Monitoramento de Sinais Vitais no Ambiente da Telemedicina

Felipe Bolzan Dela Libera[1], Alessandro André Mainardi de Oliveira[1]

[1]Curso de Bacharelado em Ciência da Computação – Universidade Franciscana
97010-032 – Santa Maria – RS – Brasil

felipe.bolzan@hotmail.com, alessandroandre@ufn.edu.br

Abstract. *This article describes the development of the prototype idea presented by Anderson Ribas, in the development of a system for remote monitoring of heart rate, oxygen saturation and body temperature of a patient in remote care. To measure these parameters, Max30102 and NTC PT100 sensors were used. Data is grouped and sent via Wi-Fi from the ESP8266 microcontroller using HTTP protocol. For storage, the MySQL database was used. With validations using comparisons with equipment registered with Inmetro.*

Resumo. *Este artigo descreve o desenvolvimento da ideia de protótipo apresentada por Anderson Ribas, no desenvolvimento de um sistema para monitoramento remoto de frequência cardíaca, saturação de oxigênio e temperatura corporal de um paciente em atendimento à distância. Para medição desses parâmetros, foram utilizados os sensores Max30102 e NTC PT100. Os dados são agrupados e enviados via Wi-Fi do microcontrolador ESP8266 utilizando protocolo HTTP. Para armazenamentos utilizou-se do banco de dados MySQL. Com validações utilizando comparações com equipamentos registrados no Inmetro.*

Palavras-chave: *Saúde à distância, Telemedicina, ESP8266, MAX30102, Batimentos Cardíacos, SpO2, Sinais Vitais*

1. Introdução

A tecnologia na saúde está mudando rapidamente o setor de assistência médica, melhorando os resultados para profissionais e pacientes. De acordo com Morsch (2020), avanços consistentes em produtos farmacêuticos e na área médica já salvaram milhões de vidas e melhoraram muitas outras. É cada vez mais comum que hospitais, clínicas médicas e consultórios apostem no uso de recursos tecnológicos para otimizar o seu dia a dia de trabalho.

Em uma consulta é comum fazer a medição dos sinais vitais, que são indicadores do estado de saúde e da garantia das funções circulatórias, respiratória, neural e endócrina do corpo. Podem servir como mecanismos de comunicação universal sobre o estado do paciente e da gravidade da doença. As medidas de temperatura, pulso, pressão arterial, frequências respiratórias e saturação de oxigênio são as mais importantes, comumente avaliados pelos profissionais da saúde [Morsch 2020].

As alterações na frequência cardíaca são normais e indicam habilidade do coração em responder aos múltiplos estímulos fisiológicos, entre eles, estresse mental, alteração hemodinâmica e metabólicas, sono e ortostatismo, bem como compensar desordens procedentes por doenças [Vanderlei 2009].

O oxigênio é vital para o funcionamento de cada célula do corpo humano. De acordo com Webster (1997), na ausência por um tempo prolongado, as células morrerão. Existem várias formas desenvolvidas para analisar o transporte no sangue arterial, o mais rápido e simples é conhecido como oximetria de pulso (SpO2). Para fazer a estimativa da saturação do oxigênio do plasma sanguíneo utilizasse um sensor que emite feixes de dois comprimentos de ondas transmitidos e absorvidos em camadas compactas de tecido, como a do dedo de um indivíduo.

Com a telemedicina já faz parte da rotina de várias instituições de saúde em todo o mundo. A prática, que objetiva a minimização das limitações físicas e geográficas, trouxe inúmeros avanços para a saúde, e revolucionou consideravelmente a forma de se prestar serviços no segmento [MaisLaudo 2020].

Então o presente projeto propõe a sugestão apresentada no trabalho final de graduação do aluno Anderson Ribas da Universidade Franciscana (UFN) no ano de 2020, mudando as tecnologias utilizadas e fazendo a validação. Com objetivo de ajudar e auxiliar o monitoramento dos batimentos cardíacos, saturação do oxigênio e temperatura corporal do paciente em um atendimento á distancia, medindo e enviando os dados coletados para o banco de dados selecionado. Podendo ser integrado com os sistemas de saúde, sejam organizações públicas ou privadas, assim possibilitando analisar uma possível intervenção médica presencial.

1.1. Objetivo Geral

O objetivo é construir um protótipo para monitorar sinais vitais do paciente no atendimento à distância. Serão medidos o batimento cardíaco, temperatura corporal e o nível de oxigênio no sangue. Logo depois o envio via *Wi-Fi* desses valores, assim possibilitando qualquer aplicação com a devida autorização acessar esses dados, para mostrar a equipe de saúde no ato do atendimento.

1.1.1. Objetivos Específicos

Os objetivos específicos deste trabalho foram:

- Implementar os sensores médicos para os dados coletados.
- Implementar o código para controle dos sensores no microcontrolador ESP8266.
- Implementar o Banco de Dados MySQL para receber os dados do ESP8266.
- Implementar no servidor o *script* PHP.
- Implementar o protocolo HTTP no ESP8266.
- Implementar a biblioteca para mudança de SSID e senha do *Wi-Fi* no ESP8266.
- Validação das medições com equipamentos já registrados no mercado.

2. Referencial Bibliográfico

Nesta seção serão apresentados conceitos e informações sobre o atendimento à distância e os elementos tecnológicos presentes neste projeto.

2.1. Telemedicina

Em 2002 foram criadas as primeiras legislações regulamentadoras da prática no Brasil. Com o surgimento do Conselho Brasileiro de Telemedicina e Telessaúde, além disso o Conselho Federal de Medicina (CFM) emitiu a resolução CFM nº 1.643/2002, que define e disciplina a prestação de serviços através da telemedicina [Morsch 2020].

Então ela possibilita que os profissionais de saúde atendam seus pacientes à distância, por meio de suas vertentes como a Teleconsulta, Telediagnóstico, Telecirurgia e Telerradiologia. Assim proporcionando um atendimento ágil e eficiente para os pacientes, o que permite a personalização do tratamento por parte dos médicos [Mayumi 2014].

2.2. Microcontrolador ESP8266

O ESP8266 (Figura 1) é um pequeno chip da fabricante Espressif Systems, desenvolvido especialmente para eletrônicos vestíveis, dispositivos móveis e aplicações em geral. Possui redimensionamento dinâmico de energia e consumo de energia muito baixo.

Este microcontrolador possui conectividade *Wi-Fi* e *Bluetooth* integradas e é capaz de funcionar com temperaturas que variam de -40° C até 125° C. O ESP8266 dispõe de dois núcleos na CPU que podem ser controlados individualmente. Este chip conta com 36 pinos GPIO e é equipado com 4 MB de memória *flash* [Martins 2019].

Figura 1. ESP8266 [Almeida 2018]

2.3. Sensor de Batimento Cardíaco e Oxímetro Max30102

O Max30102 (Figura 2) é uma solução integrada de sensor de oximetria de pulso e monitor de frequência cardíaca. Ele combina dois LEDs, um fotodetector, óptica otimizada e analógico de baixo ruído processamento de sinal para detectar oximetria de pulso e frequência cardíaca.

O módulo utiliza comunicação I2C e opera com fontes de alimentação de 1,8 V e 3,3 V e pode ser desligado por meio de *software* com corrente de espera insignificante, permitindo que a fonte de alimentação permanecer conectado o tempo todo [Datasheet 2009].

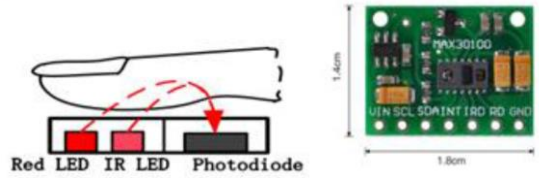

Figura 2. Sensor Max30102 [Wan et al. 2017]

O sensor funciona emitindo luz, um na faixa de comprimento de onda vermelha e outro na faixa de infravermelho, a absorção da luz que penetra na pele é medida, e dependendo da quantidade de oxigênio no sangue a proporção entre a luz vermelha e luz infravermelha serão diferentes. O sensor deve ser posicionado na ponta do dedo onde a pele não é muito grossa para que a luz penetre [Oshima 2020].

2.4. Sensor de Temperatura NTC PT100

De acordo com a fabricante Liohm (2019), o sensor NTC PT100 (Figura 3) foi desenvolvido para medição de temperatura humana e diagnósticos clínicos. Fabricados com elementos ultra precisos e rápida resposta e uma variedade de formas para cada parte do corpo humano e com baixo custo.

O sensor trabalha na faixa de temperatura que fica entre -50ºC até +200ºC. O tempo de resposta é de 5 a 10 segundos, um tempo aceitável para esse projeto. Sua tolerância é de 0.1°C até 1.0°C.

Figura 3. Sensor de Temperatura NTC [Liohm 2019]

2.5. Protocolo de Comunicação HTTP

Em 1999 surgiu a versão HTTP/1.1 para suprir a necessidade de distribuir de forma padronizada informações entres os computadores ligados a internet, HTTP é da sigla em inglês *HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto, em português).

É um protocolo de comunicação de informações de hipermídia, distribuídos e colaborativos, muito utilizado para transferência de páginas HTML. Precisa estar agregado a outros dois protocolos de rede: o TCP (*Transmission Control Protocol*) e o IP (*Internet Protocol*). Assim se forma o modelo TCP/IP, necessário para a conexão entre computadores clientes e servidores [Mello 2021].

2.6. Linguagem de Programação PHP

De acordo com Niederauer (2017) em 1995 o groenlandês Ramus Lerdorf criou para uso pessoal a ferramenta PHP/FI, onde mais tarde se tornou a linguagem de programação mais utilizada pelo desenvolvedor voltado a internet. O PHP da sigla em inglês *Personal Home Page* é uma linguagem interpretada livre, o código é interpretado no lado do servidor e gera a página *web* a ser visualizada no lado do cliente.

Também de acordo com Canto (2011), o PHP é uma das linguagens mais populares em existência, sendo usada em mais de 20 milhões de domínios. Os usuários incluem sistemas como a plataforma de aprendizado Moodle e sites de grande porte, como a rede social Facebook.

2.7. MySQL

Existem vários tipos de armazenamentos de dados na maioria dos sistemas de computador. De acordo com Silberschatz (2012), esses meios de armazenamento são classificados pela velocidade com que os dados podem ser acessados, pelo custo por unidade de dados e pela confiabilidade do meio.

O SGBD (Sistema de gerenciamento de banco de dados) MySQL foi desenvolvido em 1994 por uma empresa sueca chamada MySQL AB e adquirida em 2010 pela empresa norte-americana Oracle Corporation. É de código aberto e sendo modelo relacional [Souza 2019].

Ainda de acordo com Silberschatz (2012), um banco de dados relacional consiste em uma coleção de tabelas, cada qual recebendo um nome exclusivo. Por exemplo, um banco de dados relacional basicamente é constituído por tabelas e cada uma dessas tabelas recebe um nome único.

2.8. Métodos Ágeis

De acordo com Beck (2001), os métodos ágeis ganharam esta nomenclatura após a criação do manifesto para desenvolvimento ágil de *software*. Vários profissionais da área de desenvolvimento uniram 4 valores e 11 princípios em um manifesto, propondo ser uma alternativa, para o que eles afirmavam ser, os processos de desenvolvimento guiados por documentação e pesos-pesados.

Segundo os autores do manifesto ágil, seria necessário valorizar mais os indivíduos e interações, *software* em funcionamento, colaboração com o cliente e responder a mudanças por mais que outros objetivos também têm valor [Beck 2001].

De acordo com Abrahamsson (2003), um processo de desenvolvimento de *software* pode ser caracterizado como ágil se ele é incremental, cooperativo, descomplicado e adaptativo. Incremental significa que o processo faz entregas frequentes de *software*. Cooperativo indica que há interação frequente com os envolvidos no projeto. Os envolvidos no projeto incluem, entre outros, clientes e desenvolvedores. Descomplicado implica em um processo fácil de aprender e de adaptar. Adaptativo indica que há a possibilidade de atender às mudanças, mesmo as de última hora.

2.8.1 Extreme Programming (XP)

O XP foi criado na década de 90 por Kent Beck, Ward Cunningham e Ron Jeffries, e ainda é um método bastante utilizado, um dos quais foca na satisfação do cliente e na entrega incremental, possuindo os objetivos de desenvolver sistemas rápidos e corretos, priorizando principalmente o desenvolvimento do software sobre a análise e o projeto do sistema [Soares 2004].

O XP contém atividades metodológicas que funcionam como os pilares da gestão, são eles: planejamento, projeto, codificação e testes. E para que tudo ocorra como previsto, consistem em quatro valores principais, que são eles:

- Feedback: o programador terá informações constantes do código e do cliente.
- Comunicação: manter o melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação.
- Simplicidade: permitir a criação de código simples que não deve possuir funções desnecessárias.
- Coragem: necessário coragem para implantar os três valores anteriores. Por exemplo, em muitos casos fazer alteração no que vinha funcionando corretamente com o risco de gerar falhas no sistema [Teles 2017].

3. Trabalhos Correlatos

Nessa seção irá apresentar três artigos escritos por outros estudantes e orientadores com o mesmo objetivo ou utilizando uma tecnologia semelhante para seus projetos.

3.1. Monitor Multiparâmetro de Sinais Vitais utilizando Hardware de Baixo Custo

O trabalho de Chisostomo, Ledel e Oliveira (2019) do Instituto Federal de Ciência e Tecnologia de São Paulo (IFSP), teve como objetivo apresentar a pesquisa e desenvolvimento de um monitor multiparâmetro de sinais vitais com transmissão de dados por uma rede sem fio, utilizando *hardware* de baixo custo.

A arquitetura de *hardware* utilizada foi um Arduino Uno, sensor Max30100, display LCD Nokia 5110 e um minicomputador Raspberry Pi Zero. Conforme os autores permite obter valores de sinais vitais próximos ao obtido com um oxímetro comercial, porém com a vantagem de ser uma arquitetura aberta, com tecnologias de baixo custo e código aberto. O SGBD utilizado foi MariaDB, baseado em MySQL. Para o envio dos dados coletados foi utilizado uma API do modelo REST em PHP.

Para os dados do sensor chegar ao Respberry Pi o Arduino Uno envia *scripts* com as medições todo momento. Possibilitando assim o envio utilizando API e um processamento melhor por conta do Respberry Pi.

3.2. Desenvolvimento de um sistema para monitoramento da frequência cardíaca em atividades físicas

O trabalho de Pascoal (2020) na Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ), teve como objetivo levar informações do cuidado com as doenças cardiovasculares, medindo a frequência cardíaca com baixo custo, sendo enviado os dados em tempo real via *Wi-Fi* para um IP local.

Os *hardwares* selecionados para esse desenvolvimento foi o sensor de frequência cardíaca Max30102, microcontrolador ARM e ESP8285. Conforme Pascoal (2020), assim o processamento e transmissão de variáveis e estímulos físicos do corpo humano acontecem com precisão e exatidão que são fundamentais para minimizar as consequências do atual nível de estresse da sociedade.

Na parte da implementação, utilizou o microcontrolador ARM, que por sua vez trata dos dados provenientes do sensor Max30102 que realiza medições de frequência cardíaca. Para a comunicação *Wi-Fi* de dados o ARM envia para o ESP8285, que efetua a transmissão de informações em tempo real a um *web server*, proporcionando um endereço IP específico que possibilita a conexão com qualquer dispositivo interligado à mesma rede. Além disso uma bateria para não ser preciso o uso de tomadas próximas.

Na conclusão apresentada pelo Pascoal, umas das sugestões futuras foram na mudança da comunicação entre os *hardwares* para I2C. Fazer a verificação de testes em vários cenários diferentes. E sua principal mudança é a implementação método de detecção de queda para usuários idosos.

3.3. Dispositivo para Monitoramento Remoto de Sinais Vitais Utilizando ESP32

De acordo com Ribas (2020), que descreve o desenvolvimento de um protótipo para medir sinais vitais de um paciente em atendimento domiciliar, possibilitando ao médico analisar a possível intervenção caso os dados estiverem muito alterados.

Para aquisição dos parâmetros de frequência cardíaca, saturação de oxigênio, temperatura corporal utilizou-se sensores Max30100 e LM35. Os dados enviados ao microcontrolador ESP32 com conexão *Wi-Fi* com protocolo MQTT, para ser exibido na plataforma ThingSpeak. Onde o projeto proporciona monitoramento constante do paciente em assistência domiciliar.

Na conclusão do Ribas, apresentou alguns problemas, primeiramente a troca de sensores para melhor confiabilidade dos dados. Já na parte prática do protótipo ele sugeriu a colocação de baterias para melhor praticidade.

4. Metodologia

Esta seção irá apresentar a proposta e planejamento do *hardware* e *software* em desenvolvimento, ferramentas utilizadas e a metodologia ágil selecionada.

4.1. Proposta

O trabalho consiste na construção de um protótipo para medir a frequência cardíaca, saturação de oxigênio no sangue e temperatura corporal do paciente para a integração em um sistema de saúde, com objetivo de monitorar os sinais à distância.

O protótipo pretende apresentar medições compatíveis com outros equipamentos já validados, dando foco nos sensores de temperatura e o sensor Max30102. Já na parte de armazenamento dos dados, foi utilizado o banco de dados MySQL, que receberá os dados via protocolo HTTP.

Então foi dividido em três grandes partes. A primeira parte é dividida em duas etapas, uma é a construção do *hardware* com seu microcontrolador e sensores, e a segunda etapa é a programação do código. A segunda parte consiste na criação do *script* PHP para gravar no banco. Já na terceira e última etapa, a criação do protocolo HTTP no microcontrolador ESP8266.

No protótipo do *hardware* foi utilizado o sensor de frequência cardíaca Max30102 para medição dos batimentos cardíacos, saturação do oxigênio no sangue e o sensor NTC para temperatura corporal do paciente. Os dados coletados serão enviados

ao microcontrolador ESP8266, que fará a verificação e o agrupamento dos dados para ser enviados.

Para a comunicação do microcontrolador com o banco de dados, foi utilizado o protocolo HTTP, mesmo sendo um *hardware* de baixo processamento, vai desempenhar um papel aceitável pelo motivo de poucos dados serem enviados por vez. Mas para gravar no banco, foi utilizado um *script* PHP, onde o microcontrolador o acessa no servidor, enviando os dados coletados.

O *script* contém as informações fundamentais para registrar os dados, alguns deles são o *host*, senha, nome da tabela e o próprio "*insert"* do SQL. Além disso, o *script* contém uma confirmação no final se os dados foram registrados com sucesso para retornar ao microcontrolador. Abaixo uma demonstração geral (Figura 4) da proposta.

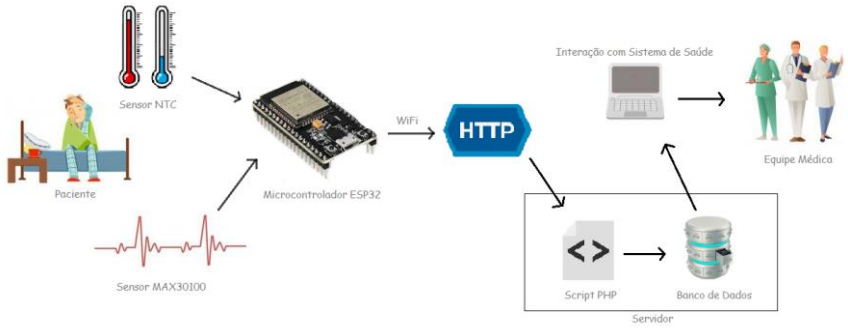

Figura 4. Visão Geral da Proposta

4.2. Planejamento

De acordo com a metodologia ágil XP, citada acima, não define um padrão de construção, mas sim práticas saudáveis para chegar a um produto funcional o mais rápido possível. Como este projeto pode sofrer com muitas mudanças ao longo do tempo, a metodologia escolhida se encaixa perfeitamente, além disso, seu principal objetivo é focar nas principais funcionalidades do sistema.

Então nessa sessão de planejamento apresentam-se diagramas da UML, que seu objetivo é descrever a estrutura, comportamento do sistema e os objetos nele contido. Para construção dos diagramas foi utilizado os programas Astah, BRModelo e Fritzing, que auxiliam nessa parte do projeto.

Para uma visão geral da construção foi desenvolvido o diagrama de implementação (Apêndice A), onde se mostra o resumo de todo projeto com informações de conexão e principais funções descritas na proposta.

Para melhor entendimento em relação ao diagrama de implementação, vem dois diagramas de atividade (Apêndice B e Apêndice C). O primeiro busca descrever o passo a passo do *hardware*, do momento que é ligado até o envio dos dados. Logo após o segundo demonstra os passos do *script* PHP no servidor.

No que se refere a construção do *hardware*, o programa Fritzing auxilia na visualização de todas as ligações entre os sensores e microcontrolador utilizando um diagrama (Apêndice D). Por fim, o diagrama de entidade e relacionamento (Apêndice E) que descreve o banco de dados, com as variáveis e tabelas.

5. Implementação

Nesta seção está descrito os principais passos da implementação e construção do *hardware* e *software* do sistema.

5.1. *Hardware*

Na construção da parte física do projeto, foi divido em duas partes, primeira parte a estrutura (Figura 5) e segunda parte a placa com as ligações dos sensores e microcontrolador (Figura 6 e Figura 7).

Na construção da estrutura foi optado pela utilização de polimetilmetacrilato branco, assim possibilitando uma fácil colocação distribuída dos sensores. Para leitura dos batimentos cardíacos e oxigênio no sangue foi colocado o sensor Max30102 na parte superior para aproximar o dedo, enquanto o sensor de temperatura (NTC PT100) faz a leitura na parte inferior, assim ficando na altura do pulso, como mostra a Figura 5.

Figura 5. Imagem da parte superior do protótipo

Na construção da placa, foram colocados *jumpers* na parte superior, assim possibilitando um fácil encaixe para manipulação dos componentes utilizados, como mostra Figura 6 [A]. Na parte inferior, foi utilizado cabos condutores para fazer as ligações, seguindo o esboçado presente no Apêndice D, com soldas de estanho para fixação na placa, como mostra a Figura 6 [B].

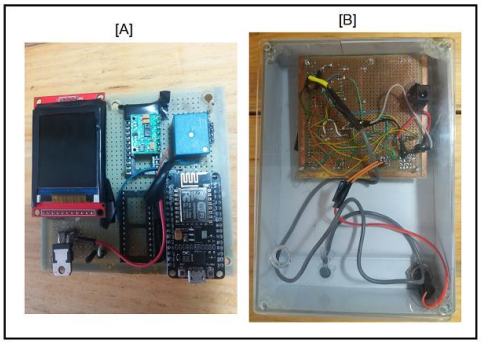

Figura 6. Imagem da parte superior e inferior da placa

5.2. *Software*

Nesta subseção será descrito a codificação com imagens para ilustrar os principais itens desenvolvidos e implementados.

5.2.1. Código do ESP8266

Foi desenvolvido o código para o ESP8266 na linguagem C++ com algumas modificações por padrão que o Arduino IDE utiliza. Os próximos tópicos vão descrever os principais processos do código implementado nesse projeto.

5.2.1.1. Conexão *Wi-Fi*

Quando o microcontrolador é ligado, primeiro passo é a tentativa de conectar com a rede *Wi-Fi* salva em sua memória, nesse caso a última conectada. Para essa primeira tentativa é chamado a função *"conectarWiFi"* (Figura 7) que fica localizado no *"setup()"* da aplicação.

```
167 void conectarWiFi(){
168     wifiManager.autoConnect();//Função para se autoconectar na rede
169     if(WiFi.status() == WL_CONNECTED){ //Se conectado na rede
170      Serial.println("Conectado a Rede!");
171     }
172     else{ //se não conectado na rede
173      configWifi(); //Chama função para configurar WiFi
174     }
175 }
```

Figura 7. Função conectar *Wi-Fi*

Na função acima, a biblioteca "*Wi-FiMananger.h"* fica responsável por se conectar à rede que está na memória, logo depois uma verificação caso tenha sucesso ou fracasso. Como neste projeto o dispositivo é de fácil transporte, então é necessário um método para trocar a rede, sem ter que reescrever o código fonte do microcontrolador para novas informações toda vez. Então é chamado a função *"configWiFi()"*(Figura 8) caso não tenha sucesso na conexão da rede no primeiro momento.

```
208 void configWifi(){
209   Serial.println("Configurar WiFi pelo Smartphone"); //Abre o portal
210   wifiManager.resetSettings(); //Apaga Configurações de Rede Salvas
211   if(!wifiManager.startConfigPortal("ESP-CONFIG", "12345678") ){ //Rede e Senha gerada pela ESP
212     Serial.println("Erro ao conectar"); //Se caso não conectar na rede mostra mensagem de falha
213     delay(2000);
214     ESP.restart(); //Reinicia ESP após não conseguir conexão na rede
215   }else{       //Se caso conectar
216     Serial.println("Conectado na Rede!!!");
217     ESP.restart(); //Reinicia ESP após conseguir conexão na rede
218   }
219 }
```

Figura 8. Função para configurar Wi-Fi

Nessa função acima tem como objetivo criar um ponto de acesso para o ESP8266 receber os dados que necessita para realizar a nova tentativa de conexão com a rede. Neste momento o microcontrolador entra no Modo AP (*Access Point Mode*), que o dispositivo se comporta como um roteador *Wi-Fi* gerando uma rede com SSID e senha (Linha 211 da Figura 8) predefinido no código.

Fazendo a utilização de um *smartphone* é possível se conectar ao ESP8266 com as informações descritas anteriormente. Para configurar, basta abrir o navegador e pesquisar pelo endereço IP (*Internet Protocol*) do microcontrolador, disponível nas informações de *"Gerenciar roteador"* do dispositivo móvel. Na figura 9 [A] demonstra a tela inicial, onde se encontra a opção *"Configure WiFi",* assim levando para a próximo passo que demonstra a figura 9 [B].

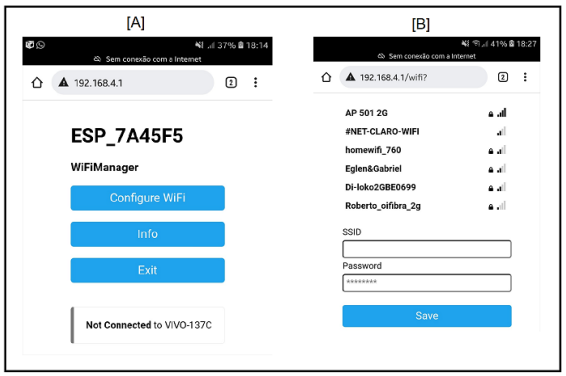

Figura 9. Tela de configuração no *smartphone*

Logo após salvar os novos dados de rede utilizando o *smartphone*, basta reiniciar o microcontrolador e os processos de conectar na rede (Figura 7) irá ser repetido. Caso tenha sucesso com essas novas informações, irá entrar em Modo Estação (*Station Mode*), onde o dispositivo se comporta como um cliente sem fio e se conectando a um roteador *wireless*. Sendo assim não vai entrar no Modo AP, que por sua vez a rede ESP8266 não ficará visível para os *smartphone*.

5.2.1.2. Obtendo medições

Utilizando o sensor NTC PT100 para obtenção da temperatura, foi codificado uma função (Figura 10) para calcular a média dos valores obtidos. O primeiro passo é somar as amostras medidas *"soma += analogRead(pinTermistor)"*, na variável *"pinTrermistor"* é referente ao pino *"A0"* que o sensor está conectado na placa.

O cálculo também se faz uso de algumas variáveis pré-definidas, como a *"vcc"*, que recebe o valor da voltagem que o sensor está recebendo, nesse caso 3.3 *volts*. Na variável *"R"* é armazenado o valor da resistência 10k. No *"beta"*, que significa ganho estático de corrente de um transistor na configuração de emissor comum é 3950.0. Para definição dos valores o próprio fabricante do sensor tem o manual de recomendações para utilização.

```
221 float getTemperatura(){
222   int soma = 0;
223   for (int i = 0; i < nAmostrasColetadas; i++) {
224     soma += analogRead(pinTermistor);
225     delay (10);
226   }
227
228   // Determina a resistência do termistor
229   double v = (vcc*soma)/(nAmostrasColetadas*1024.0);
230   double rt = (vcc*R)/v - R;
231
232   // Calcula a temperatura
233   double t = beta / log(rt/rx);
234
235   return (t-273.0);
236 }
```

Figura 10. Função para obtenção da Temperatura

No sensor MAX30102 foi utilizado a biblioteca *"MAX30100_PulseOximeter.h"*, assim possibilitando o uso do *"PulseOximeter pox"*, que por sua vez tem todos os métodos. Para obtenção dos batimentos cardíacos ficou *"pox.getHeartRate()"* (Figura 11), e para obtenção do SpO2 *"pox.getSpO2()"*(Figura 11), ambos retornando sem precisar de cálculos adicionais.

```
238 float getBatimentos(){
239   return pox.getHeartRate();
240 }
241 float getSpO2(){
242   return pox.getSpO2();
243 }
```

Figura 11. Obtenção das medições do Max30102

5.2.1.3. Envio das medições

Para o envio das medições foi criado uma função chamada *"enviarDadosBD"* (Figura 12) que recebe por parâmetros os dados de batimentos cardíacos, oximetria, temperatura e ID do *hardware* que está vinculado os sensores.

```
96  void enviarDadosBD (float batimentos, int spO2, float temperatura, int hardware){
97    if(client.connect(server, 80)){
98      client.printf("GET /tfg/bdtfg.php?b=%.2f&s=%d&t=%.2f&hardware=%d&chave=chave12 HTTP/1.1\r\n",batimentos, spO2, temperatura, hardware);
99      client.println("Host: 192.168.0.15:80");
100     client.println("Connection: close");
101     client.println();
102   }
103   delay(2000); //Tempo para receber a resposta do servidor
104   while (client.available()){
105     char c = client.read();
106     resposta += c;
107   }
108   Serial.println("-----------------------------------");
109   Serial.println("Resposta do Servidor: " + resposta);
110   Serial.println("-----------------------------------");
111   if(!client.connected()){
112     if(resposta.indexOf("Comando SQL - Sucesso") >= 0){
113       Serial.println("SUCESSO - Comando SQL");
114     }else{
115       Serial.println("ERRO - Comando SQL");
116     }
117     client.stop();
118   }
119   delay(10000);
120 }
```

Figura 12. Função para envio dos dados

Definindo uma conexão com o servidor (Linha 97 da Figura 12) utilizando o IP e porta predefinida, já é possível o envio dos dados com comando HTTP (Linha 98 da Figura 12). O comando tem informações das medições e chave de segurança para ser verificado no *script* mais tarde. Depois de enviado a requisição para gravar no servidor, é esperado uma resposta de confirmação dos dados gravados com sucesso.

5.2.2. Banco de dados

Como o projeto foi pensado em terminais de telemedicina, pode haver diversos *hardwares* trabalhando juntos, então foi criado algumas tabelas para relacionar e separar essas medições aos equipamentos.

Como mostra o diagrama lógico (Apêndice F) desenvolvido no BRModelo, existem três tabelas. A principal chamada *"terminal"*, que é utilizado para informações gerais do terminal de atendimento. Já a segunda tabela chamada *"hardware"*, serve para armazenar informações de modelo de sensores e uma chave estrangeira apontada para tabela *"terminal"*, assim possibilitando identificar em qual terminal está localizado. Já a terceira tabela *"dados_medicoes"*, registram-se os dados e tem a chave estrangeira da tabela "hardware", sendo assim, identificar em qual *hardware* está vinculado os dados.

Como mostra a figura acima, tabela *"dados_medicoes"* é a única que recebe diretamente do microcontrolador. Mas entre esses dados, tem a coluna *"id_dados"*, que por sua vez incrementa conforme a quantidade de linhas existentes na tabela. Já a coluna *"data_hora"*, é registrado automaticamente a data e hora do servidor no momento do registro dos dados.

5.2.3. *Script* PHP

O *script* PHP fica localizado no próprio servidor que está o banco de dados, ele vai ficar responsável por receber os dados do microcontrolador e registrar. Quando os dados são recebidos, são armazenados nas variáveis locais (Figura 13) e verificado a chave de segurança que somente o microcontrolador possui.

```php
$temperatura = $_GET['t'];
$batimentos = $_GET['b'];
$spO2 = $_GET['s'];
$chave = $_GET['chave'];
$hardware = $_GET['hardware'];

if($chave != 'chave12'){
    echo 'chave invalida';
    die();
}
```

Figura 13. Variáveis locais

O próximo passo do script é construir o comando SQL (*Structured Query Language*) (Figura 14), utilizando dados previamente determinados para utilização do banco, como *host* e *password*. E por fim o *script* executa o comando SQL, com *"$stmt->execute()"*, onde também retorna a confirmação para o microcontrolador se os dados foram registrados com sucesso.

```php
$dbh;
$stmt;
$connStr = 'mysql:host=localhost;dbname=dados';
try{
    $dbh = new PDO($connStr, 'root', '');
}catch(PDOException $e){
    echo $e->getMessage();
    die();
}
$stmt = $dbh->prepare(
    'INSERT INTO dados.medicoes (batimentos, spO2, temp, id_hardware)
        VALUES (:batimentos, :spO2, :temperatura, :hardware)'
);

$stmt->bindValue(':batimentos', $batimentos);
$stmt->bindValue(':spO2', $spO2);
$stmt->bindValue(':temperatura', $temperatura);
$stmt->bindValue(':hardware', $hardware);

if($stmt->execute()){
    echo 'Comando SQL - Sucesso';
}else{
    echo 'Comando SQL - Erro';
}
```

Figura 14. Comando SQL

5.3. Testes e validação

Nos testes para validação do projeto, foram feitos através da comparação entre o dispositivo desenvolvido e equipamentos já aprovados no Instituto Nacional de Metrologia, Qualidade e Tecnologia (Inmetro). Equipamentos utilizados foram o G-TECH LED e termômetro digital CARETECH TS-101, ambos com selo de aprovação.

Para realização da comparação, foi medido cinco vezes, cada um em diferentes horas do dia. Primeiro em um homem de 23 anos (Tabela 1), segundo uma mulher de 27 anos (Tabela 2), ambos sem problemas cardíacos ou histórico de doenças relacionadas. As tabelas apresentam a descrição de: "Bpm" (Batimentos cardíacos); "SpO2" (Saturação de oxigênio); "T" (Temperatura).

Tabela 1. Comparações dos Dados

Verificação	Dados do Dispositivo			Outros Equipamentos			Diferença de medições		
Homem, 23 Anos	Bpm	SpO2	T	Bpm	SpO2	T	Bpm	SpO2	T
1°	80	96%	35.6	71	97%	35.7	9	1%	0.1
2°	67	96%	36.2	75	97%	36	8	1%	0.2
3°	140	95%	35.8	66	98%	35.5	74	3%	0.3
4°	82	95%	35.5	88	98%	36.4	6	3%	0.9
5°	75	97%	35.9	70	97%	36.1	5	0%	0.2

Tabela 2. Comparações dos Dados

Verificação	Dados do Dispositivo			Outros Equipamentos			Diferença de medições		
Mulher, 27 Anos	Bpm	SpO2	T	Bpm	SpO2	T	Bpm	SpO2	T
1°	75	97%	35.8	70	99%	36	5	2%	0.2
2°	36	96%	35.5	65	98%	35.4	29	2%	0.1
3°	72	96%	35.5	80	99%	35.5	8	3%	0
4°	90	96%	36.2	72	98%	35.9	18	2%	0.3
5°	85	95%	36	90	99%	36.6	5	4%	0.6

Além dos dados acima apresentados, foram realizadas mais de 150 verificações, chegando a números muito próximos, mas por outro lado, chegando a números não compatíveis, sendo assim uma instabilidade muito grande. Principalmente o sensor Max30102 que apresenta os batimentos cardíacos e oxímetro de pulso, não teve bom desempenho, mesmo buscando mudanças nas bibliotecas e codificações.

6. Conclusão

O presente trabalho apresentou a construção de um protótipo para medir e enviar os sinais vitais de um paciente no atendimento a distância. Foi utilizado as boas práticas da metodologia XP, que por sua vez auxiliou no planejamento com diagramas da UML e o desenvolvimento da codificação.

Então objetivo principal foi cumprido, criação do protótipo com os diversos testes, que por sua vez ao longo da comparação foi feito mudanças para buscar números mais aceitáveis. Mesmo com diversos contratempos e problemas referentes ao *hardware*, foi possível chegar no final e apresentar uma análise crítica baseada em dados.

Com as diversas comparação das validações é possível definir as mudanças para trabalhos futuros, sejam elas a substituição do equipamento ou modificações buscando uma margem de erro mais aceitável. Ressaltando também a dificuldade da utilização de equipamentos com baixo custo para área da saúde.

Portanto, o avanço da tecnologia permite cada vez mais a facilidade e utilidades de *hardwares* e *softwares* aplicados na área médica, com base nisso, este trabalho chega para agregar a vasta expansão do cuidado e auxílio com a saúde.

6.1. Problemas e Dificuldades

Os sensores com baixo custo apresentam muitos problemas, foi o caso do MAX30102 com baixa estabilidade, como apresentado nas tabelas das medições (Tabela 1 e 2),

mostrando inconsistência nos dados dos testes em relação a equipamento já validado. No que se refere ao sensor de temperatura, mostraram-se medições muito próximas, sendo promissor a utilização dele.

Outro problema apresentado ao longo do desenvolvimento é a dificuldade da obtenção dos dados por parte do MAX30102, muitos erros e dados absurdos no começo foi detectado, sendo assim, chegando à conclusão que não é viável atualmente a utilização dele para um sistema de saúde.

6.2. Trabalhos Futuros

A principal sugestão proposta será a uma nova validação dos dados, fazendo mais testes com alterações de código do sensor MAX30102 ou talvez a mudança dele. Logo depois, comparando novamente com equipamentos já registrados e aprovados no Inmetro. Outras sugestões são:

- Colocar sensor de pressão arterial, que se refere à pressão exercida pelo sangue contra a parede das artérias, assim disponibilizando mais um dado importante nos atendimentos.

- Fazer a substituição da placa atual por uma de circuito impresso, facilitando a manipulação sem a utilização de soldas e cabos.

- Trocar o *display*, colocando um maior com a função *touch screen,* para assim servir como botão de iniciar as leituras, substituindo o botão físico atual.

- Desenvolver uma API para interação com sistemas de saúde.

7. Referências

Morsch, J. A. (2020). “Guia completo sobre a tecnologia na saúde: como aplicar na medicina?”, “Morsch”. Disponível em: https://telemedicinamorsch.com.br/blog/tecnologia-na-saude

Vanderlei, L. C. M., Pastre, C. M., Hoshi, R. A., Carvalho, T. D. D., & Godoy, M. F. D. (2009). Noções básicas de variabilidade da frequência cardíaca e sua aplicabilidade clínica. Brazilian Journal of Cardiovascular Surgery, 24(2), 205-217.

Webster, J. G. (Ed.). (1997). Design of pulse oximeters. CRC Press.

MaisLaudo. (2020). “A evolução da telemedicina no mundo”, “MaisLaudo”. Disponível em: https://maislaudo.com.br/blog/telemedicina-no-mundo/

Mayumi, Y. (2014). “8 avanços da tecnologia aplicada à saúde”, “iClinic”. Disponível em: https://blog.iclinic.com.br/o-poder-da-tecnologia-aplicada-a-saude/

Martins, V. F. (2019). Automação residencial usando protocolo MQTT, Node-RED e Mosquitto Broker com ESP32 e ESP8266.

Almeida, G. (2018). “Configurando o ambiente de desenvolvimento do ESP32 no Windows”, “Embarcados”. Disponível em: https://www.embarcados.com.br/ambiente-esp32-no-windows/

Datasheet, M. (2009). Pulse oximeter and heart-rate sensor IC for wearable health.

Wan, J. et al. (2017) “Sistema de detecção de saturação de oxigênio no sangue tipo reflexivo baseado no MAX30100”. In: 2017 Conferência Internacional sobre Segurança, Análise de Padrões e Cibernética (SPAC). IEEE, 2017. p. 615-619.

Oshima, C. K. (2020). Desenvolvimento e validação de sistema de aferição de esforço físico em equipamento de treinamento de surfe.

Liohm. (2019). “Sensores para uso médico e laboratorial”, “Liohm”. Disponível em: https://liohm.com.br/produto/sensores-ntc-medica-e-laboratorial/

Thomsen, A. (2014). “O que é Arduino?”, “FilipeFlop”. Disponível em: https://www.filipeflop.com/blog/o-que-e-arduino/

Mello, A. “O significado e a origem do protocolo HTTP”, “Católica EAD”, Disponível em: https://ead.catolica.edu.br/blog/significado-origem-protocolo-http. Acessado em maio de 2021.

Alves, L. P., Medeiros, O., dos Santos, V., Mello, L. C. R., & Nunes, A. B. Oficina de Python Aplicado a Meteorologia. z, 78.

Niederauer, J. (2017). PHP para quem conhece PHP. Novatec Editora.

Canto, F. H. (2011). Vulnerabilidades da linguagem PHP.

Silberschatz, A; Korth, H.F; Sudarshan, S. Sistemas de Banco de Dados. 6.ed. Rio de Janeiro: Campus, 2012.

Souza, E. C., & de Oliveira, M. R. (2019). Comparativo entre os bancos de dados mysql e mongodb: quando o MongoDB é indicado para o desenvolvimento de uma aplicação. Revista Interface Tecnológica, 16(2), 38-48.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for agile software development.

P. Abrahamsson, J. Warsta, M. T. Siponen and J. Ronkainen (2003). "New directions on agile methods: a comparative analysis," 25th International Conference on Software Engineering.

dos Santos Soares, M. (2004). Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. Revista Eletrônica de Sistemas de Informação, 3(1).

Teles, V. M. (2017). Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. Novatec Editora.

de Moura Chrisostomo, B. M., Ledel, L., & de Oliveira, R. F. Monitor Multiparâmetro de Sinais Vitais utilizando Hardware de Baixo Custo.

Pascoal, P. G. (2020). Desenvolvimento de um Sistema para Monitoramento da Frequência Cardíaca em Atividades Físicas.

Ribas, A. R. (2020). Dispositivo para Monitoramento Remoto de Sinais Vitais Utilizando ESP32. Trabalho Final de Graduação - Universidade Franciscana, Santa Maria - RS.

8. Anexo

Nesta seção é apresentado os apêndices conforme comentados ao longo do artigo acima.

Apêndice A – Diagrama de Implementação do Projeto

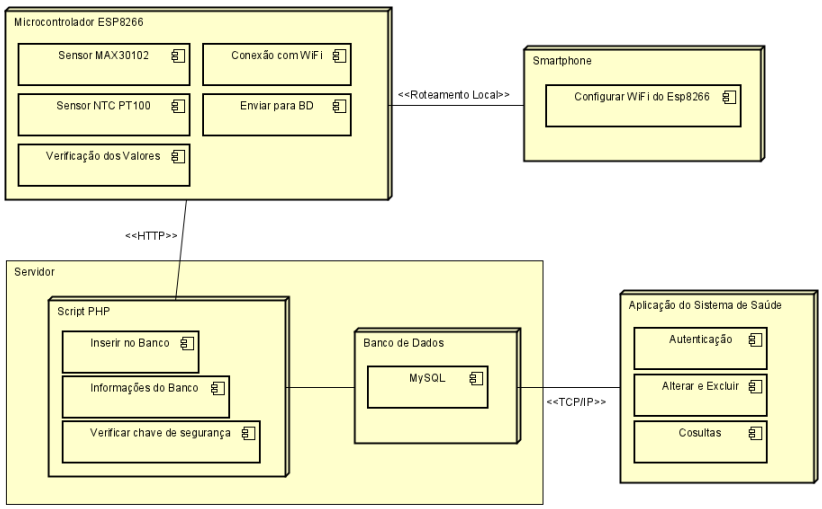

Figura 15. Diagrama de Implementação

Apêndice B – Diagrama de Atividade do ESP8266

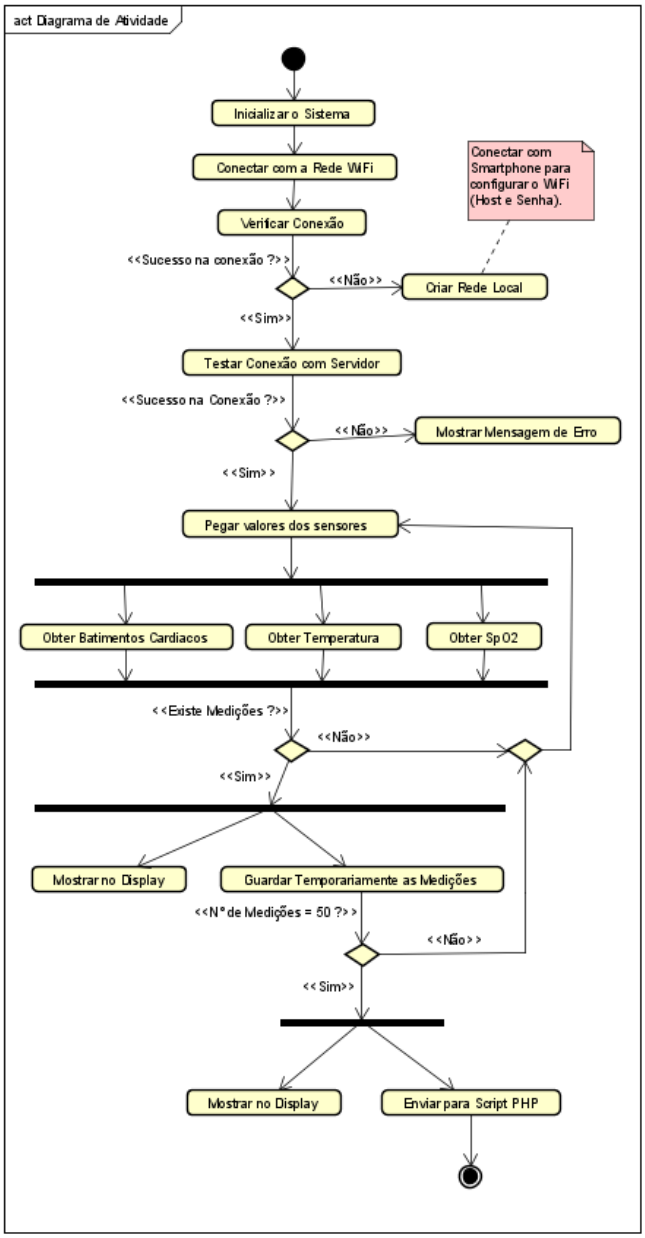

Figura 16. Diagrama de Atividade do *Hardware*

Apêndice C – Diagrama de Atividade do *Script* PHP

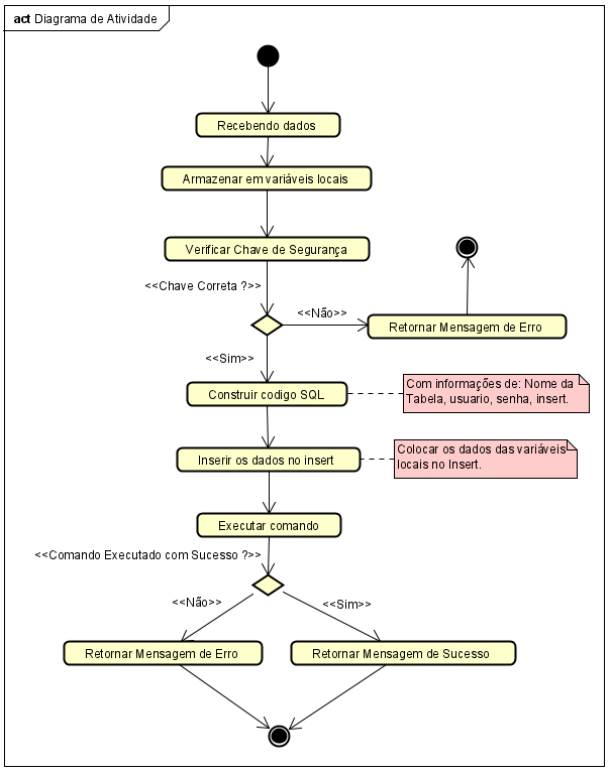

Figura 17. Diagrama de Atividade do *Script*

Apêndice D – Diagrama do Hardware

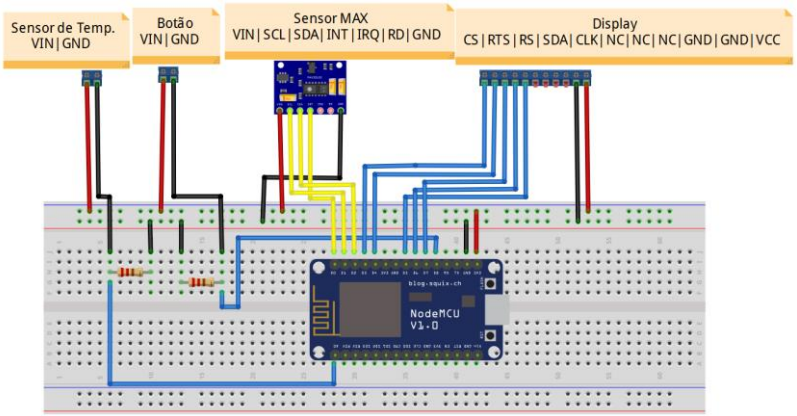

Figura 18. Diagrama do Hardware

Apêndice E – Diagrama de Entidade e Relacionamento

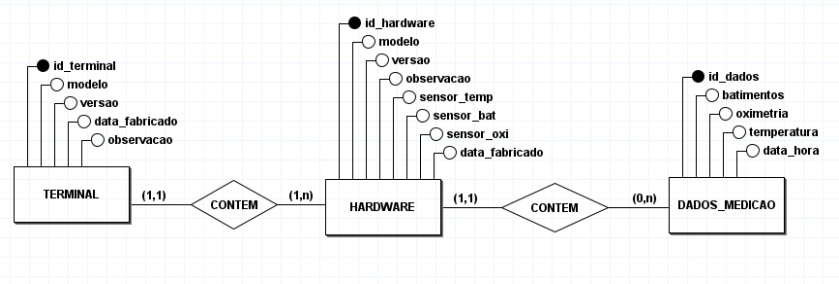

Figura 19. Diagrama de Entidade e Relacionamento

Apêndice F – Diagrama lógico do banco de dados

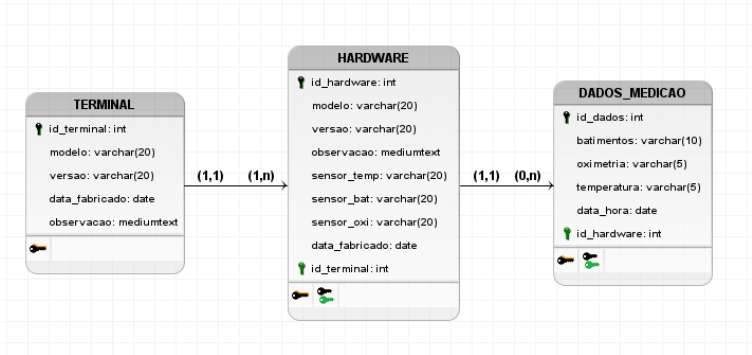

Figura 20. Diagrama lógico do banco de dados