje, o Java não apenas permeia a *Internet*, mas também é a força invisível por trás de muitos aplicativos e dispositivos que fortalecem nosso cotidiano. De telefones celulares a dispositivos portáteis, jogos e sistemas de navegação a soluções de *e-business*, o Java está em toda parte.

2.4.4 Linguagem de programação em C

Linguagem de programação em C é uma linguagem de programação estruturada, de escopo variável lexical e com recursão. Por concepção, a C fornece construções que mapeiam de forma eficiente as instruções típicas da máquina e, portanto, encontrou uso duradouro em aplicativos anteriormente codificados em linguagens de montagem, incluindo sistemas operacionais, bem como vários *softwares* de aplicativos para computadores que vão desde supercomputadores a sistemas embarcados .

O C foi desenvolvido originalmente por Dennis Ritchie entre 1969 e 1973 em Bell Labs, e usado para re-implementar o sistema operacional *Unix*. Desde então, tornou-se uma das linguagens de programação mais utilizadas de todos os tempos, com compiladores C de vários fornecedores disponíveis para a maioria das arquiteturas de computadores e sistemas operacionais existentes. O C também foi padronizado pelo *American National Standards Institute* (ANSI) desde 1989 e posteriormente pela Organização Internacional de Padronização (ISO) (RITCHIE, 2018).

C é uma linguagem processual imperativa e foi projetada para ser compilada usando compiladores relativamente direto, para fornecer acesso de baixo nível à memória, para fornecer construções de linguagem que mapeiem eficientemente as instruções da máquina e exigem suporte mínimo em tempo de execução. Apesar das suas capacidades de baixo nível, a linguagem foi projetada para incentivar a programação entre plataformas. Um programa C compatível com padrões e portátil pode ser compilado para uma grande variedade de plataformas de computadores e sistemas operacionais com poucas mudanças em seu código-fonte. A linguagem tornou-se disponível em uma ampla gama de plataformas, desde microcontroladores à supercomputadores (RITCHIE, 2018).

Como a maioria das linguagens imperativas, o C possui instalações para programação estruturada e permite o alcance e a recursão variável lexical, enquanto

um sistema de tipo estático evita muitas operações não intencionais. Em C, todo o código executável está contido nas sub-rotinas, que são chamadas de "funções"(embora não no sentido estrito da programação funcional). Os parâmetros de função são sempre passados por valor. A referência *pass-by* é simulada em C, passando explicitamente os valores do ponteiro.

Muitas linguagens posteriores vieram direta ou indiretamente de C, incluindo C ++, Java, *JavaScript*, C#, *Objective-C*, PHP, *Python*, *Swift* e Verilog (linguagem de descrição de *hardware*). Essas linguagens atraíram muitas de suas estruturas de controle e outras características básicas de C. A maioria delas (sendo *Python* a exceção) também são muito sintaticamente semelhantes ao C (UNIVERSITY, 2009).

2.4.5 PHP

2.4.5.1 Introdução

O PHP é uma linguagem simples, mas poderosa, projetada para criar conteúdo HTML e possui plataforma para ser executada e configurada. Assim, o PHP como qualquer outra linguagem, tem um passo a passo de lógica de programação para elaborar programas para executar tarefas comuns, como por exemplo, o processamento de dados de um formulário que interage com um banco de dados para criar gráficos (TATROE, 2013).

2.4.5.2 Aplicação

O PHP pode ser usado de três maneiras principais: *script* do lado do servidor, *script* de linha de comando, e aplicativos GUI do lado do cliente.

O *scripts* do lado do servidor: o PHP foi originalmente projetado para criar conteúdo *WEB* dinâmico, sendo o mais adequado para essa tarefa. Para gerar HTML, é necessário o analisador PHP e um servidor *WEB* através do qual será enviado os documentos codificados. O PHP também se tornou popular para gerar documentos XML, gráficos, animações em *Flash* e arquivos PDF.

O PHP pode executar *scripts* da linha de comando, como Perl, awk ou o *shell Unix*. Com a possibilidade de ser usado os *scripts* da linha de comando para tarefas de administração do sistema, como análise de *backup* e *log*; mesmo alguns *scripts* de tipo de trabalho do CRON podem ser feitos dessa maneira (tarefas PHP não visuais).

Os aplicativos GUI do lado do cliente: Usando PHP-GTK, pode-se escrever aplicações de GUI de plataforma completa em vários PHP.

O PHP é executado em todos os principais sistemas operacionais, das variantes do *Unix*, incluindo Linux, FreeBSD, Ubuntu, Debian e Solaris para *Windows* e Mac OS X. Ele pode ser usado com todos os líderes servidores *WEB*, incluindo o Apache, o *Microsoft* IIS e os servidores *Netscape* / *iPlanet* tendo o próprio idioma é extremamente flexível. O PHP criou em suporte ara gerar arquivos PDF, imagens GIF, JPEG e PNG e filmes em *Flash*. Uma das características mais importantes do PHP é o seu amplo suporte para bancos de dados. PHP suporta todos os principais bancos de dados (incluindo *MySQL*, *PostgreSQL*, *Oracle*, Sybase, MS-SQL, DB2 e bancos de dados compatíveis com ODBC) e até mesmo muitos obscuros. Mesmo mais Também são suportados bancos de dados de estilo NoSQL recentes, como SQLite e MongoDB. Com PHP, criando páginas da *WEB* com conteúdo dinâmico de um banco de dados é notavelmente simples. Finalmente, o PHP fornece uma biblioteca de código PHP para executar tarefas comuns, como banco de dados abstração, manipulação de erros, e assim por diante, com o PHP *Extension and Application Repos- itory* (PEAR). PEAR é um sistema de estrutura e distribuição para componentes.

2.4.6 HTML

*Hypertext Markup Languag*e (HTML) é uma linguagem de marcação padrão para criar páginas e aplicativos da *WEB*. Com *Cascading Style Sheets (CSS) e JavaS- cript*, formam uma tríade de tecnologias para a World Wide Web. Os navegadores da *WEB* recebem documentos HTML de um servidor *WEB* ou do armazenamento local e os tornam em páginas *WEB* multimídia. O HTML descreve a estrutura de uma página da *WEB* de forma semântica e incluiu originalmente sugestões para a aparência do documento. Os elementos HTML são os blocos de construção de páginas HTML e construções HTML, imagens e outros objetos, como formulários interativos, podem ser incorporados na página renderizada. Sendo que ele fornece um meio para criar documentos estruturados, denotando a semântica estrutural para texto, como títulos, parágrafos, listas, *links*, citações e outros itens. Os elementos HTML são delineados por *tags*, escritos usando colchetes angulares. *Tags* como <img /> e <input /> introduzem conteúdo na página diretamente. Outros como <p> ... </ p> *surround* e fornecem infor-

mações sobre o texto do documento e podem incluir outras *tags* como sub-elementos. Os navegadores não exibem as *tags* HTML, mas usam-nas para interpretar o conteúdo da página.

HTML pode incorporar programas escritos em uma linguagem de *script*, como o *JavaScript* que afetam o comportamento e o conteúdo das páginas da *WEB*. A inclusão do CSS define o aspecto e o *layout* do conteúdo. O *World Wide Web Consortium* (W3C), mantenedor dos padrões HTML e CSS, incentivou o uso de CSS em HTML de apresentação explícito desde 1997 (DUCKETT, 2010).

2.4.6.1 Desenvolvimento

Em 1980, o físico Tim Berners-Lee, contratado no CERN, propôs e projetou o INQUIRE, um sistema para pesquisadores do CERN usarem e compartilhar documentos. Em 1989, Berners-Lee escreveu um memorando propondo um sistema de hipertexto baseado na Internet, no qual especificou o HTML e escreveu o *software* do navegador e servidor no final de 1990. Nesse ano, Berners-Lee e o engenheiro de sistemas de dados do CERN, Robert Cailliau, colaboraram em um pedido conjunto de financiamento, mas o projeto não foi formalmente aprovado pelo CERN (BERNERS-LEE, 1998).

A primeira descrição disponível publicamente do HTML foi um documento chamado Tags HTML, mencionado pela primeira vez na Internet por Tim Berners-Lee no final de 1991. Ele descreve 18 elementos que compõem o projeto inicial e relativamente simples de HTML. Exceto para a tag de *hiperlink*, estes foram fortemente influenciados pelo SGMLguid, um formato de documentação baseado em SGML *(Standard Generalized Markup Language)* no CERN, sendo que onze desses elementos ainda existem em HTML 4.

O HTML é uma linguagem de marcação que os navegadores da *WEB* usam para interpretar e compor textos, imagens e outros materiais em páginas visuais ou audíveis. As características padrão para cada item de marcação HTML são definidas no navegador e essas características podem ser alteradas ou aprimoradas pelo uso adicional do CSS por parte do designer de páginas *WEB*. Muitos dos elementos de texto são encontrados no relatório técnico ISO TR 9537 Técnicas para usar o SGML, que por sua vez cobre os recursos das linguagens de formatação precoce de texto,

como a usada pelo comando RUNOFF desenvolvido no início dos anos 1960 para o CTSS (Tempo Compatível Sistema de compartilhamento) sistema operacional: esses comandos de formatação foram derivados dos comandos usados por tipos para formatar documentos manualmente. No entanto, o conceito SGML de marcação generalizada usa em elementos (intervalos anotados com atributos) em vez de apenas efeitos de impressão, com também a separação de estrutura e marcação (BERNERS-LEE, 1998).

Berners-Lee considerou o HTML como uma aplicação do SGML. Foi formalmente definido como tal pela *Internet Engineering Task Force* (IETF) com a publicação de meio de 1993 da primeira proposta para uma especificação HTML, o "Hypertext Markup Language (HTML)"Internet Draft de Berners-Lee e Dan Connolly, que incluiu uma definição de tipo de documento SGML para definir a gramática. O rascunho expirou depois de seis meses, mas foi notável pelo reconhecimento da marca personalizada do navegador NCSA Mosaic para incorporar imagens em linha, refletindo a filosofia da IETF de basear padrões em protótipos de sucesso. Da mesma forma, o Internet-Draft concorrente de Dave Raggett, "HTML + (*Hypertext Markup Format*)", do final de 1993, sugeriu padronizar recursos já implementados, como tabelas e formulários de preenchimento (BERNERS-LEE, 1998).

Depois que os rascunhos HTML e HTML expiraram no início de 1994, o IETF criou um Grupo de Trabalho HTML, que em 1995 completou "HTML 2.0", a primeira especificação HTML destinada a ser tratada como um padrão contra o qual futuras implementações devem ser baseadas. O desenvolvimento adicional sob os auspícios do IETF foi paralisado por interesses concorrentes. Desde 1996, as especificações HTML foram mantidas, com a entrada de fornecedores de *software* comercial, pelo *World Wide Web Consortium* (W3C). No entanto, em 2000, o HTML também se tornou um padrão internacional (ISO / IEC 15445: 2000). O HTML 4.01 foi publicado no final de 1999, com outras erratas publicadas até 2001. Em 2004, o desenvolvimento começou no HTML5 no Grupo de Trabalho de Tecnologia de Aplicação de Hipertexto da *WEB* (WHATWG), que se tornou um produto compartilhado com o W3C em 2008 e completado e padronizado em 28 de outubro de 2014 (BERNERS-LEE, 1998).

2.4.6.2 Estrutura

Uma página *WEB* muito simples não precisa de nenhum programa especial para escrever páginas da *WEB*, simplesmente é necessário usar um editor de texto, como o bloco de notas no *Windows* ou *TextEdit* em um Mac, e depois salvar o arquivo com a extensão de arquivo .html. A estruturação de documentos segue a sequência de um conjunto de elementos: demarcado por duas tags a '<' tag inicial e a *tag* final '>'. A linguagem HTML pode ser dividida em elementos, atributos e entidades. Elemento é a variável que recebe o documento, quando o elemento não possui conteúdo é chamado de "elemento vazio". Atributo é o que caracteriza o elemento, a sintaxe e da forma: nomeDoAtributo = "valor". Entidade é a referência do documento, sua sintaxe segue a forma: &nome-da-entidade.

2.4.7 *Framework*

Em sistemas de informação, um *framework* é muitas vezes uma estrutura em camadas que indica o tipo de programas que podem ou devem ser construídos e como eles se interrelacionariam. Algumas estruturas de sistema de computador também incluem programas reais, especificam interfaces de programação ou oferecem ferramentas de programação para usar os *frameworks*.

Um *framework* pode ser um conjunto de funções dentro de um sistema e como elas se inter-relacionam; as camadas de um sistema operacional; as camadas de um subsistema de aplicação; como a comunicação deve ser padronizada em algum nível de uma rede; e assim por diante. Um *framework* geralmente é mais abrangente do que um protocolo e mais prescritivo do que uma estrutura.

2.4.8 Laravel

Laravel é uma estrutura de aplicação *WEB* com sintaxe expressiva, a qual proporciona o desenvolvimento criativo. Laravel tenta tirar o sofrimento do desenvolvimento facilitando tarefas comuns usadas na maioria dos projetos da *WEB*, como autenticação, roteamento, sessões e armazenamento em cache.

Assim a Laravel é uma ferramenta para facilitar o processo de trabalho do desenvolvedor sem sacrificar a funcionalidade do aplicativo, combinando pontos positivos de outras estruturas da *WEB*, incluindo *frameworks* implementados em outras

linguagens, como Ruby on Rails, ASP.NET MVC e Sinatra. Deste modo, o Laravel é acessível e poderoso, o qual fornece ferramentas necessárias para aplicações grandes e robustas (OTWELL, 2015).

2.4.8.1 História

Taylor Otwell criou a Laravel como uma tentativa de fornecer uma alternativa mais avançada ao *framework* CodeIgniter, que não forneceu determinados recursos, como o suporte interno para autenticação e autorização de usuários. A primeira versão beta da Laravel foi disponibilizada em 9 de junho de 2011, seguida do lançamento Laravel 1, mais tarde no mesmo mês. O Laravel 1 incluiu suporte incorporado para autenticação, localização, modelos, visualizações, sessões, roteamento e outros mecanismos, mas faltava suporte para controladores que impediram que ele fosse uma verdadeira estrutura MVC (OTWELL, 2015).

Laravel 2 foi lançado em setembro de 2011, trazendo várias melhorias do autor e da comunidade. Os principais novos recursos incluíram o suporte para controladores, o que fez do Laravel 2 uma estrutura totalmente compatível com MVC, suporte interno para o princípio de inversão de controle (IoC) e um sistema de modelos chamado Blade. Como desvantagem, o suporte para pacotes de terceiros foi removido em Laravel 2 (OTWELL, 2015).

O Laravel 3 foi lançado em fevereiro de 2012 com um conjunto de novos recursos, incluindo a interface de linha de comando (CLI) chamada Artisan, suporte interno para mais sistemas de gerenciamento de banco de dados, migrações de banco de dados como uma forma de controle de versão para *layouts* de banco de dados, suporte para manipulação eventos e um sistema de embalagem chamado *Bundles*. Um aumento da base de usuários e popularidade Laravel alinhados com o lançamento de Laravel 3 (OTWELL, 2015).

Laravel 4, codinome Illuminate, foi lançado em maio de 2013. Ele foi feito como uma reescrita completa do framework Laravel, migrando seu *layout* para um conjunto de pacotes separados distribuídos pelo Composer, que serve como um gerenciador de pacotes de nível de aplicativo. Tal *layout* melhorou a extensão do Laravel 4, que foi emparelhado com o seu programa oficial de lançamento regular que abrange seis meses entre lançamentos de pontos menores. Outros novos recursos no lançamento

do Laravel 4 incluem semeadura de banco de dados para a população inicial de bancos de dados, suporte para filas de mensagens, suporte interno para o envio de diferentes tipos de *e-mail* e suporte para a eliminação adiada de registros de banco de dados chamados de eliminação suave (OTWELL, 2015).

O Laravel 5 foi lançado em fevereiro de 2015 como resultado de mudanças internas que acabaram em renumerar a futura versão do Laravel 4.3. Novos recursos na versão Laravel 5 incluem suporte para agendar tarefas executadas periodicamente através de um pacote chamado Scheduler, uma camada de abstração chamada Flysystem que permite que o armazenamento remoto seja usado da mesma forma que os sistemas de arquivos locais, o melhor gerenciamento de ativos do pacote através do Elixir e autenticação simplificada externamente manipulada através do pacote opcional de Socialite. Laravel 5 também introduziu uma nova estrutura de árvore de diretório interno para aplicativos desenvolvidos. O No entanto, o Laravel 5.1, lançado em junho de 2015, é o primeiro lançamento da Laravel a receber suporte de longo prazo (LTS), com a disponibilidade planejada de correções de erros por dois anos e mantimentos de segurança por três anos. Os lançamentos LTS de Laravel devem ser lançados a cada dois anos. Já o Laravel 5.3, lançado em 23 de agosto de 2016, os novos recursos em 5.3 estão focados em melhorar a velocidade do desenvolvedor, adicionando melhorias adicionais fora da caixa para tarefas comuns. Então, o Laravel 5.4, lançado em 24 de janeiro de 2017, esta versão possui muitos recursos novos, como Laravel Dusk, Laravel *Mix*, *Blade Components e Slots*, *E-mails* de *Markdown*, fachadas automáticas, melhorias de rotas, mensagens de ordem superior para coleções e muitos outros. Atualmente, o Laravel 5.5, lançado em 30 de agosto de 2017 (OTWELL, 2015).

2.4.9 *Bootstrap*

A linguagem Bootstrap foi idealizada com o objetivo de facilitar a criação de *layout* responsivo na *web*, seus inventores são dois funcionários do *Twitter*: Mark Otto e Jacob thornton. O objetivo da responsividade é programar um código que seja melhor apresentado em diferentes telas sem a necessidade de reescreve-lo, como em: *tablets*, *smartphones*, *smart* tv e *smartwatches*, além do PC *desktop*. O *Bootstrap* nada mais é do que uma biblioteca do CSS que contém diversos itens pré-programados que auxiliam na elaboração (desenvolvimento) de um site (SPURLOCK, 2013).

2.4.10 *MySQL*

É uma ferramenta *open source*, que foi criado pela empresa MySQL AB, e atualmente é um dos sistemas de gerenciamento de banco de dados mais populares desde pequenas aplicações até ao uso de grandes corporações. Deste modo, é um banco de dados definido pelo padrão ANSI / ISO SQL, que está em processo de evolução desde 1986, visto que é uma ferramenta livre para estudo do código-fonte e alteração de acordo as necessidades de cada aplicação. Para adicionar, acessar e processar dados armazenados em um computador, é necessário um sistema de gerenciamento de banco de dados como o *MySQL Serve*r, o qual desempenha um papel central na computação (MYSQL, 2018).

O *MySQL Server* foi originalmente desenvolvido para lidar com grandes bancos de dados muito mais rápido do que as soluções existentes e tem sido usado com sucesso em ambientes de produção. Embora em constante desenvolvimento, oferece conjunto de funções com sua conectividade, velocidade e segurança para acessar bancos de dados na Internet. Também funciona em sistemas cliente / servidor, e possui ferramentas administrativas e variedade de interfaces de programação de aplicativos (APIs) (MYSQL, 2018).

O banco de dados utilizado é relacional, no qual associa tabelas separadas, e organiza arquivos a fim de otimizar velocidade. Com relação a parte lógica, a plataforma oferece um a ambiente de programação flexível, o qual é possível configurar diferentes campos de dados entre diferentes tabelas. Não menos importante, a linguagem é estruturada e padronizada para acessar bancos de dados, a qual possibilita gerar relatórios, incorporar instruções SQL no código escrito ou usar uma API específica de acordo com a aplicação. Além disso, o servidor do banco de dados *MySQL* é confiável, fácil de usar e rápido, o qual permite ser executado em sinegia com outros aplicativos. Por fim, e for dedicado uma máquina para *MySQL*, é possível ajustar as configurações para aproveitar toda a memória, a potência da CPU (MYSQL, 2018).

2.4.11 *Webview*

É um componente do sistema que usa o navegador *Google Chrome* que permite que *apps Android* exibam conteúdo da *Web*. Esse componente vem pré-instalado no dispositivo e deve ser mantido atualizado para garantir que você receba as atualizações

de segurança mais recentes e outras correções de *bugs*, além de ser uma excelente ferramenta para converter um siste responsivo em aplicativo (MATOS, 2018).

O *Webview* permite aos aplicativos que abram janelas do navegador de forma interna, sem ter que 'chamar' o navegador externamente, e assim abrindo uma outra aplicação, consumindo mais recursos do smartphone. Essa funcionalidade é importante para os aplicativos que são utilizados com frequência e que têm muitos links compartilhados em si como *Facebook, Twitter, Instagram* e demais redes sociais (MATOS, 2018).

A plataforma *Android* oferece três padrões básicos de design: android puro, HTML externo puro e modo misto. No primeiro caso é possível criar um app *Android* usando o SDK e manter o *app* no dispositivo usando recursos primitivos do *Android* sendo uma opção rica em controles de interface gráfica e lógica, mas é limitada ao design (MATOS, 2018).

No segundo caso, é possível criar um *site* remoto e permitir que dispositivos *Android* busquem páginas da *Web* usando o navegador do dispositivo. De modeo que essa opção é a versão mais portátil. Qualquer dispositivo com acesso à *Internet* pode interagir com o site remoto externo; no entanto este modelo não usar qualquer um dos vários recursos de *hardware* e *software* do *Android* a exceção do seu navegador de sistema) (MATOS, 2018).

Por fim, no mod misto é possível criar um site interno hospedado no dispositivo. Permitir que as páginas HTML locais interajam com recursos locais do *Android* de modo que cada abordagem oferece vantagens e desvantagens.

2.4.12 *Raspberry Pi*

Os dispositivos *Raspberry Pi* são sistemas embarcados com processamento equivalente ao de computadores de baixo custo voltado principalmente a iniciativa educacional de promoção do ensino de computação principalmente para o público infantojuvenil, devido ao tamanho reduzido e a ampla comunidade de desenvolvedores o equipamento se torna uma boa opção para construção de projetos e protótipos, principalmente na área de computação e eletrônica embarcada (FOUNDATION, 2012).

Uma ampla gama de microcomputadores foram projetados pela fundação *Raspberry Pi*, até o ano de 2018 nove modelos foram lançados e a diferenciação entre

eles são principalmente pela quantidade de portas de entrada e saída USB (*Universal synchronous bus*) e pela tecnologia de taxa de transmissão, como o USB 1.0, USB 2.0 ou o USB 3.0, o processador do equipamento, enquanto os modelos mais simples utilizam processadores com *clock* na faixa de 512 MHz, a amostra mais recentemente lançada é equipado com o Cortex-A53 de 1 GHz e pela memória RAM dos dispositivos que variam entre 512 MB à 1GB de armazenamento (EKLOT, 2018).

2.4.13 Amazon Web Service

A *Amazon WeB Service* (AWS), é uma plataforma da empresa norte-americana Amazon, a qual iniciou suas atividades em 1994 na garagem do fundador Jeff Bezos, e que oferece serviços de computação em nuvem com armazenamento de dados e outras funcionalidades. A oferta dessa infraestrutura começou em 2006, de modo a ser extramente útil a aplicações de desenvolvedores, visto que a computação em nuvem reduz despesas iniciais de infra-estrutura. Pois, a computação em nuvem é a entrega de recursos sob demanda em que não é necessário fazer grandes investimentos iniciais em *hardware* e gerenciamento. Assim, as empresas que estão começando novos negócios não precisarem planejar e adquirir servidores e pagam apenas os recursos utilizados da plataforma AWS (SERVICE, 2018).

O *AWS Cloud* fornece um amplo conjunto de serviços de infraestrutura, como capacidade de computação, opções de armazenamento, redes e bancos de dados que são entregues sob demanda de acordo com o pacote do serviço utilizado. De forma a permitir que start-ups, pequenas e médias empresas possam acessar os blocos de construção da plataforma de acordo com suas aplicações (SERVICE, 2018).

Há três tipos de modelos de computação em nuvem: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS). O IaaS oferece serviços de infraestrutura de rede para fazer a função dos servidores e *datacenters* tradicionais com o uso da AWS. Já o PaaS, é a disposnilização de uma plataforma de desenvolvimento e gerenciamento de aplicativos para o usuário, permitindo que o mesmo não se preocupe sobre aquisição de recursos, planejamento de capacidade, manutenção de *software*, ou qualquer envolvidos na execução do aplicativo. Por fim o SaaS, é a disponibilidade do *software* como serviço (SERVICE, 2018).

Um recurso importante da plataforma AWS é a *Amazon Elastic Compute*

Cloud (Amazon EC2), que é um serviço da *Web* que fornece computação segura e capacidade na nuvem, a qual foi projetada para tornar a programação mais fácil para desenvolvedores. Essa interface permite elasticidade de alcance de instâncias na *web* aumentando ou diminuindo a capacidade em minutos de forma segura por meio de APIs (SERVICE, 2018).

2.4.14 HomeCare

Homecare é um termo geral que representa uma ampla gama de serviços para tratamento domiciliar que visa satisfazer as necessidades sociais e de saúde das pessoas, de maneira apropriada por cuidadores formais e informais utilizando recursos tecnólogicos dentro de um ambiente equilibrado e acessível. O emprego de serviços em tratamento domiciliar começou a ser utilizado nos Estados Unidos na década de 80, mas chegou no Brasil no início de 90. No entanto, somente em 2006 que Anvisa estabeleceu regras para serviços de atendimento domiciliar, as quais foram constituídas com a colaboração da Agência Nacional de Saúde Suplementar e da Secretaria de Atenção a Saúde do Ministério da Saúde (ANVISA, 2018).

3 DESENVOLVIMENTO

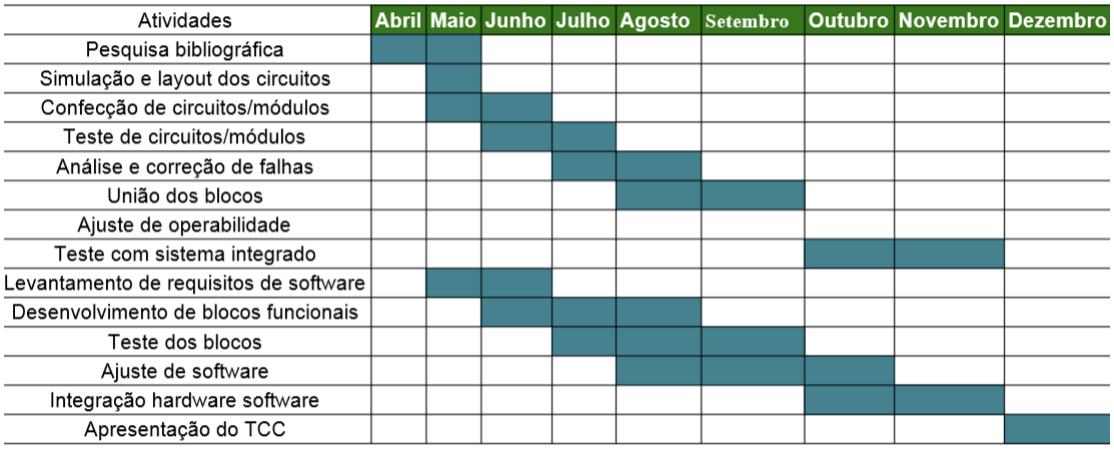
3.1 ESTRUTURA DO PROJETO

A estrutura do projeto inicia a metodologia com a pesquisa bibliográfica de projetos similares, seguido da utilização de ferramentas de simulação, afim de evitar custos desnecessários e definir blocos chaves para seguimento do projeto. Logo após passa pelos testes com circuitos/módulos, garantindo o funcionamento dos blocos antes mesmo de qualquer interligação entre eles.

Posteriormente, há a união de blocos que formam subsistemas, validando a integração dos blocos, seguido do levantamento de requisitos do sistema computacional e desenvolvimento de blocos funicionais do programa até chegar a integração *hardware-software*.

A figura 3.1 mostra o diagrama de Gantt elaborado pelos integrantes da equipe para as tarefas a serem realizadas.

FIGURA 3.1 – Diagrama de Gantt do projeto.



Fonte: Autoria própria, 2018

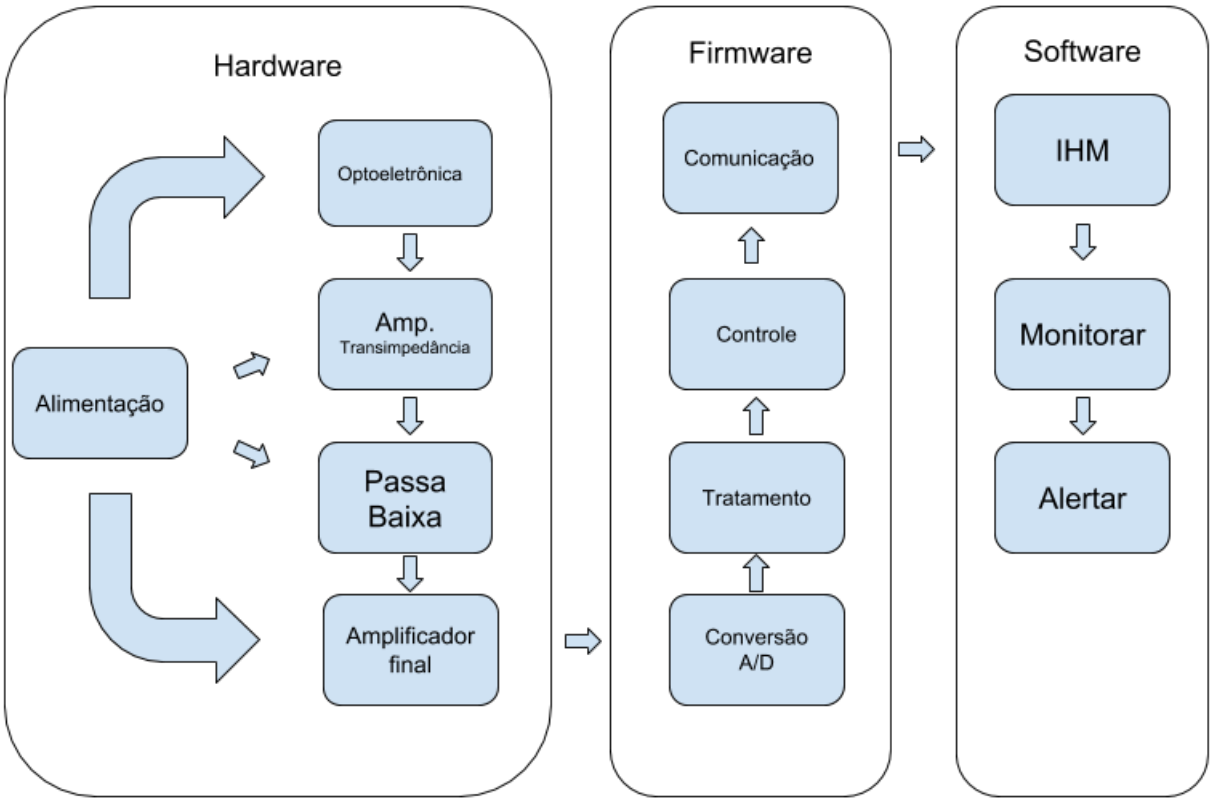
Como a figura 3.1 ilustra, as tarefas foram espalhadas em períodos de tempo de 2 semanas, tempo confortável, porém não demasiadamente grande.

É interessante ressaltar o cuidado aplicado na escolha das datas, o que resultou em períodos para identificação e correção de eventuais falhas ocasionadas do processo de desenvolvimento, tanto de *hardware* quanto de *software*.

3.2 DIAGRAMA DO PROJETO

O diagrama da figura 3.2 representa o fluxograma aplicado ao projeto de oximetria, é possível notar a presença de circuitos aplicados na área de instrumentação médica como: filtros passa-baixa, amostradores e amplificadores.

FIGURA 3.2 – Fluxograma de oximetria.



Fonte: Autoria própria, 2018

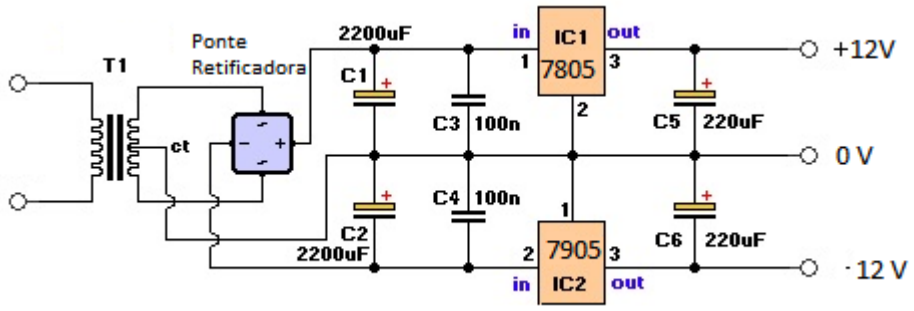
3.3 ALIMENTAÇÃO

O sistema de potência está representado na figura 3.3. Devido a uitlização de amplificadores de potência é de vital importância que a fonte seja simétrica, justificando a utilização dos reguladores de tensão 7805 e 7905 que serão empregados.

A primeira conversão será feita pelo transformador que será responsável por converter a magnitude de tensão de 127V para 12V, tanto para valores positivos quanto valores negativos. Os diodos formam a retificação da fonte de entrada. Os capacitores utilizados são do tipo eletrolítico devido a alta capacitância exigida pela aplicação. Por se tratar de sistemas que não possuem alta demanda de energia, a corrente fornecida

por ambos os reguladores é suficiente para a atividade a ser desempenhada pelo restante do circuito.

FIGURA 3.3 – Alimentação do circuito.



Fonte: Alterado de Sparkfun(2016)

3.4 PROJETO INICIAL

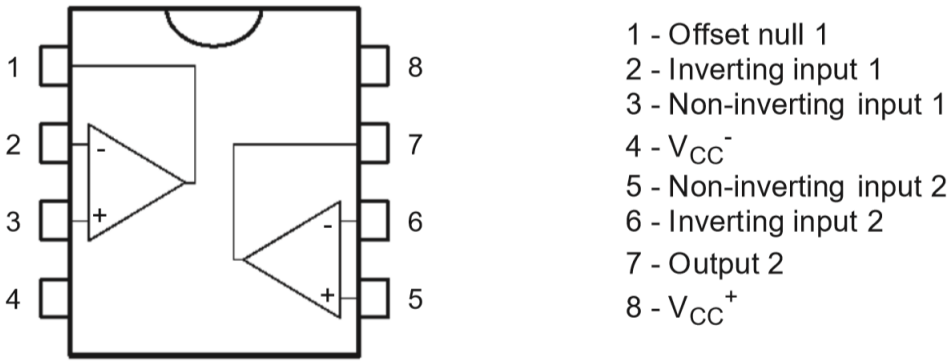
A ideia original para o desenvolvimento do projeto era utilizar um sensor médico de oximetria, assim a equipe ficaria encarregada do processo de filtragem, tratamento e amostragem do sinal. Durante o desenvolvimento os participantes tiveram dificuldades em isolar o sinal de oximetria, devido o baixo nível de tensão, assim a fim de conseguir prosseguir com o projeto a equipe realizou a compra de um módulo sensor que embarca o sistema de tratamento, amostragem e processamento.

Apesar da mudança repentina a equipe optou manter a descrição dos hardwares que foram desenvolvidos durante o trabalho e focar na plataforma *mobile* e AWS.

3.4.1 Amplificadores

Existem dois estágios de amplificação no projeto, o primeiro é referente ao amplificador de transimpedância utlizado para conversão da corrente do fotodiodo em tensão, o que torna possível aplicar os demais processos para tratamento do sinal. E o segundo está relacionado com a etapa final de amplificação, para ajustes mais finos do processo. Ampos os blocos utilizam o amplificador operacional TL082, mostrado na figura 3.4. O modelo escolhido apresenta alta impedância de entrada, *Slew Rate* (CMRR) de 13V/μs e baixa corrente de entrada e de *offset*.

FIGURA 3.4 – Amplificador TL082.



Fonte: Datasheet TL082

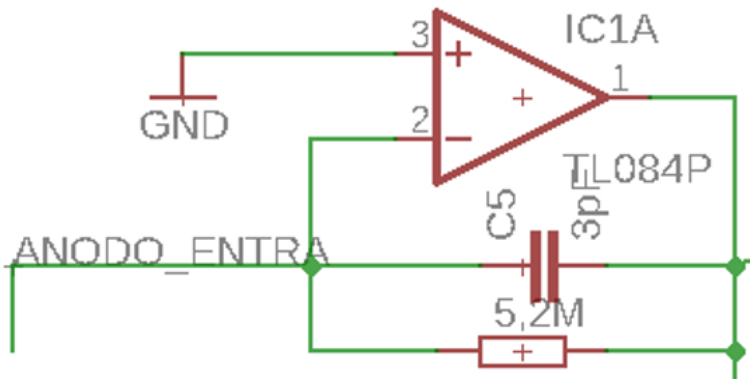
3.4.2 Circuitos de tratamento de sinal

Os circuitos de tratamento são elaborados para fazer a adequação do sinal e são compostos pelo amplificador de transimpedância, filtro passa baixa e o amplificador de estágio final.

3.4.2.1 Amplificador de transimpedância

Conforme descrito na revisão bibliográfica o amplificador de transimpedância, realiza a mudança de domínio do sinal da corrente do fotodiodo para a tensão de saída do amplificador, tornando a tensão de saída uma função da corrente de entrada, ou seja, é utilizado para o sinal deixar de ser portado na corrente e ser captado pela tensão. Para que o sistema aumente sua robustez foi utilizado o circuito da figura 3.5

FIGURA 3.5 – Amplificador de transimpedância.



Fonte: (CLARK et al., 1997)

Os valores escolhidos para o para o capacitor e resistor são baseados na frequência de limitação da banda do amplificador, conforme a equação 3.1 os valores são de 3 pF e 5,2 M %ohm, assim a tensão de saída será dada por:

$$Vin - Vout = IdiodoZ \tag{3.1}$$

Vin é a tensão de entrada na porta inversora e Vout a tensão de saída do amplificador, Idiodo é a conversão do sinal luminoso em corrente elétrica e Z corresponde a impedância que capacitor e resistor formam. evoluindo a equação 3.1 tem-se a equação 3.2.

$$Vin - Vout = \frac{Idiodo.(1 + R.jwC)}{R} \tag{3.2}$$

Através da 3.2 é possível perceber que um aumento na frequência causa um aumento na reatância capacitiva resultando em acréscimo da diferença entre tensão de entrada e saída, ou seja, o sinal é atenuado, enquanto que para frequências na grandeza de algumas centenas de Hertz o valor de impedância fica muito próximo da reatância capacitiva sozinha sem a interferência do resistor.

3.4.2.2 Filtro passa baixa

A faixa de frequência interessante para o aquisição de dados é diretamente relacionado com o sistema cardiorrespiratório. No projeto desenvolvido por McKee et al. (2015) o filtro aplicado foi da forma digital e sua ordem foi de 7, o que garante uma transição entre a banda de passagem e a banda de rejeição abrupta na frequência de corte de 4 Hz selecionado pela equipe do projeto, o critério de decisão utilizado foi baseado no batimento cardíaco humano que está na faixa entre 0 à 4 Hz.

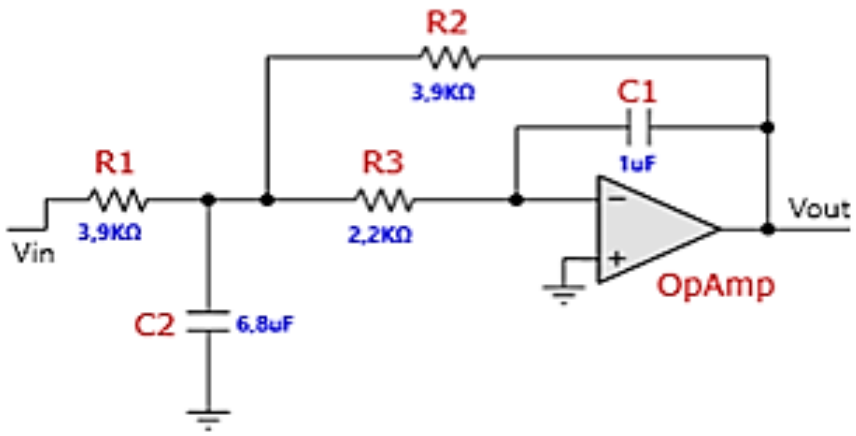
Devido a pouca experiência da equipe com o uso de técnicas de filtragem digital foram aplicados filtros analógicos, utilizando o mesmo raciocíno empregado por McKee et al. (2015) para a frequência de corte, entretanto devido ao fato do filtro não apresentar ordem elevada a frequeência de corte foi multiplicada para uma década, a fim das distorções da função de transferência terem o efeito mínimizado sobre o módulo do sinal a ser amostrado.

O filtro construido foi projetado com o auxílio do software *Filter Pro* da *Texas Instruments*, através do programa é possível configurar o tipo de topologia do filtro,

a frequência de corte, a ordem e aplicar o ajuste para que o software utilize valores comerciais durante a concepção do filtro.

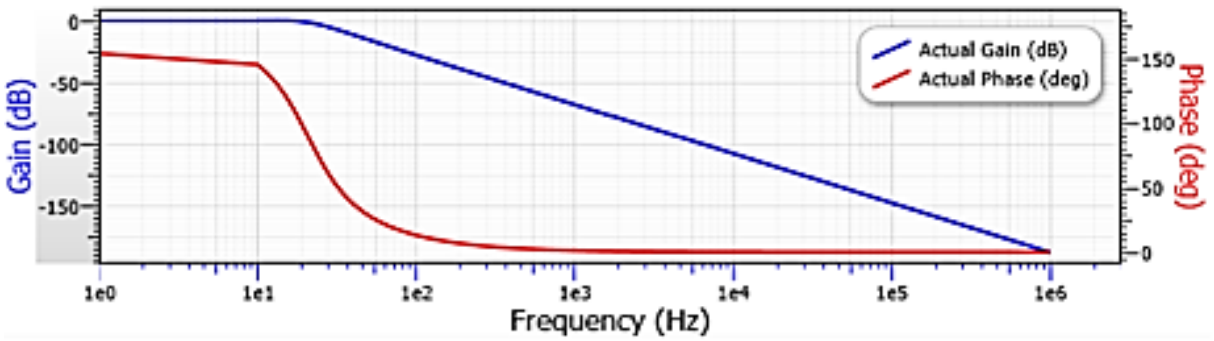
A figura 3.6 mostra o filtro passa baixa desenvolvido para o projeto, enquanto que a figura 3.7 3-11 expressa a resposta em frequência do mesmo.

FIGURA 3.6 – Filtro de segunda ordem.



Fonte: Autoria própria

FIGURA 3.7 – Resposta em frequência do filtro de segunda ordem.



Fonte: Autoria propria

Na figura 3.7 apresenta duas curvas, a curva azul representa a atenuação do sinal no domínio da frequência, enquanto que a curva vermelha é relacionada ao defasamento do sinal de saída com relação ao sinal de entrada.

3.4.3 Microcotrolador

O Microcontrolador escolhido pelos integrantes da equipe foi o modelo MSP430 G2553, representado pela figura 3.8, da *Texas Instruments* por apresentar bons resul-

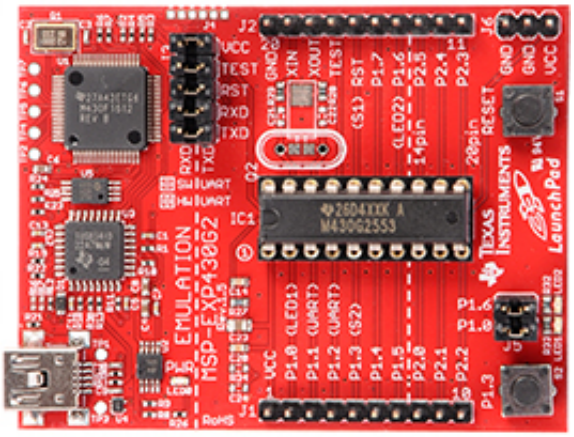
tados e ambiente amigável ao desenvolvimento.

A utilização de tal plataforma produzirá desenvolvimento ágil e de fácil implementação devido a experiência dos integrantes do projeto.

A arquitetura do modelo G2553 é a RISC de 16 bits e sistema von-Neumann de barramento, seu enfoque é em aplicações de baixa energia. O dispositivo conta com 5 níveis de *low power mode* (LPW) para que o consumo de energia não seja impeditivo na aplicação do microcontrolador, o modelo possui conversor analógico digital de 10 bits com múltiplos canais que pode ser multiplexados de acordo com a necessidade do desenvolvedor (INSTRUMENTS, 2011).

Devido a experiências anteriores e o fato de um dos integrantes já possuir o *launchpad*, a codificação torna-se mais viável e ágil quando comparado a outras opções do mercado. O diagrama presente na figura 3.9 ilustra os blocos que compõem o *hardware* escolhido.

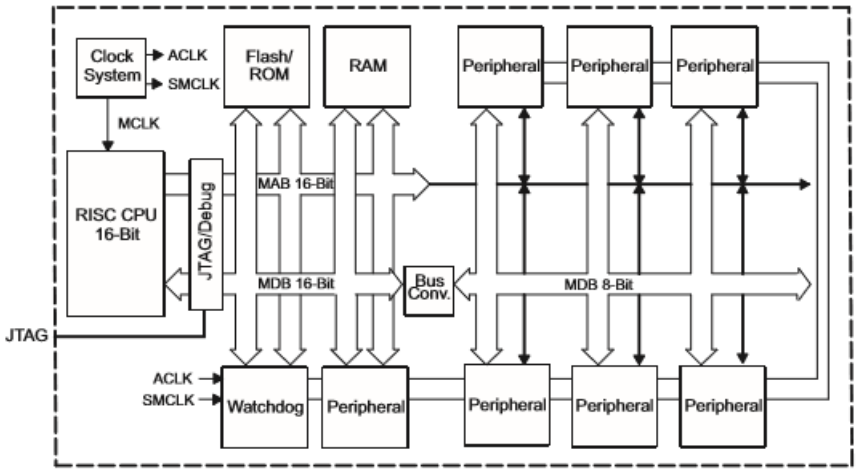
FIGURA 3.8 – Microcontrolador escolhido.



Fonte: (INSTRUMENTS, 2011)

O controlador central do microcontrolador segue a arquitetura von Neumann, em que a memória de dados e memória de comandos compartilham a mesma região, suas instruções são do tipo RISC e são executadas em poucos ciclos de clock, porém mais código é necessário para executar as mesmas funções quando comparado com a topologia CISC.

FIGURA 3.9 – Diagrama do microcontrolador escolhido.



Fonte: (INSTRUMENTS, 2011)

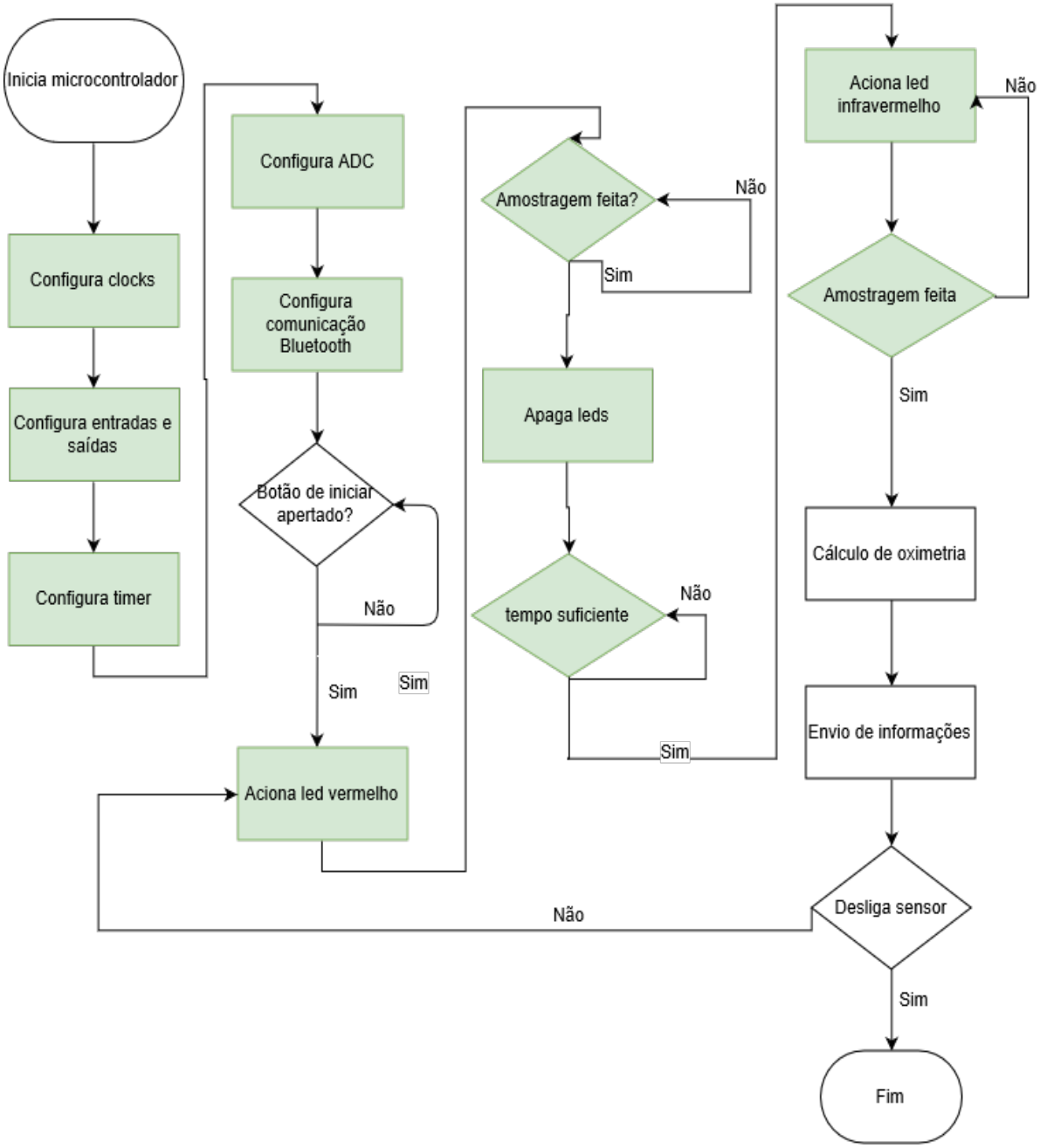
O sistema de endereçamento segue a lógica de 16 bits, o que permite 65536 endereços, o dipositivo tem 512 *bytes* de RAM espaço em que o código e as variáveis são armazenadas. A memória ROM fica a cargo do fabricante que insere os endereços dos vetores de interrupção, assim o usuário não poderá altera-los causando desordem sobre o funcionamento correto do dispositivo (INSTRUMENTS, 2013). Alguns dos principais periféricos presentes no *launchpad* são os *timers* que tem como função temporizar os processos secundários do dispositivo, como o conversor analógico digital de 10 *bits*, a comunicação UART (*Universal Assynchronous Receiver/Transmitter*), ou o módulo I2C (*Inter Integrated Communication*) para comunicação serial entre dispositivos.

3.4.4 Fluxograma algoritmo

A programação inserida sobre o microcontrolador segue a máquina de estados representada pela figura 3.10.

Os blocos verdes são os trechos de código que já foram desenvolvidos no microcontrolador, enquanto os brancos são os blocos funcionais que ainda faltam. Pelo fato de o projeto desenvolver o sensor o fluxograma apresenta a configuração dos *timers* para o chaveamento dos LEDs, assim uma malha fechada compõem o fluxo de instruções de acionamento, amostragem e envio dos dados.

FIGURA 3.10 – Fluxograma do algoritmo.



Fonte: Autoria própria, 2018

3.5 PROJETO VIGENTE

Devido a dificuldade de estabilização do sinal oximétrico por relação a sua baixa amplitude de tensão, os integrantes da equipe decidiram focar no sistema de comunicação paciente-médico e utilizar um sensor proveniente de um módulo comercial que realiza o encapsulamento do tratamento e processamento do sinal oximétrico.

Os próximos capítulos apresentam o desenvolvimento frente as mudanças

realizadas pelos integrantes.

3.5.1 Módulo Max30100

Conforme supracitado o módulo escolhido foi o sensor Max30100 que devido a suas dimensões reduzidas é otimizado para aplicação *wearable*, a figura 3.12 representa o módulo escolhido.

FIGURA 3.11 – Módulo oximétrico utilizado.

Fonte: (FELIPE, 2010)

O sensor utiliza dois LED, um de comprimento de onda vermelho e outro do infravermelho, com processamento digital de sinais de baixo ruído e remoção da luminosidade do ambiente possibilitando a detecção do nível de saturação de oxigênio e de batimento cardíaco. O dispositivo contém um conversor analógico-digital de 16 bits com amostragem entre 50 Hz à 1 kHz, também possui *driver* para a regulação da corrente dos LEDs controlando a intensidade do sinal luminoso que será captado pelo sensor. Pelo fato do equipamento processar digitalmente os dados, a comunicação para aquisição dos dados é utilizado o protocolo I2C (*Inter-Integrated Circuit*) que utiliza um pino para a comunicação e outro para a sincronização dos dispositivos.A faixa de tensão operante do dispostivo se encontra entre 1,8V à 3,3V.

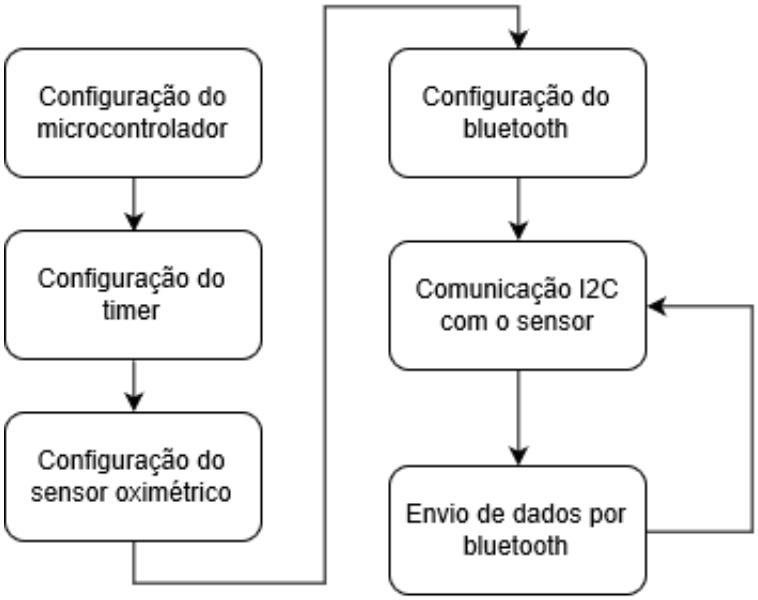
3.5.2 Microcontrolador

O microcontrolador contínua o mesmo utilizado para o projeto inicial planejado pelos integrantes da equipe. Assim a descrição do hardware realizado no capítulo 3.4.3

contínua válida para o projeto vigente.

A diferença está presente na estruturação do diagrama de blocos do algoritmo do microcontrolador que deixou de ser a figura 3.10 para a figura 3.12.

FIGURA 3.12 – Fluxograma algoritmo vigente.



Fonte: Autoria própria

O fluxograma do algoritmo inicia pela configuração dos periféricos, através quatros primeiros blocos, que serão utilizados para a aplicação do algoritmo.

No primeiro ocorre o ajuste do microcontrolador começando pela desabilitação do *watchdog* da unidade controladora, regulação do *clock* principal do dispositivo para 16 MHz, indicação dos pinos 1.1 e 1.2 no modo secundário e configuração da comunicação I2C e UART do dispositivo.

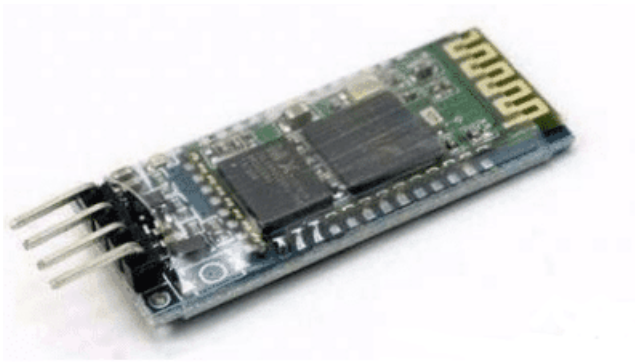
O segundo passo é a configuração do timers que retira o dispositivo do *low power mode* para que seja possível a leitura do sensor e o envio dos dados, o *timer* também será responsável por realizar o *debounce* dos botões de entrada. Em seguida acontece a configuração do sensor oximétrico max30100 onde ocorre a configuração básica na qual o dispositivo operará e por último do *bluetooth*.

Os últimos dois blocos são responsáveis pela aquisição dos dados do sensor através da comunicação I2C com *clock* de 100 kHz e o seguinte para o envio de dados com o bluetooth através da comunicação serial UART.

3.5.3 *Bluetooth*

A comunicação *Bluetooth* se estabelece entre o microcontrolador MSP430 e o micro computador *raspberry pi*, o módulo *bluetooth* utilizado é o modelo HC-06 representado pela imagem 3.13.

FIGURA 3.13 – Módulo HC-06.



Fonte: Autoria própria

Para a comunicação com o dispositivo foi utilizado o protocolo *UART* (*Universal Asynchronous receiver/transmitter*) com taxa de simbolo de 9600 baud.

3.5.4 *Raspberry*

O microcomputador *raspberry* utilizado é o modelo pi 3B que tem processador de 1 GHz e 1 GB de memória RAM, a função do dispositivo é a mesma de um gatesway entre o MSP430 e a nuvem que faz o armazenamento dos dados, assim os dados são recebidos através do *bluetooth* embarcado no dispositivo, e são armazenados ao servidor utilizando o *amazon web service*.

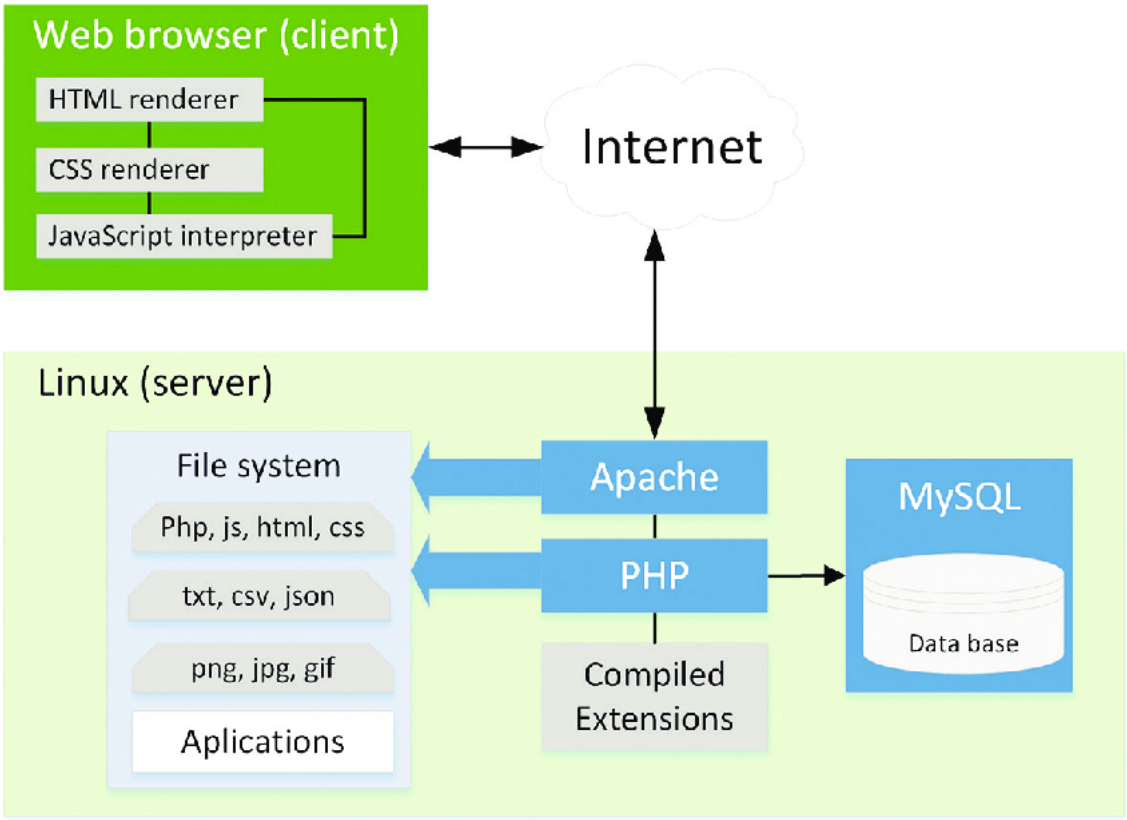
A fim de garantir que o raspberry esteja operante a cada momento que for ligado, o conceito de serviço foi utilizado para que o processo de conexão, tanto do bluetooth quanto com a AWS, seja estabelecida no inicializar do sistema operacional.

Sobre o sistema operacional do computador são executados *scripts* em linguagem php que realizam a conexão com o módulo *bluetooth* conectado ao MSP430 e a comunicação com a máquina virtual na AWS, assim através do comando GET os dados são enviados sobre o cabeçalho do protocolo HTTP (*Hyper Text Transfer Protocol*).

3.5.5 Amazon Web Service

Sobre o AWS é executada a combinação LAMP, como exibido pela figura 3.14 em que ocorre a junção dos *softwares Linux*, Apache, *MySQL* que será gerado para armazenamento e verificação de dados relacionados a oximetria e a linguagem PHP para criação e manutenção do serviço *web* juntamente com a estrutura *front end* formada por HTML e CSS que faz a interface com o usuário.

FIGURA 3.14 – Modelo computacional LAMP.



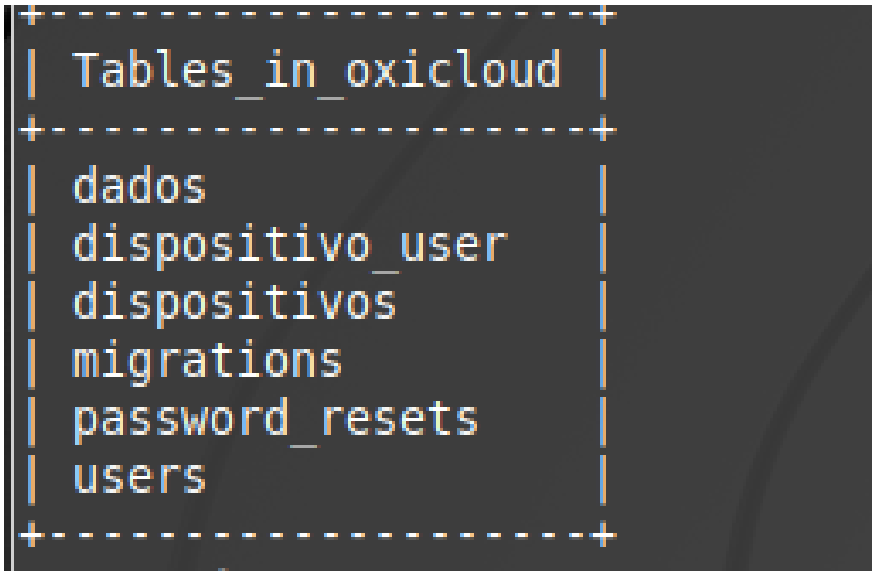
Fonte: (MARTOS, 2018)

O Linux se trata de um máquina virtual sendo executada sobre os servidores da Amazon, assim o usuário não necessita se preocupar em manter a infraestrutura da rede para garantir a conexão e manutenção dos dados.

A fim de integrar com agilidade a parte AMP foi utilizado o *framework* Laravel possibilitando desenvolvimento enxuto e focado na entrega dos resultados. O apache atua como o servidor que receberá os comandos do protocolo HTTP e gerará as respostas conforme as requisições do usuário, assim ele fará o roteamento dos arquivos PHP e HTML que serão processados localmente no servidor o resultado será encapsulado no protocolo HTTP e o apache fará o envio do pacote.

O papel do banco de dados não é feito pelo *MySQL* mas sim pelo Mariadb que utiliza a mesma estrutura de dados do *MySQL*, mas como já vem embarcado no *framework* Laravel aumenta a praticidade para implementação, 6 tabelas foram criadas para administração dos dados, conforme mostrado pela tabela 3.15

FIGURA 3.15 – Tabelas do banco de dados.



Tables_in_oxicloud
dados
dispositivo_user
dispositivos
migrations
password_resets
users

Fonte: Autoria própria

Cada tabela da figura 3.15 tem uma função para a construção do site. A tabela dados faz o armazenamento do id (número de identificação) dos dados, id do usuário, id do dispositivo, o valor das medidas, o horário que o dado foi criado e o horário de armazenamento, conforme na figura 3.16

FIGURA 3.16 – Tabela dados.

id	user_id	dispositivo_id	medida	created_at	updated_at
135	1	1	70.2	2018-11-18 01:19:30	2018-11-18 01:19:30
138	1	1	88.57	2018-11-18 01:43:12	2018-11-18 01:43:12
139	1	1	87.70	2018-11-18 01:45:08	2018-11-18 01:45:08
140	1	1	87.99	2018-11-18 01:45:36	2018-11-18 01:45:36
141	1	1	77	2018-11-18 01:45:45	2018-11-18 01:45:45
142	1	1	79	2018-11-18 01:46:45	2018-11-18 01:46:45
143	1	1	81	2018-11-18 01:46:50	2018-11-18 01:46:50
144	1	1	85	2018-11-18 01:46:54	2018-11-18 01:46:54
145	1	1	87.75	2018-11-18 02:16:41	2018-11-18 02:16:41
146	1	1	87.00	2018-11-18 02:16:48	2018-11-18 02:16:48
147	1	1	87.6	2018-11-18 02:16:53	2018-11-18 02:16:53
148	1	1	80	2018-11-18 02:41:04	2018-11-18 02:41:04

Fonte: Autoria própria

A tabela dispositivo_user salva a relação entre os usuários e seus dispositivos, assim há uma conexão de qual dispositivo de determinado id pertence ao usuário de determinado id, como ilustrado pela figura 3.17.

FIGURA 3.17 – Tabela dispositivo_user.

id	user_id	dispositivo_id	medida	created_at	updated_at
135	1	1	70.2	2018-11-18 01:19:30	2018-11-18 01:19:30
138	1	1	88.57	2018-11-18 01:43:12	2018-11-18 01:43:12
139	1	1	87.70	2018-11-18 01:45:08	2018-11-18 01:45:08
140	1	1	87.99	2018-11-18 01:45:36	2018-11-18 01:45:36
141	1	1	77	2018-11-18 01:45:45	2018-11-18 01:45:45
142	1	1	79	2018-11-18 01:46:45	2018-11-18 01:46:45
143	1	1	81	2018-11-18 01:46:50	2018-11-18 01:46:50
144	1	1	85	2018-11-18 01:46:54	2018-11-18 01:46:54
145	1	1	87.75	2018-11-18 02:16:41	2018-11-18 02:16:41
146	1	1	87.00	2018-11-18 02:16:48	2018-11-18 02:16:48
147	1	1	87.6	2018-11-18 02:16:53	2018-11-18 02:16:53
148	1	1	80	2018-11-18 02:41:04	2018-11-18 02:41:04

Fonte: Autoria própria

A tabela dispositivos é onde há o arquivamento de qual sensor já foi cadastrado, com o armazenamento do id do dispositivo, o id do usuário que efetuou o cadastro, observações, o momento que foi criado e o momento que foi salvo, a figura 3.19 exemplifica a descrição.

FIGURA 3.18 – Tabela dispositivos.

id	user_id	observacoes	created_at	updated_at
1	1	Protótipo 01	2018-11-22 15:50:07	2018-11-22 15:50:07

Fonte: Autoria própria

A tabela *password_reset* guarda as requisições de mudança de senha, a figura 3.19 evidencia os dados armazenados.

FIGURA 3.19 – Tabela *password_resets*.

id	user_id	observacoes	created_at	updated_at
1	1	Protótipo 01	2018-11-22 15:50:07	2018-11-22 15:50:07

Fonte: Autoria própria

A tabela *users* arquiva todos os dados referentes aos usuários cadastrados na plataforma do *site*, os dados são: o id, o nome, permissão, *e-mail*, *password*, *remember_token*, a data de criação e a de atualização, a figura 3.20 exibe a tabela.

FIGURA 3.20 – Tabela *users*.

id	name	permissao	email	password	remember_token	created_at	updated_at
1	Guilherme Canova Bueno	administrador	guilherme_512_7@hotmail.com	$2y$10$ZOhu.Rr9mQCZHz8PedYD9.KGSCZ6clIQQuCgp0bin4tuXNblTp2.u	75BvCbQTCv2rYVVZg9WurV0QyHQJwvH0zk9qzVA3BvWdcxpjm8G05wG1C3yi	2018-11-11 19:00:20	2018-11-23 02:55:34
2	André	administrador	andremachoski2009@hotmail.com	$2y$10$Atpzfka6MqZJaz/0q24i3ugfC/6uxVYzgyXfMJtQ88SqyryCxnkFa	1ypS7L14XwH5QtPEBmUvcV2l36vpF98cSImdkwFnBDqL7aK2HlNRl8bhkxhE	2018-11-18 10:25:58	2018-11-18 10:25:58
4	André Allan	usuario	adrcsv@gmail.com	$2y$10$jCzJ80pWFJg3D2Ob5ifsE.YdnPv0y2tQJA8DLLjhsgwQ4JvQIgdHu	8PfDb69qNNCCaJONoe2VKm9Nek0faCe4fTaC9lI76cBSUuzkux5kmdwXx4B1	2018-11-23 14:10:53	2018-11-23 14:10:53
5	Marcelo Cruz	usuario	marcelo.gaviliki@gmail.com	$2y$10$FOxXxT2q3AhhHTTWJs3kMutI1gUyV.WE4uwvImVnhhBJ2.5O.uI42	NULL	2018-11-24 12:19:43	2018-11-24 12:19:43

Fonte: Autoria própria

O PHP é utilizado para criar a interatividade com o usuário, como a geração de gráficos, tabelas e os sistemas de *login* do *site*. O objetivo do PHP é gerar dinamicamente as páginas *web* de acordo com as requisições do usuário, assim esta encarregado de realizar consultas e escritas ao banco de dados no mariadb onde estão armazenada as informações dos usuários e sensores cadastrados, sistema para mudança de senha e o índice de tabelas presente no banco de dados.

Em suma, o Apache recebe as requisições do usuário realizando a execução dos scripts em PHP que faz acesso aos bancos de dados e realiza a parte lógica do *site*, após a execução os resultados são interpretados em respostas gráficas juntamente com o HTML e CSS que serão empacotados no protocolo HTTP e enviados de volta ao usuário pelo servidor Apache.

3.5.6 *Front-End*

O desenvolvimento *Front-End* é construido utilizando as linguagens de marcação HTML e CSS assim todo o leiaute é gerado através da interpretação destas estruturas. O *Front-End* aplicado apresenta responsividade, assim consegue se adaptar de acordo com a tela que usuário está utilizando, desde das menores como em *smartphones* até mesmo à telas de computador.

3.5.7 Aplicativo *Android*

Para desenvolvimento em plataforma mobile foi utilizado o ambiente de desenvolvimento integrado (IDE - *Integrated Development Environment*) disponibilizada

pela empresa Google, assim utilizando as API (*Application Programming Interface*) da própria criadora do sistema ocorre a programação sobre o ambiente *Android*. O *software* desenvolvido para acessibilidade utiliza o componenete *webview* onde requisição web são emcaminhadas ao servidor, hospedado na AWS, através do protocolo HTTP. O servidor, por seu lado, responderá normalmente como uma solicitação padrão de um cliente. Ao receber os dados a *webview* fará o redimensionamento da tela para que a informção a ocupe totalmente, tornando o *site* reponsivo para diferentes dispositivos. Devido a ampla gama de aparelhos que executam o sistema operacional android existirá uma diferença entre cada tela, porém não acarretará em perda de informação significativa.

4 RESULTADOS

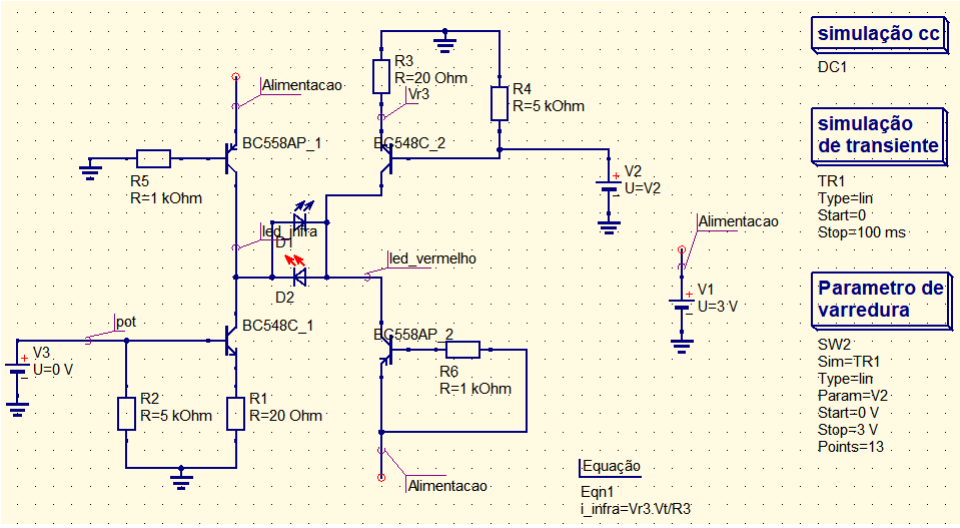
4.1 RESULTADOS DO PROJETO INICIAL

Os resultados que podem ser apresentados até o momento são relacionados ao ambiente computacional, ou seja, simulações dos circuitos que compõem o projeto, *boards* desenvolvidas em *softwares* de CAD para desenvolvimento da rede de processamento, programação java do ambiente *mobile* e desenvolvimento do *firmware* que será aplicado ao microcontrolador

4.1.1 Driver de controle dos leds

Os LEDs aplicados ao projeto são do tipo alto brilho, devido ao fato da sensibilidade do sensor empregado se alterar conforme o comprimento de onda é necessário o uso de circuitos que controlem a potência de cada LED. A figura 4.1 mostra o circuito de controle de potência dos LEDs.

FIGURA 4.1 – Circuito de controle de potência.

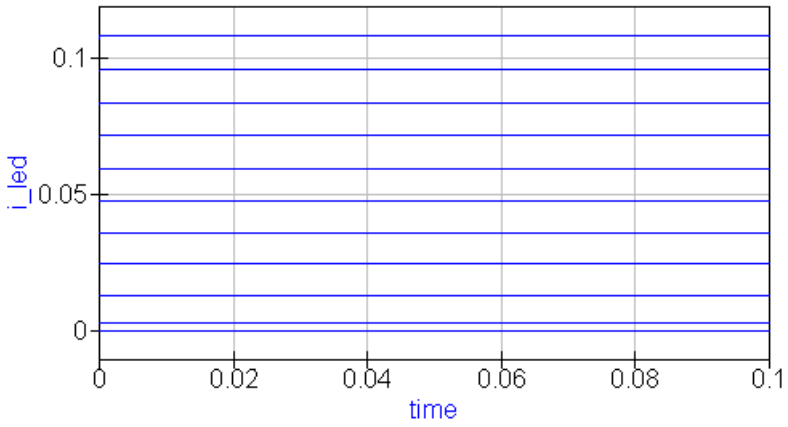


Fonte: Autoria própria, 2018

O circuito se assemelha a uma ponte H, utilizada para controle de motores CC. A figura 4.2 indica a corrente resultante que flui sobre os leds quando há o controle de potência através dos transistores BC 548 do tipo NPN. Os transistores BC 558 do tipo PNP são utilizados para o chaveamento dos LEDs.

A tensão aplicada sobre a base dos transistores NPN varia na faixa de 0V à 3V com passo de 250mV, assim 13 curvas de corrente são geradas. Conforme é possível observar na figura 4.2 a corrente sobre os LEDs variou, discretamente, entre os valores de 0A à aproximadamente 100mA, o que resultará na variação do fluxo luminoso gerado pelo dispositivo emissor de luz.

FIGURA 4.2 – Correntes resultantes.

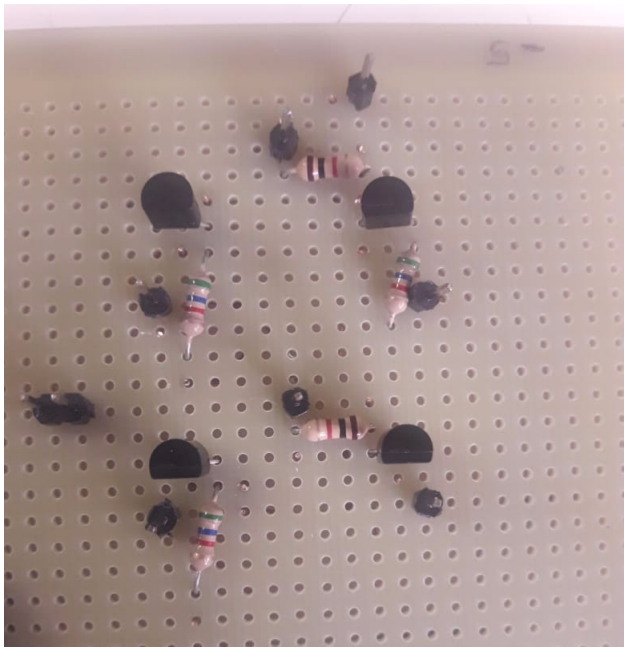


Fonte: Autoria própria, 2018

A circuito foi testado na *protoboard* e apresentou a resposta esperada, conforme a tensão aplicada sobre a base aumenta a corrente que segue pelo LED se intensifica, causando o aumento de brilho. Há dois controles aplicados ao LEDs, o controle lógico que é aplicado à base dos transistores PNP para selecionar qual LED será acionado e o de potência que controla a potência resultante sobre o LED com a alimentaçao de diferentes tensões sobre a base do transistor NPN. A figura 4.3 mostra o circuito confeccionado para o controle dos LEDs.

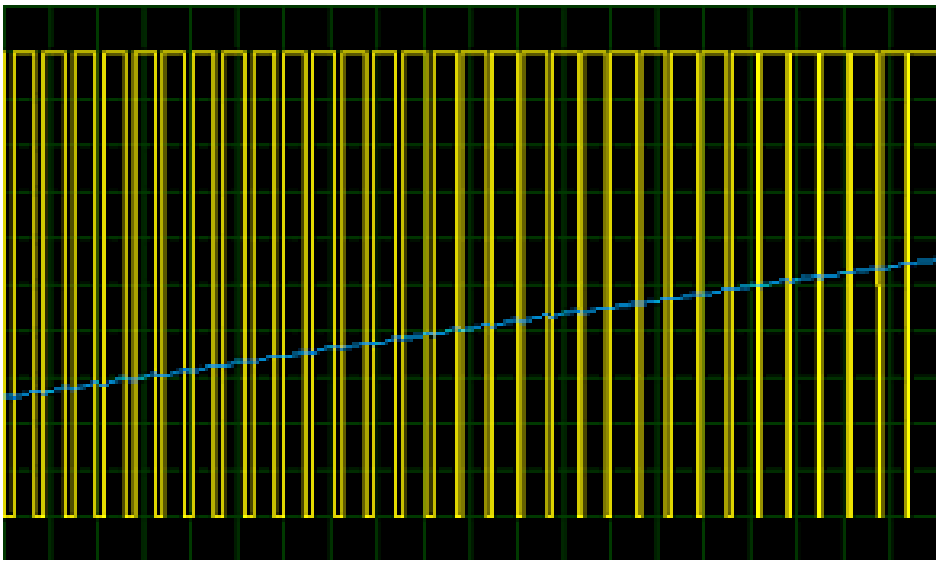
Para que nenhum componente fosse danificado, a corrente era limitada com os resistores de 5,6K. Para o controle digital da ordem em que os LEDs serão acessos foi utilizado o microcontrolador MSP430 G2553 em que os *timers* do dispositivo foram programados de acordo com o tempo em que os leds ficarão acessos, como representado na figura 4.4.

FIGURA 4.3 – Circuito de controle dos leds.



Fonte: Autoria própria, 2018

FIGURA 4.4 – Chaveamente dos leds.

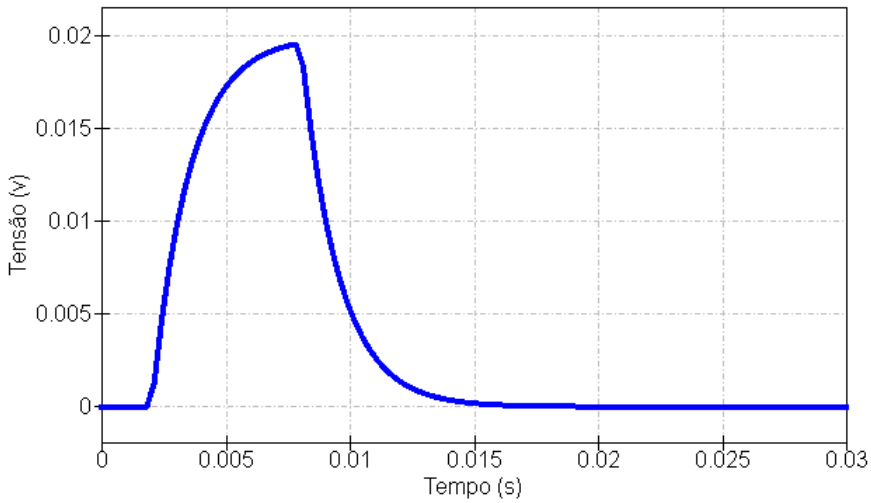


Fonte: Autoria própria, 2018

4.1.2 Amplificador de transimpedância

O amplificador de transimpedância foi simulado conforme é mostrado na figura 4.5, para simular o fotodiodo escolhido é aplicada uma fonte de corrente pulsada. O resultado da tensão de saída também é pulsado.

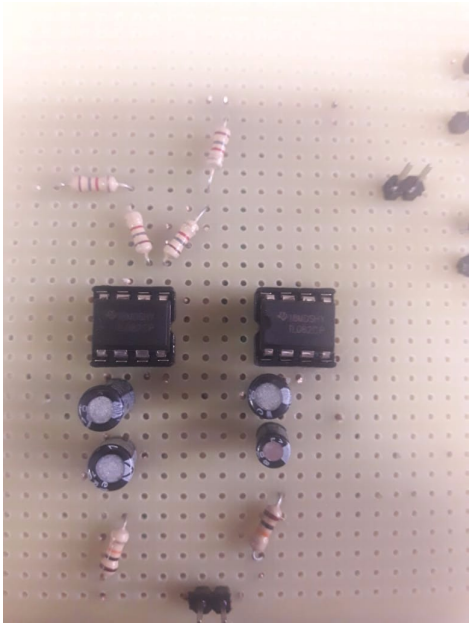
FIGURA 4.5 – Tensão de saída do amplificador de transimpedância.



Fonte: Autoria própria, 2018

O circuito foi confeccionado e está representado na figura 4.6.

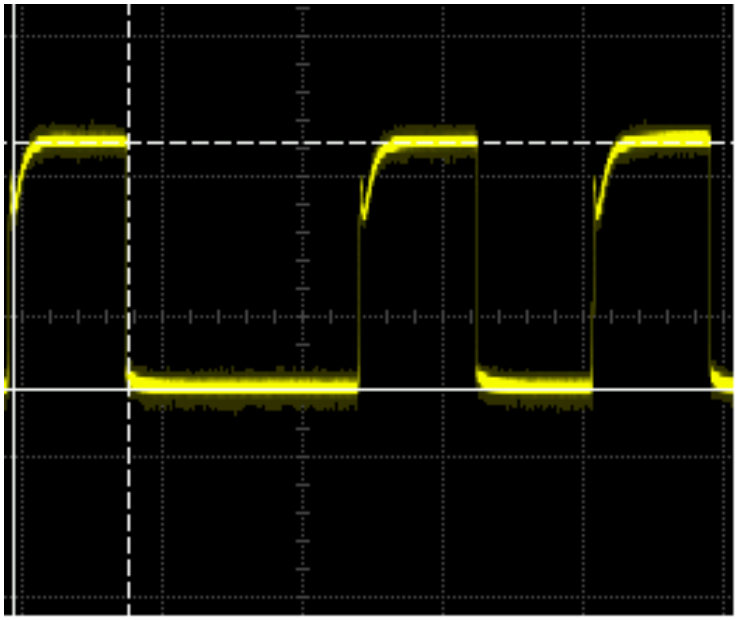
FIGURA 4.6 – Amplificador de transimpedância confeccionado.



Fonte: Autoria própria, 2018

Os resultados práticos do amplificador de transimpedância estão mostrados na figura 4.7, os testes foram feitos com a corrente gerada a partir de um transistor npn, somente para simular a característica de transformar corrente elétrica em tensão.

FIGURA 4.7 – Forma de onda do teste do amplificador de transimpedância.



Fonte: Autoria própria, 2018

A forma de onda da 4.7 apresenta uma leve diferença da forma de onda simulada na figura 4.5 devido a substituição do capacitor de 1pF para 1uF, assim o circuito trans-amplificador apresentará maior seletibilidade a frequências próximas ao batimento cardíaco. Como a taxa de batimento média no corpo humano é de 80 bpm, valor próximo de 1,5Hz, a frequência de corte adotada para o amplificador de transimpedância foi de 4Hz, assim a capacitância será dada pela equação 4.1.

$$C = \frac{1 + \sqrt{1 + 8\pi RfCifc}}{4\pi Rf(Ci + Cf)} \tag{4.1}$$

Sendo que Rf é o resistor utilizado no circuito, Ci é relacionado as capacitâncias de entrada dos amplificadores e do fotodiodo. Subsituindo os valores na fórmula tem-se a equação 4.2

$$C = \frac{1 + \sqrt{1 + 8\pi 10K25pF4}}{4\pi 10K(25pF)} \tag{4.2}$$

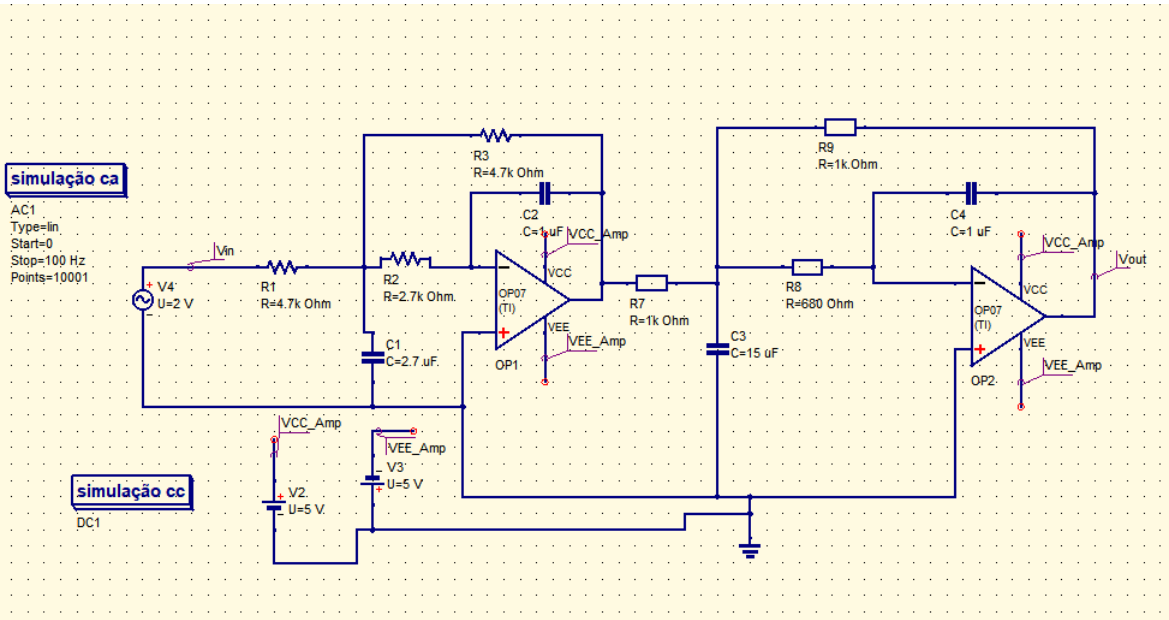
O resultado da equação 4.2 é de aproximadamente 3uF, assim a diferença entre as formas de onda é justificada devido a troca de capacitores.

4.1.3 Filtro passa baixa

A figura 4.8 expõe o circuito de simulação desenvolvido no *Quite Universal Circuit Simulator* (Qucs), como pode ser visto o filtro segue exatamente o que foi

projetado no desenvolvimento do projeto.

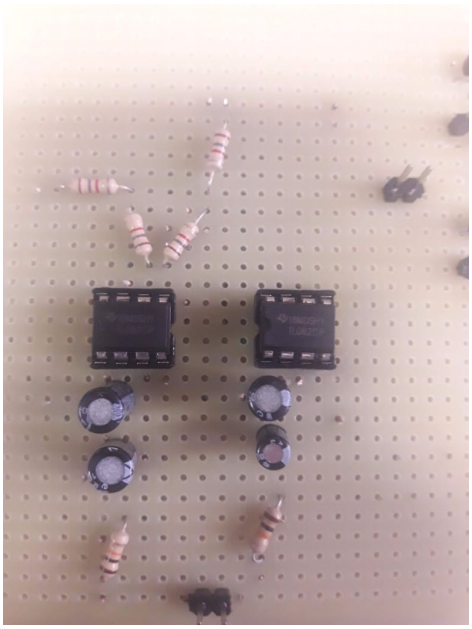
FIGURA 4.8 – Esquemático de simulação do filtro passa baixa.



Fonte: Autoria própria, 2018

A resposta em frequência do filtro é mostrado na figura 4.10, em que a frequência de corte resultante é de 48,45 Hz. A 4.9 representa a imagem do filtro após sua confecção

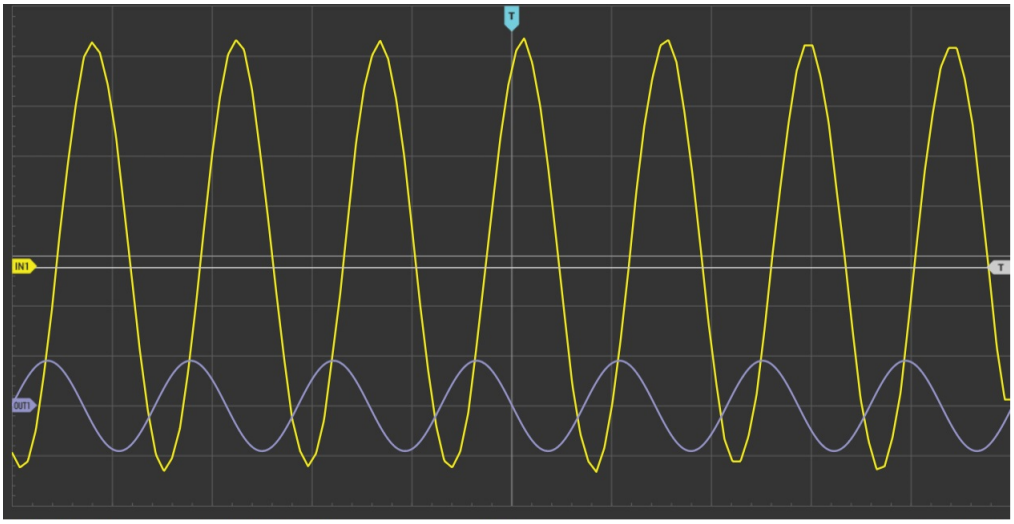
FIGURA 4.9 – Confecção do filtro.



Fonte: Autoria própria, 2018

A figura 4.10 ilustra as imagens do teste do filtro confeccionado. A forma de onda amarela representa a entrada aplicada ao filtro, enquanto que a forma azul indica a saída após sofrer atenuação.

FIGURA 4.10 – Confecção do filtro.



Fonte: Autoria própria, 2018

4.1.4 Amplificador do estágio final

O amplificador de estágio final tem como função realizar o ajuste para que a tensão de entrada não ultrapasse os limites da *general purpose input output* (GPIO) ocasionando problemas ao microcontrolador que fará a leitura.

Por se tratar de um componente que fará ajuste de tensão, sua simulação não representará resultados fidedignos, visto que sua calibração ocorrerá através de testes de bancada.

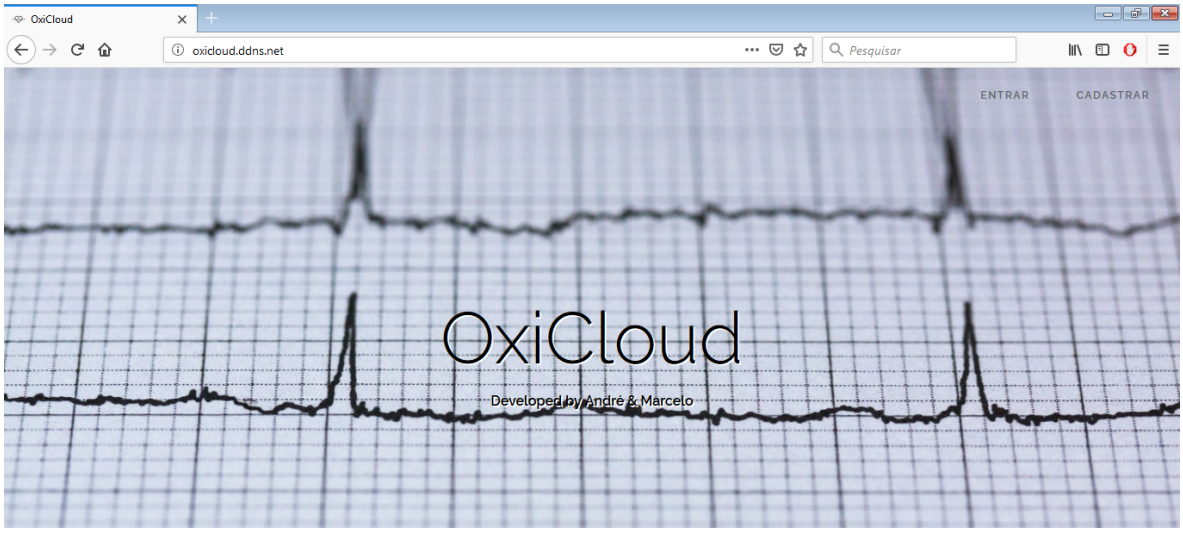
4.2 RESULTADOS DO PROJETO VIGENTE

Devido as mudanças que ocorreram durante o densenvolvimento do projeto, esse capítulo apresenta os resultados obtidos para o projeto vigente.

4.2.1 *Site*

Para ter acesso aos dados de parâmetros médicos do paciente que foram coletados pelo dispositivo é necessário acessar a página inicial oxicloud.ddns.net conforme a figura 4.11.

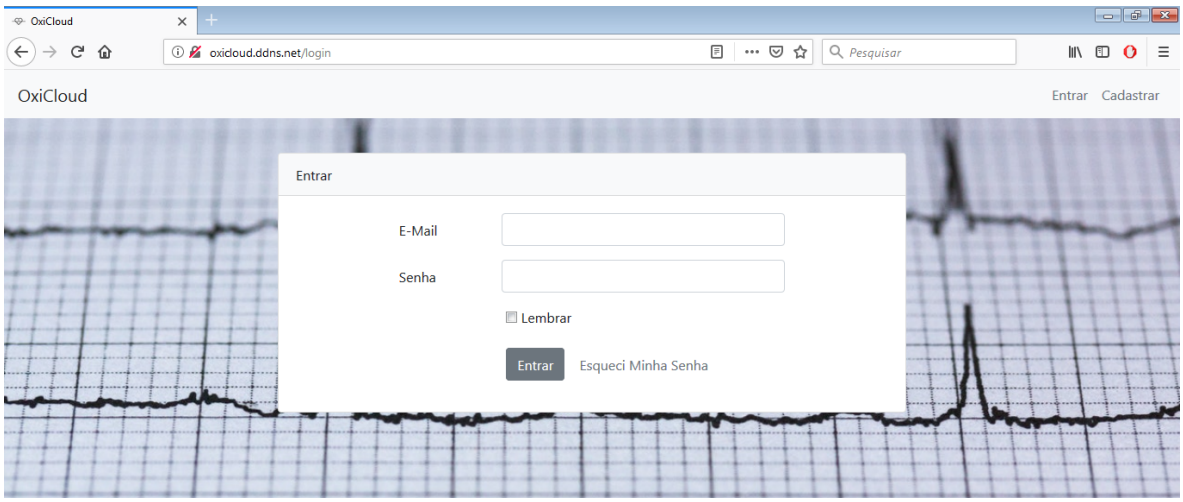
FIGURA 4.11 – Página inicial



Fonte: Autoria própria, 2018

A página inicial permite a entrada de um usuário cadastrados de acordo com figura 4.12, a qual exige inserção de *e-mail* e senha, além de conter a opção de recuperação de senha.

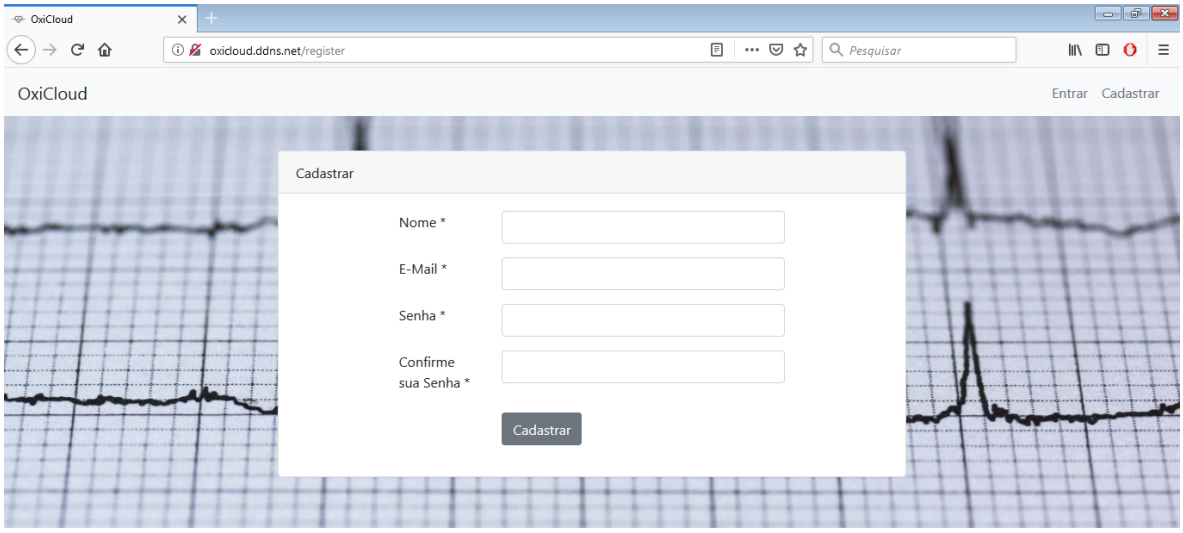
FIGURA 4.12 – Página de login



Fonte: Autoria própria, 2018

Não menos importante, há um menu para cadastramento de novos usuários na página incial conforme figura 4.13, a qual exige a inserção do nome do usuário, *e-mail*, senha e confirmação de senha.

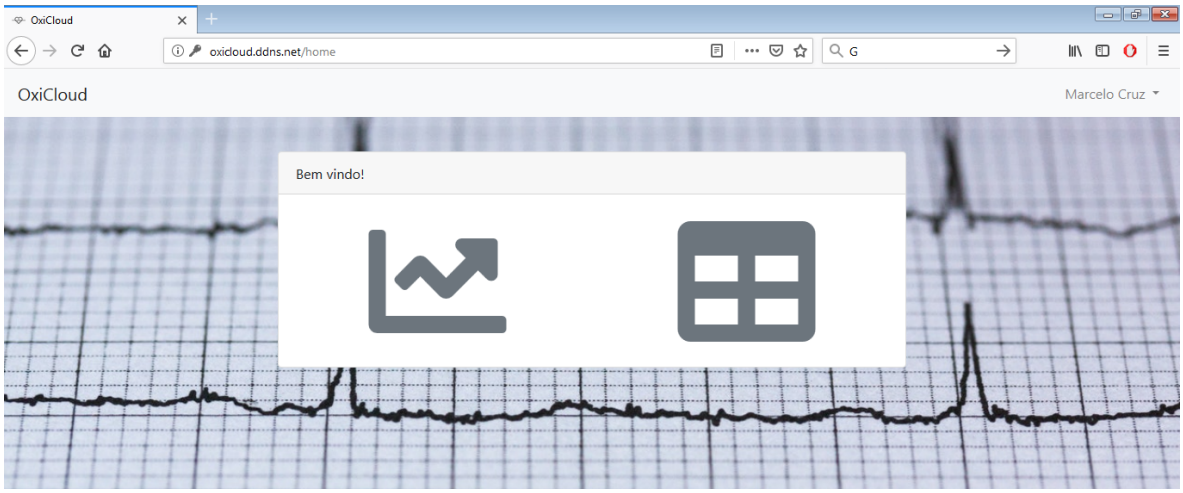
FIGURA 4.13 – Página de cadastro



Fonte: Autoria própria, 2018

De acordo com a figura 4.14, após realizar cadastro, o usuário tem acesso a uma página para visualização de um gráfico e uma tabela dos dados coletados.

FIGURA 4.14 – Página do usuário



Fonte: Autoria própria, 2018

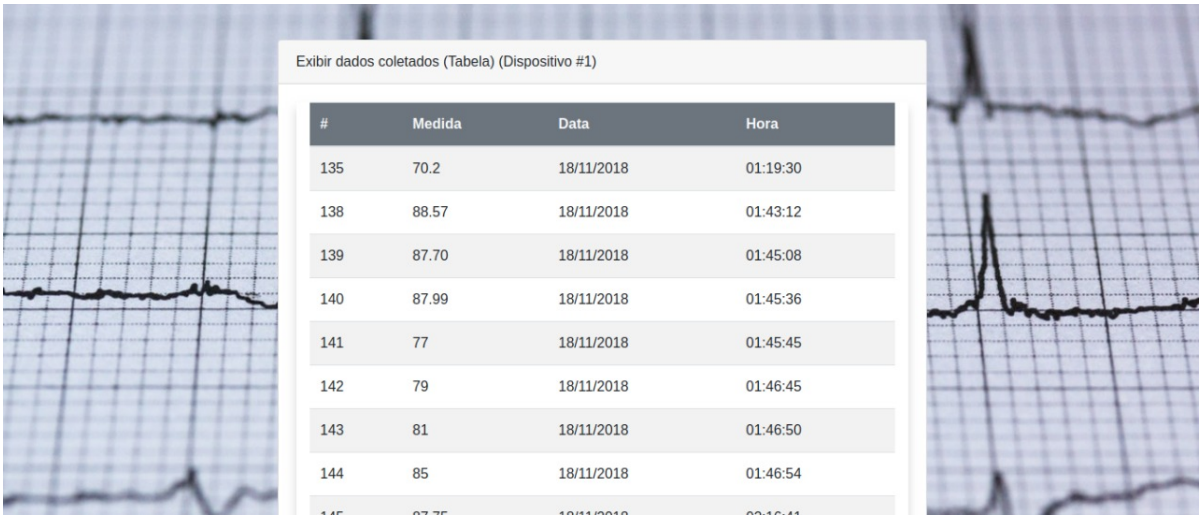
Conforme figura 4.15 e figura 4.16, há no site a opção de visualização de gráficos ou tabelas de medidas de sinais e que também exibem o número de medidas, o valor da medida mínima, máxima e média do parâmetro coletado.

FIGURA 4.15 – Gráfico de dados do paciente coletado pelo dispositivo

Fonte: Autoria própria, 2018

A figura 4.16 relaciona os dados apresentados no gráfica da figura 4.15 em forma de tabela, para visualização mais precisa de dados pontuais.

FIGURA 4.16 – Tabela de dados do paciente



Fonte: Autoria própria, 2018

4.2.2 Aplicativo

Conforme descrito no desenvolvimento do projeto o aplicativo *Android* utiliza o *webview* para acessar e mostrar os dados do sensores cadastrados para cada usuário.

A figura 4.17 ilustra a home sendo aberta no aplicativo *Oxicloud*.

FIGURA 4.17 – Home no aplicativo *mobile*

Fonte: Autoria própria, 2018

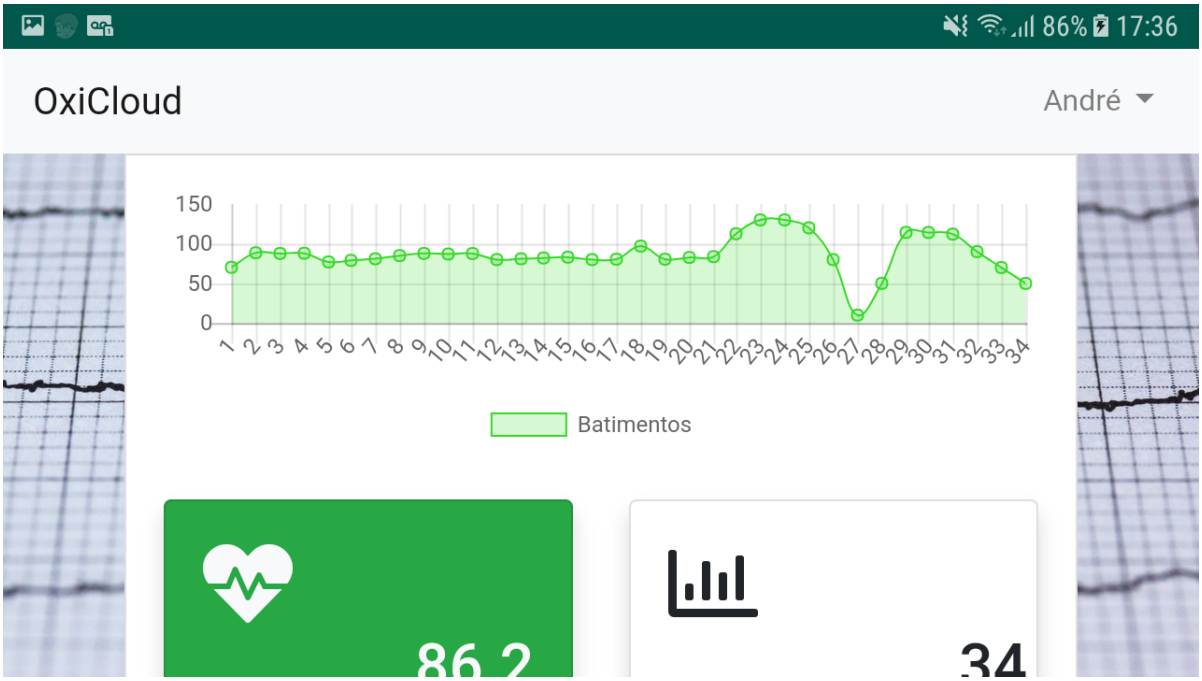
A figura 4.18 expõe o menu de escolha entre gráfico e tabela no aplicativo.

FIGURA 4.18 – Menu no aplicativo *mobile*

Fonte: Autoria própria, 2018

A figura 4.19 expressa o gráfico aberto no aplicativo *mobile*.

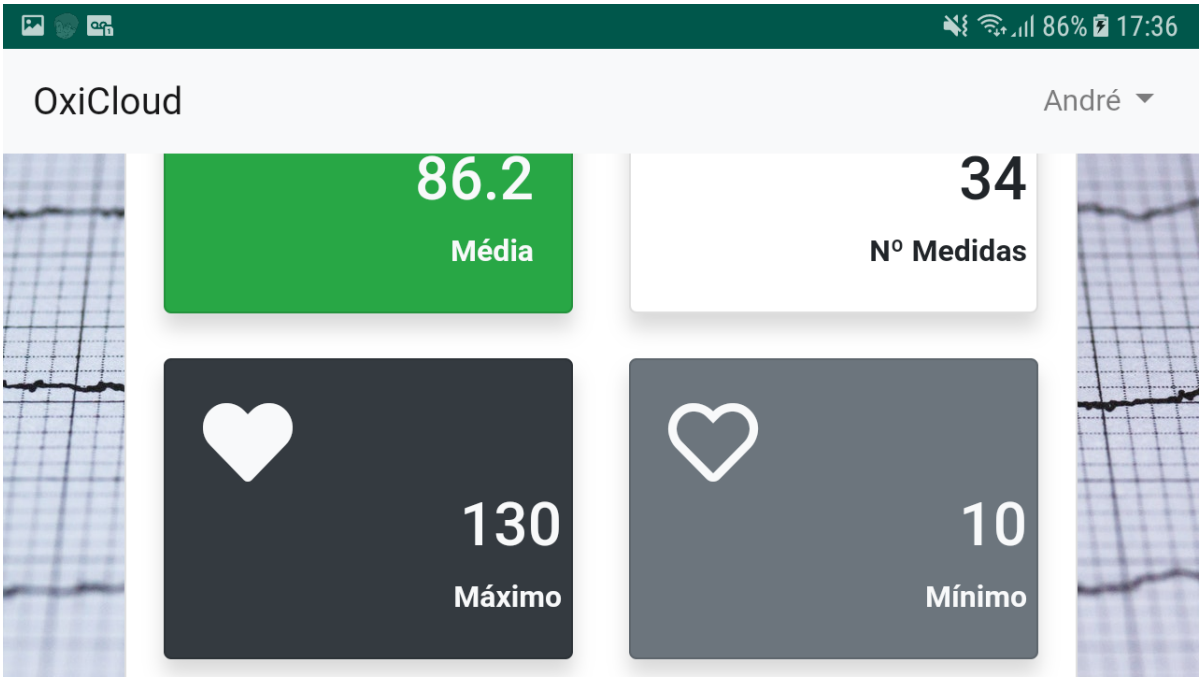
FIGURA 4.19 – Gráfico no aplicativo *mobile*



Fonte: Autoria própria, 2018

As médias são mostradas na figura 4.20.

FIGURA 4.20 – Médias no aplicativo *mobile*



Fonte: Autoria própria, 2018

Em casos que o dado sai de uma faixa de valores estipulada, uma mensagem é enviada para o *e-mail* da pessoa ligada ao sensor para informar da anormalidade do

dado, conforme indicado pela figura 4.21.

FIGURA 4.21 – Mensagem de alerta *mobile*

Fonte: Autoria própria, 2018

4.2.3 *Hardware*

O módulo que contém o sensor funcionaou de forma eficaz de acordo com o esperado, com taxa símbolo de 50 Hz, em nenhum momento houve sinal de interferência da rede elétrica ou de ruído sobre os dados lidos do módulo, comprovando o correto funcionamento dos filtros embarcados na programação do dispositivo.

4.2.3.1 MSP430

O microcontrolador MSP430 conseguiu estabilizar comunicação com o módulo MAX30100 através do periférico USCI no modo I2C, e também enviar os dados para o raspberry pi utilizando o módulo bluetooth empregando a comunicação serial UART.

4.2.3.2 *Raspberry*

O microcomputador consegue manter uma comunicação duradoura com o *bluetooth* do MSP430 e o envio de informações para a AWS.

Devido ao sistema operacional ser uma variante do linux o bluetooth é acessado através de scripts que executam linhas de comando no bash do sistema operacional. Após os dados serem acessados são validados através de lógica de validação numérica, em que ocorre a verificação se os dados recebidos são numeros. Caso a lógica resulte em respsota positiva há a abertura do canal de comunicação com o servidor, envio das informações e fechamento do canal.

4.2.3.3 Considerações finais dos resultados

Apesar das modificações realizadas com relação ao *hardware* inicial, a equipe foi capaz de realizar a concepção de um sistema de comunicação para servir de alerta em situações de emergência. O qual possibilita acesso remoto através de um computador ou de um sistema *mobile*, e que também dispõe de funcionalidades como *login* e cadastro de pacientes. Além de uma implementação para executar e enviar um alerta segurança através do *e-mail* ao responsável do paciente caso ocorra alguma disparidade de medição do enfermo. Portanto, a aplicação desse sistema agrega troca rápida e segurança de informação com confiabilidade de medição ao usuário.

5 CONCLUSÃO

O desenvolvimento do trabalho permitiu o recolhimento de informações sobre instrumentação médica, por lidar com pessoas a tecnologia necessita atender o mais alto nível de qualidade. O desafio está relacionado à aplicação de técnicas relacionadas a eletrônica da área óptica, como a aplicação de filtros e tratamento de sinais.

A etapa inicial do projeto envolvia o desenvolvimento da unidade de tratamento do sinal, banco de dados para armazenamento das informações e a ferramenta para acesso *mobile*. Os circuitos de tratamento como o amplificador de transimpedância, filtro passa baixa e o amplificador de estágio final foram simulados e testados em bancada e funcionaram de acordo com o esperado quando ensaiados individualmente, porém ao serem integrados em série o sinal era suprimido por interferência da rede. Outras topologias foram testadas a fim de verificar se a mudança poderia tratar o sinal de melhor forma, entretanto o mesmo problema ocorreu, os circuitos funcionaram adequadamente quando testados isoladamente, porém ao serem acoplados entre si o sinal era afetado por componentes de 60 Hz.

A fim de amenizar o tempo desperdiçado durante o desenvolvimento do *hardware* e entregar resultados sólidos a equipe alterou o foco do projeto e utilizou módulo comercial que encapsula os sistemas de tratamento do sinal oximétrico.

Apesar dos problemas encontrados com o desenvolvimento do *hardware*, a equipe obteve êxito na comunicação com o módulo MAX30100 e no envio de informações por *bluetooth* para o *raspberry pi*. A plataforma AWS se tornou uma opção viável pois disponibiliza um ano de acesso sem cobranças pelo uso com limitações de uso o que não limitou a execução do projeto.

Assim, como análise dos resultados entregues a equipe obteve êxito nos testes dos circuitos de *hardware* porém enfrentou problemas na integração, mas conseguiu estabelecer comunicação com o sensor e envio de informações para o *raspberry* e estabilizar a comunicação com o servidor presente na AWS. Assim como a disponibilização de uma plataforma para cadastramento e *login* de usuários, tanto por navegadores *web* quanto com ambiente *mobile* desenvolvendo o *frontend* e *backend* do servidor.

REFERÊNCIAS

ANDRADE, L. A. K. de. Sistema de medição para oximetria de pulso. 2009. Citado 2 vezes nas páginas 14 e 16.

ANVISA. *Home Care*. 2018. Http://portal.anvisa.gov.br/. Acesso em 21 de outubro de 2018. Citado na página 47.

ASSOCIATION, A. H. All about heart rate (pulse). 2018. Citado 2 vezes nas páginas 22 e 23.

AUGUSTO, V. M.; MACHADO, L. D. *Oximetria de pulso*. 2018. Https://sbpt.org.br/portal/publico-geral/doencas/oximetria-de-pulso/. Acesso em 17 de novembro de 2018. Citado na página 17.

BERNERS-LEE, T. 1998. Disponível em: <https://www.w3.org/History/1989/proposal.html>. Acesso em: 20 out 2018. Citado 2 vezes nas páginas 39 e 40.

BONFIM, M. J. d. C. *Filtros Ativos*. 2010. Departamento de Engenharia Elétrica. Universidade Federal do Paraná. Acesso em 14 junho 2018. Citado na página 28.

CIRCULATION, J. *Cardiovascular Health in Brazil*. 2016. Http://circ.ahajournals.org/content/133/4/422. Acesso em 23 março 2018. Citado na página 19.

CLARK, S. A. et al. *Design of pulse oximeters*. [S.l.]: Institute of Physics, 1997. Citado 5 vezes nas páginas 14, 15, 25, 26 e 51.

DEVELOPERS, A. *Activity*. 2018. Https://developer.android.com/reference/android/app/Activity. Acesso em 18 de outubro de 2018. Citado 2 vezes nas páginas 33 e 35.

DUCKETT, J. *Beginning HTML, XHTML, CSS, and JavaScript*. [S.l.]: Wrox, 2010. Citado na página 39.

EKLOT. *RaspberryPI models comparison*. 2018. Http://socialcompare.com/en/comparison/raspberrypi-models-comparison. Acesso em 17 de outubro de 2018. Citado na página 46.

FELIPE. *Sensor de Batimento Cardíaco*. 2010. Disponível em: <https://www.filipeflop.com/produto/sensor-de-batimento-cardiaco-e-oximetro-max30100/>. Acesso em: 20 out 2018. Citado na página 57.

FILHO, S. N. *Filtros seletores de sinais*. [S.l.]: editora UFSC, 2010. v. 1. Citado na página 29.

FOUNDATION, R. P. *About us*. 2012. Https://www.raspberrypi.org/about/. Acesso em 17 de outubro de 2018. Citado na página 45.

GONDOLFO, M. B. *Oximetria de pulso como sinal preditor de pneumonia: ferramenta útil na atenção primária?* 2018. Https://pebmed.com.br/oximetria-de-pulso-como-sinal-preditor-de-pneumonia-ferramenta-util-na-atencao-primaria/. Acesso em 17 de novembro de 2018. Citado na página 17.

HUANG, L. R. A. *Bluetooth for Programmers*. 2005. Acesso em 25 de março de 2018. Citado na página 32.

HUCH, J. S. J. R. S. K. F. H. Limitations of forehead pulse oximetry. *Journal of Clinical Monitoring*, 1994. Citado 2 vezes nas páginas 14 e 15.

INSTRUMENTS, T. *Mixed Signal Microcontroller*. 2011. Http://www.ti.com/lit/ds/symlink/msp430g2553.pdf. Acesso em 30 de março de 2018. Citado 2 vezes nas páginas 54 e 55.

____. *MSP430x2xx Family User's Guide*. 2013. Http://www.ti.com/lit/ug/slau144j/slau144j.pdf. Acesso em 17 de outubro de 2018. Citado na página 55.

JAVA. 2000. Https://www.oracle.com/index.html. Acesso em 25 de março de 2018. Citado 2 vezes nas páginas 35 e 36.

MARTOS, J. *Implemented LAMP stack client-server communication diagram.* 2018. Disponível em: <https://www.researchgate.net/figure/Implemented-LAMP-stack-client-server-communication-diagram_fig22_317311938>. Acesso em: 10 out 2018. Citado na página 60.

MATOS, V. *Using WebViews*. Cleveland State University, 2018. Disponível em: <http://grail.cba.csuohio.edu/~matos/notes/cis-493/lecture-notes/slides/Android-Chapter08-WebKit-Handout.pdf>. Acesso em: 21 out 2018. Citado na página 45.

MYSQL. *What is MySQL*. 2018. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>. Acesso em: 20 out 2018. Citado na página 44.

OGEDEGBE, G.; PICKERING, T. Principles and techniques of blood pressure measurement. 2013. Citado na página 22.

OMS, W. H. O. *Raised blood pressure*. 2018. Http://www.who.int/gho/ncd/risk_factors/blood_pressure_prevalence_text/en/. Acesso em 22 março 2018. Citado 2 vezes nas páginas 18 e 19.

OPAS, O. P.-A. da S. *Doenças cardiovasculares*. 2017. Http://www.paho.org/bra/index.php?option=com_contentview=articleid=5253:doencas-cardiovascularesItemid=839. Acesso em 22 de março de 2018. Citado na página 18.

OTWELL, T. 2015. Disponível em: <https://laravel.com/docs/4.2/introduction>. Acesso em: 20 out 2018. Citado 2 vezes nas páginas 42 e 43.

OXIMETRY. *Pulse Oximetry*. 2002. Https://web.archive.org/web/20150318054934/http://www.oximetry.org:80/pulseox/principles.htm. Acesso em 24 de março de 2018. Citado na página 17.

PERTENCE, A. *Amplificadores Operacionais e Filtros Ativos*. 2003. Editora Bookman. Citado na página 28.

RITCHIE, D. M. *The Development of the C Language*. 2018. Disponível em: <http://www.bell-labs.com/usr/dmr/www/chist.html>. Acesso em: Acesso em 22 de outubro de 2018. Citado na página 36.

SEDRA ADEL; SMITH, K. C. *Microelectronic Circuits*. [S.l.: s.n.], 1991. Citado 2 vezes nas páginas 27 e 31.

SERVICE, A. W. *Overview of Amazon Web Services AWS Whitepaper*. 2018. Https://docs.aws.amazon.com/aws-technical-content/latest/aws-overview/aws-overview.pdf?icmpid=link_from_whitepapers_page. Acesso em 21 de outubro de 2018. Citado 2 vezes nas páginas 46 e 47.

SIQUEIRA, A. de S. E.; SIQUEIRA-FILHO, A. G. de; LAND, M. G. P. Análise do impacto econômico das doenças cardiovasculares nos Últimos cinco anos no brasil. Scientific Electronic Library Online, 2017. Citado na página 20.

SOCIETY, A. T. *The Global Burden of Respiratory Disease*. 2013. Https://www.atsjournals.org/doi/abs/10.1513/AnnalsATS.201311-405PS. Acesso em 16 de outubro de 2018. Citado 2 vezes nas páginas 17 e 18.

SPURLOCK, J. *Bootstrap Responsive Web Development*. [S.l.]: O'Reilly Media, 2013. Citado na página 43.

TATROE, P. M. e. R. L. K. *Programming PHP*. [S.l.: s.n.], 2013. Citado na página 37.

TEIXEIRA, C. C. *VITAL SIGNS MEASUREMENT: AN INDICATOR OF SAFE CARE DELIVERED TO ELDERLY PATIENTS*. 2015. Scientific Electronic Library Online. Citado 2 vezes nas páginas 20 e 21.

UK, A. Principles of pulse oximetry). 2004. Citado na página 15.

UNIVERSITY, A. N. *Verilog inspired by the C programming languag*. [S.l.]: The Research School of Computer Science at the Australian National University, 2009. Citado na página 37.

WALKER, H. K.; HALL, W. D.; HURST, J. W. *Clinical Methods: The History, Physical, and Laboratory Examinations*. [S.l.]: Butterworths, 1990. Citado na página 23.

6 APÊNDICE

6.1 CÓDIGOS DO MICROCONTROLADOR MSP430

O código abaixo representa o algoritmo desenvolvido para o microcontrolador MSP430.

```
#include "msp430g2553.h"

char dado_rx;
unsigned int responda = 0, envia_leitura = 0;
unsigned int ocupado = 0;
unsigned int cont250ms = 0, debounce = 0;

void tx_frase(char * frase);
void tx_bits(unsigned long frase);
void tx_byte(char dado);
//char nome[] = "AT+NAMEOxiCloud0002\r\n";
//char pin[] = "AT+PIN12345\r\n";

// recebe um caracter pela interface serial. Guarda este valor
//na variavel 'dado_rx' Quando um caracter chega pela interface
// serial, gera uma interrupcao e armazena na variavel dado_rx
#pragma vector=USCIAB0RX_VECTOR
__interrupt void rx_uart()
{
    __disable_interrupt(); // desabilita as interrupcoes
    dado_rx = UCA0RXBUF;
    tx_byte(dado_rx);
    IFG2 &= ~UCA0RXIFG;
    __enable_interrupt(); // habilita as interrupcoes
}
```

```
/**
 * FUNCAO:
 * transmite um caracter pela interface serial
 */
void tx_byte(char dado)
{
    // espera enquanto a interface serial
    //esta' ocupada (enviando um dado)
    while(!(IFG2 & UCA0TXIFG));
    UCA0TXBUF = dado;   // inicia a transmissao do dado
}

/**
 * FUNCAO:
 * transmite BITS pela interface serial
 */
void tx_bits(unsigned long frase)
{

    int i=0;
    // espera enquanto a interface serial
    //esta' ocupada (enviando um dado)
    while(!(IFG2 & UCA0TXIFG));
    UCA0TXBUF = frase>>i; // inicia a transmissao do dado

}

/**
 * Transmite uma string pela interface
 serial, usando a funcao tx_byte();
 */
void tx_frase(char * frase)
{
```

```
    int i=0;
    while(ocupado);
    ocupado = 1;
    // enquanto nao encontra o fim da string ('\0')
    //transmite o caracter da posicao i
    for(i=0;*(frase+i);i++)
        tx_byte(*(frase+i));
    ocupado = 0;
}

/**
 * Laco principal
 */
void main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

    // Configurando o clock em 16MHz. Ajusta o clock para
    //16MHz para gerar a temporizacao correta para
    //a interface serial
    BCSCTL1 = CALBC1_16MHZ;
    DCOCTL = CALDCO_16MHZ;

    // Configurando a interface serial em 9600, 8n1
    UCA0CTL1 |= UCSWRST;          // Coloca UART em reset


    // Sem paridade, bit menos
    //significativo primeiro, 8 bits de dados,
    //1 stop bit, modo UART, modo assincrono
    UCA0CTL0 = 0x00;

    // Seleciona SMCLK como o sinal
```

```
//de clock da interface serial
UCA0CTL1 |= UCSSEL_2;

UCA0BR0 = 0x82;             // Seleciona taxa de 9600
UCA0BR1 = 0x06;             // Seleciona taxa de 9600
UCA0MCTL = 0x0C;            // Seleciona taxa de 9600

// Configurando timer A0 (debouce)
TA0CTL = TASSEL_2+ //SMCLK
         ID_3+     //Div8
         MC_1+     //up mode
         TAIE;     //habilita interrupção
//SMCLK em 16MHz. Div8, Clock do timer em 2MHz
//Preciso de 0.5s (0.001s com contador ate 500).
//Então preciso que o up conte até:
//0.001/(1/2000000) = 2000

TA0CCR0 = 2000;

// Configura pinos de RX e TX
// Seleciona a funcao de TX e RX da
//serial para os pinos P1.1 e P1.2
P1SEL |= BIT1 + BIT2;
// Seleciona a funcao de TX e RX da
//serial para os pinos P1.1 e P1.2
P1SEL2 |= BIT1 + BIT2;

/*****SAIDAS******/
//Configura os leds como saída,
//escrevendo 1 nos bits correspondentes em P1DIR
P1DIR |= (BIT0 + BIT6);
P1OUT &= ~(BIT0 + BIT6);
```

```
    /*****ENTRADA******/
    P1DIR &= ~(BIT3);
    P1OUT |=  (BIT3); // Configura para ser resistor de pullup
    P1REN |=  (BIT3); // Liga o resistor de pullup
    P1IE  |=  (BIT3); // Ativa a interrupção
    P1IES &= ~(BIT3);
    // interrupção ativada na borda de subida
    P1IFG &= ~(BIT3); // Limpa a flag da interrupcao

    UCA0CTL1 &= ~UCSWRST;     // Retira UART do reset

    // configura interrupcaoes da interface serial (recepcao)
    IE2 |= BIT0;              // Ativa a interrucao da interface
    //serial (recepcao)
    IFG2 &= ~UCA0RXIFG;       // limpa o flag de interrupcao da
    //interface serial (recepcao)

    __enable_interrupt();   // habilita as interrupcoes
/*
    //Configura modulo Bluetooth
    tx_frase("AT\r\n");
    P1OUT |= (BIT0);
    while(!responda);
    responda = 0;
    P1OUT |= (BIT6);

    tx_frase(nome);
    P1OUT &= ~(BIT6);
    while(!responda);
    responda = 0;
    P1OUT |= (BIT6);

    tx_frase(pin);
```

```
        P1OUT &= ~(BIT6);
        while(!responda);
        responda = 0;
        P1OUT |= (BIT6);
    */
        P1OUT &= ~(BIT0 + BIT6);
        while (1){
            if (responda){
                tx_frase("Recebido\r\n");
                responda = 0;
            }
            if (envia_leitura){
                envia_leitura = 0;
                tx_frase("Lido\r\n");
            }
        }
    }

    #pragma vector=TIMER0_A1_VECTOR
    __interrupt void TA0_IST(void){
        __disable_interrupt();   // desabilita as interrupcoes
        TA0CTL &= ~TAIFG;
        //vai entrar aqui a cada 0.001s
        cont250ms++;
        if (cont250ms >= 250){
            cont250ms = 0;
            if (debounce <= 2){
                debounce++;
            }
        }

        __enable_interrupt();   // habilita as interrupcoes
    }
```

```
#pragma vector=PORT1_VECTOR
__interrupt void port1_int(void)
{
    //__disable_interrupt();   // desabilita as interrupcoes

    if (P1IFG & BIT3 && debounce >= 2){
    //interrupção do gatilho 1
        debounce = 0;
        envia_leitura = 1;
        P1OUT ^= (BIT0 + BIT6);
    }
    P1IFG = 0; // Limpa flag da interrupção
    //__enable_interrupt();   // habilita as interrupcoes
    return;
}
```

6.2 CÓDIGOS DO *RASPBERRY*

O código do raspberry para aquisição dos dados do *bluetooth* é exibido abaixo.

```
<?php
    $ttyPath = '/dev/rfcomm0'; // Porta bluetooth
        hile(1){
    try{
      $ttyHandler = fopen($ttyPath, 'rb');
      // Abrindo a conexão bluetooth
      $bytes = rtrim(fgets($ttyHandler, 128));
      try{
        fclose($ttyHandler); // Fechando a conexão.
        if (is_numeric ( $bytes )){
                $my_var = file_get_contents(
                "http://52.67.52.222/inserirDado/1/1/".$bytes);
        }
```

```
      }
      catch(\Exception $e){
         echo 'erro';
      }
    }
    catch(\Exception $e){
       echo 'erro';
    }
        }
?>
```

6.3 CÓDIGOS *AMAZON WEB SERVICE*

Por se tratar de vários scripts desenvolvidos em diferentes linguagens somente os trechos mais importantes serão exibidos nos próximos tópicos.

Os códigos não exibidos são similares aos inseridos no relatório, ou são muito extensos. Os subtópicos a seguir apresentam trechos do *Frontend* e *Backend*.

6.3.1 Frontend

Abaixo estão os códigos relacionados ao *frontend* do projeto.

6.3.1.1 *Home* do aplicativo

```
<!doctype html>

<html lang="{{ app()->getLocale() }}">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>{{ config('app.name', 'OxiCloud') }}</title>

        <!-- Fonts -->
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com
        /bootstrap/4.1.3/
```

```
css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27
NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous"><link rel
="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/
css/all.css"integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZT
Y5qE0Id1GSseTk6S+L3BlXeVIU" crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzp
bzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<link href="https://fonts.googleapis.com/css?family=Raleway:300,600"
rel="stylesheet" type="text/css">

<!-- Styles -->
<style>
    html, body {
        background-color: #fff;
        color: #000;
        font-family: 'Raleway', sans-serif;
        font-weight: 100;
        height: 100vh;
        margin: 0;
    }

    .full-height {
        height: 100vh;
    }

    .flex-center {
        align-items: center;
        display: flex;
        justify-content: center;
    }

    .position-ref {
        position: relative;
```

```
}

.top-right {
    position: absolute;
    right: 10px;
    top: 18px;
}

.content {
    text-align: center;
}

.title {
    font-size: 84px;
}

.links > a {
    color: #636b6f;
    padding: 0 25px;
    font-size: 12px;
    font-weight: 600;
    letter-spacing: .1rem;
    text-decoration: none;
    text-transform: uppercase;
}

.m-b-md {
    margin-bottom: 30px;
}
.img{
    background-image: url("{{ asset('imagens/heart.jpeg') }}");
    background-repeat:no-repeat;
    background-size:cover;
    bottom: 0;
    color: black;
```

```
            left: 0;
            overflow: auto;
            padding: 3em;
            position: fixed;
            right: 0;
            text-align: center;
            top: 0;
        }
    </style>
</head>
<body>
  <div class="img"></div>
    <div class="flex-center position-ref full-height">
        @if (Route::has('login'))
            <div class="top-right links">
                @auth
                    <a href="{{ url('/home') }}">INÍCIO</a>
                @else
                    <a href="{{ route('login') }}">ENTRAR</a>
                    <a href="{{ route('register') }}">CADASTRAR</a>
                @endauth
            </div>
        @endif

        <div class="content">
            <div class="title m-b-md my-0" style="text-shadow:
            0.02em 0.02em #fff">
                OxiCloud
            </div>

            <h6 class="mt-0" style="text-shadow: 0.1em 0.1em #fff">
              <b>
                Developed by André & Marcelo
              </b>
            </h6>
```

```
            </div>
        </div>
        <script>
            function mostrarToast(texto){
                Android.mostrarToast(texto);
            }
            mostrarToast("Seja Bem Vindo ao OxiCloud!\nRealize login para acessar
            o App!");
        </script>
    </body>
</html>
```

6.3.1.2 Visualização da tabela

```
@extends('layouts.app')

@section('content')
<div class="container">
    <div class="row">
<div class="offset-md-2 col-md-8">
  @foreach($dispositivos as $dispositivo)
    <div class="card">
        <div class="card-header">
        Exibir dados coletados (Tabela)
        (Dispositivo #{{ $dispositivo->id }})</div>

        <div class="card-body">
            @if (session('status'))
                <div class="alert alert-success">
                    {{ session('status') }}
                </div>
            @endif

            <div class="col-md-12 mb-4 table-responsive
            table-striped shadow">
```

```
                <table class="table table-hover">
                  <thead>
                    <tr class="bg-secondary text-light">
                      <th scope="col">#</th>
                      <th scope="col">Batimentos</th>
                      <th scope="col">Oxigenação</th>
                      <th scope="col">Data</th>
                      <th scope="col">Hora</th>
                    </tr>
                  </thead>
                  <tbody>
                    @foreach($dispositivo->dados as $dado)
                        <tr>
                          <td>{{ $dado->id }}</td>
                          <td>{{ $dado->batimento }}</td>
                          <td>{{ $dado->oxigenacao }}</td>
                          <td>{{ $dado->created_at->
                          ('d/m/Y') }}</td>
                          <td>{{ $dado->created_at->
                          format('h:i:s') }}</td>

                        </tr>
                    @endforeach
                  </tbody>
                </table>
              </div>
          </div>
      </div>
    @endforeach
          </div>
      </div>
</div>
@endsection
```

6.3.2 Visualização do gráfico

```
@extends('layouts.app')

@section('content')
<meta name="viewport" content="width=device-width, initial-scale=1,
user-scalable=no">
<style type="text/css">
    #warning-message { display: none; }
    @media only screen and (orientation:portrait){
        #wrapper { display:none; }
        #warning-message { display:block; }
    }
    @media only screen and (orientation:landscape){
        #warning-message { display:none; }
    }
</style>
<div id="warning-message">
  <div class="row">
      <div class="col-md-12">
          <div class="card">
            <div class="card-header bg-danger text-light">
              Atenção
            </div>
            <div class="card-body">
              <div class="alert alert-danger" role="alert">
                Esta página é visivel somente no modo Paisagem!
                Entre novamente no aplicativo com o celular no modo paisagem!
              </div>
            </div>
          </div>
      </div>
  </div>
</div>
<div class="container" id="wrapper">
```

```
<div class="row">
    <div class="col-md-12">
      @foreach($dispositivos as $dispositivo)
        <div class="card">
            <div class="card-header">Exibir dados coletados (BATIMENTOS)
            (Gráfico) (Dispositivo #{{ $dispositivo->id }})</div>

            <div class="card-body">
                @if (session('status'))
                    <div class="alert alert-success">
                        {{ session('status') }}
                    </div>
                @endif

                  <script src="https://cdnjs.cloudflare.com/ajax/libs/
                  Chart.js/2.7.2/Chart.min.js"></script>

                        <canvas id="dispositivo_{{ $dispositivo->id }}"
                        width="400" height="120%"></canvas>

                  <script>
                  @php
                    $media = 0;
                    $maximo = 0;
                    $minimo = 300;
                    $numero_medidas = 0;
                  @endphp
                  var ctx = document.getElementById("dispositivo_{{
                  $dispositivo->id }}").getContext('2d');
                  var graficoBatimentos = new Chart(ctx, {
                      type: 'line',
                      data: {
                          labels: [
                            @foreach($dispositivo->dados as $dado)
                              @php
```

```
                $media+=$dado->batimento;
                if ($dado->batimento < $minimo) {
                  $minimo = $dado->batimento;
                }
                if ($dado->batimento > $maximo) {
                  $maximo = $dado->batimento;
                }
                $numero_medidas++;
              @endphp
              "{{ $numero_medidas }}",
            @endforeach
          ],
          datasets: [{
              label: 'Batimentos',
              data: [
                @foreach($dispositivo->dados as $dado)
                  {{ $dado->batimento }},
                @endforeach
              ],
              backgroundColor:'rgba(52, 218, 37, 0.2)',
              borderColor:'rgba(52, 218, 37, 1)',
              borderWidth: 1
          }]
      },
      options: {

          legend: {
              display: true,
              position: 'bottom',
          }
      }
});
</script>

@php
```

```
                        $media /= $numero_medidas;
                        $media = number_format($media, 1);
                    @endphp

<div class="row mt-4">
<div class="col-md-3 col-sm-6 col-xs-6 mb-4">
<div class="card bg-success shadow" style="padding: 20px;  ">
    <div class="media widget-ten row">
        <div class="media-left media media-middle col-md-auto">
            <span><i class="fas fa-3x fa-heartbeat text-light"></i></span>
        </div>
        <div class="media-body col-md-auto" style="text-align: right;">
    <h2 class=" text-light"> {{ $media }} </h2>
            <p class=" text-light" style="margin-bottom: 0px!important;">
            <b>Média </b></p>
        </div>
    </div>
</div>
</div>
<div class="col-md-3 col-sm-6 col-xs-6 mb-4">
<div class="card bg-pink shadow" style="padding: 20px; ">
<div class="media widget-ten row">
 <div class="media-left media media-middle col-md-auto">
    <span><i class="far fa-3x fa-chart-bar"></i></span>
</div>
<div class="media-body" style="text-align: right;">
     <h2 class="color-white"> {{ $numero_medidas }} </h2>
     <p style="margin-bottom: 0px!important;"><b>Nº Medidas </b></p>
</div>
</div>
</div>
</div>
<div class="col-md-3 col-sm-6 col-xs-6 mb-4">
<div class="card bg-dark shadow" style="padding: 20px; ">
<div class="media widget-ten row">
```

```
 <div class="media-left media media-middle col-md-auto">
    <span><i class="fas fa-3x fa-heart text-light"></i></span>
</div>
<div class="media-body" style="text-align: right;">
     <h2 class="text-light"> {{ $maximo }} </h2>
     <p class="text-light" style="margin-bottom: 0px!important;">
     <b>Máximo </b></p>
</div>
</div>
</div>
</div>
<div class="col-md-3 col-sm-6 col-xs-6 mb-4">
<div class="card bg-secondary shadow" style="padding: 20px; ">
 <div class="media widget-ten row">
     <div class="media-left media media-middle col-md-auto">
        <span><i class="far fa-3x fa-heart text-light"></i></span>
    </div>
    <div class="media-body" style="text-align: right;">
         <h2 class="text-light"> {{ $minimo }} </h2>
         <p class="text-light" style="margin-bottom:
         0px!important;"><b>Mínimo </b></p>
    </div>
</div>
</div>
</div>
</div>
</div>
            </div>
            <br>
            <div class="card">
                <div class="card-header">Exibir dados coletados (OXIGENAÇÃO)
                (Gráfico) (Dispositivo #{{ $dispositivo->id }})</div>

                <div class="card-body">
                    @if (session('status'))
```

```
            <div class="alert alert-success">
                {{ session('status') }}
            </div>
        @endif

<script src="https://cdnjs.cloudflare.com/ajax/libs/
Chart.js/2.7.2/Chart.min.js"></script>

  <canvas id="dispositivo_oxigenacao_{{ $dispositivo->id }}"
  width="400" height="120%"></canvas>

        <script>
        @php
          $media = 0;
          $maximo = 0;
          $minimo = 300;
          $numero_medidas = 0;
        @endphp
var oxg = document.getElementById("dispositivo_oxigenacao
_{{$dispositivo->id }}").getContext('2d');
        var graficoOxigenacao = new Chart(oxg, {
            type: 'line',
            data: {
                labels: [
                  @foreach($dispositivo->dados as $dado)
                    @php
                      $media+=$dado->oxigenacao;
                      if ($dado->oxigenacao < $minimo) {
                        $minimo = $dado->oxigenacao;
                      }
                      if ($dado->oxigenacao > $maximo) {
                        $maximo = $dado->oxigenacao;
                      }
                      $numero_medidas++;
                    @endphp
```

```
                        "{{ $numero_medidas }}",
                      @endforeach
                    ],
                    datasets: [{
                        label: 'Oxigenação',
                        data: [
                          @foreach($dispositivo->dados as $dado)
                            {{ $dado->oxigenacao }},
                          @endforeach
                        ],
                        backgroundColor:'rgba(0, 0, 255, 0.2)',
                        borderColor:'rgba(0, 0, 255, 1)',
                        borderWidth: 1
                    }]
                },
                options: {

                    legend: {
                        display: true,
                        position: 'bottom',
                    }
                }
            });
            </script>

            @php
              $media /= $numero_medidas;
              $media = number_format($media, 1);
            @endphp

            <div class="row mt-4">
              <div class="col-md-3 col-sm-6 col-xs-6 mb-4">
        <div class="card bg-primary shadow" style="padding: 20px;  ">
<div class="media widget-ten row">
<div class="media-left media media-middle col-md-auto">
```

```
<span><i class="fas fa-3x fa-heartbeat text-light"></i></span>
</div>
<div class="media-body col-md-auto" style="text-align: right;">
<h2 class=" text-light"> {{ $media }} </h2>
 <p class=" text-light" style="margin-bottom: 0px!important;"><b>Média </b></p>
</div>
</div>
</div>
</div>
<div class="col-md-3 col-sm-6 col-xs-6 mb-4">
<div class="card bg-pink shadow" style="padding: 20px; ">
<div class="media widget-ten row">
<div class="media-left media media-middle col-md-auto">
<span><i class="far fa-3x fa-chart-bar"></i></span>
</div>
<div class="media-body" style="text-align: right;">
 <h2 class="color-white"> {{ $numero_medidas }} </h2>
 <p style="margin-bottom: 0px!important;"><b>Nº Medidas </b></p>
</div>
</div>
</div>
</div>
<div class="col-md-3 col-sm-6 col-xs-6 mb-4">
<div class="card bg-dark shadow" style="padding: 20px; ">
<div class="media widget-ten row">
 <div class="media-left media media-middle col-md-auto">
    <span><i class="fas fa-3x fa-heart text-light"></i></span>
</div>
<div class="media-body" style="text-align: right;">
     <h2 class="text-light"> {{ $maximo }} </h2>
     <p class="text-light" style="margin-bottom: 0px!important;">
     <b>Máximo </b></p>
</div>
</div>
</div>
```

```
</div>
<div class="col-md-3 col-sm-6 col-xs-6 mb-4">
<div class="card bg-secondary shadow" style="padding: 20px; ">
<div class="media widget-ten row">
 <div class="media-left media media-middle col-md-auto">
    <span><i class="far fa-3x fa-heart text-light"></i></span>
</div>
<div class="media-body" style="text-align: right;">
     <h2 class="text-light"> {{ $minimo }} </h2>
     <p class="text-light" style="margin-bottom: 0px!important;">
     <b>Mínimo </b></p>
</div>
</div>
</div>
</div>
                        </div>
                    </div>
                </div>
            @endforeach


        </div>
    </div>
</div>
@endsection
```

6.4 CÓDIGO DO APLICATIVO *ANDROID*

Abaixo está a representação do código d *Manifest.xml*.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tcc.ufpr.tccb9">



<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity"
            android:screenOrientation="landscape"
            android:configChanges="keyboardHidden|orientation|screenSize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

O trecho de código abaixo é correspondente ao arquivo *MainActivity* onde é executada a *webview*.

```
package tcc.ufpr.tccb9;

import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.os.Build;
import android.support.annotation.RequiresApi;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebSettings;
```

```
import android.webkit.WebView;
import android.webkit.WebViewClient;

import java.util.Objects;

import tcc.ufpr.tccb9.javascript.JsInterface;

public class MainActivity extends AppCompatActivity {
    private final int REQUEST_PERMISSIONS_CODE = 123;
    private WebView webView;

    @TargetApi(Build.VERSION_CODES.M)
    @RequiresApi(api = Build.VERSION_CODES.M)
    @SuppressLint("SetJavaScriptEnabled")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            Objects.requireNonNull(getSupportActionBar()).hide();
            //Remove a barra de títulos
        } else {
            getSupportActionBar().hide();
        }
        setContentView(R.layout.activity_main);

        // Requisita permissoes de localizacao


        webView = findViewById(R.id.webView);
        webView.addJavascriptInterface(new JsInterface(this), "Android");

        WebSettings wsLogin = webView.getSettings();
        wsLogin.setJavaScriptEnabled(true);
```

```
        // ativa o suporte a javascript
        wsLogin.setJavaScriptCanOpenWindowsAutomatically(true);
        wsLogin.setSupportZoom(false);
        // não deixa o usuário utilizar o zoom
        wsLogin.setAppCacheEnabled(true);
        wsLogin.setDatabaseEnabled(true);
        wsLogin.setDomStorageEnabled(true);
        wsLogin.setGeolocationDatabasePath(getFilesDir().getPath());

        webView.loadUrl("http://oxicloud.ddns.net");
        webView.setWebViewClient(new WebViewClient());}
        //com esse método podemos navegar (clicando nos links) sem sair
        //do app, ou seja, os links não são redirecionados
        //para o navegador do celular

    @Override
    //Aqui voltamos para página anterior quando o
    //usuário clica no botão return do
    //celular, ao invés de fechar o app
    public void onBackPressed() {
        if (webView.canGoBack()) {
        //verifica se existe uma página para voltar
            webView.goBack();
        } else {
            super.onBackPressed();
        }
    }
}
```