DESENVOLVIMENTO DE BASE DE DADOS E DE APLICATIVO PARA MONITORAMENTO REMOTO DE PACIENTES

Luís Silva dos Reis Neto[1], Idalmir de Souza Queiroz Júnior[2]

Resumo: A telemedicina, impulsionada pela pandemia de Covid -19, emerge como uma alternativa para aumentar a eficácia do atendimento médico e aliviar as demandas do setor de saúde. O monitoramento remoto de pacientes, é uma modalidade deste tipo de atendimento que demonstra um grande potencial, utilizando sensores biométricos para realizar um acompanhamento médico contínuo. No entanto, desafios como a exclusão digital e a falta de conhecimento médico entre a população vêm dificultando sua adoção. Tendo esse cenário em mente, evidencia-se a necessidade por plataformas intuitivas que ofereçam acesso fácil e simples interpretação dos dados clínicos aos pacientes, reduzindo idas desnecessárias aos hospitais e permitindo a adoção de tratamentos personalizados baseados no histórico do paciente. Este trabalho desenvolveu uma aplicação *web*, um banco de dados e utilizou sensores biométricos para acompanhamento médico em tempo real. Para isso, o projeto foi desenvolvido em quatro partes, com a montagem do circuito para captação dos dados biométricos, a criação da base de dados, o estabelecimento da comunicação entre esses elementos e a criação de um aplicativo para demonstrar essas informações. O desenvolvimento do aplicativo e a comunicação com a base de dados foram realizadas com sucesso. No entanto, persistiram problemas com o processo de leitura de um dos sensores utilizados. Ainda assim, como os outros aspectos não foram prejudicados por conta disso, a implementação deste sistema de monitoramento pode servir como base para desenvolvimentos futuros, contribuindo assim para maiores avanços nos estudos da telemedicina e no entendimento do que implicaria um aumento em sua adoção.
Palavras-chave: Monitoramento Remoto de Pacientes; Telemedicina; Sensores Biométricos; IoT.

1. INTRODUÇÃO

Com a perene e grande demanda por atendimentos médicos e com o avanço da tecnologia, a medicina vem buscando se renovar com a utilização de métodos mais modernos para o atendimento às necessidades dos pacientes. Alavancada pelas medidas de isolamento social globalmente aplicadas durante o período da pandemia da Covid-19 [1], a telemedicina e a suas modalidades vem ganhando notoriedade como uma alternativa aos meios tradicionais de atendimento médico.

Dentre essas modalidades, o monitoramento remoto de pacientes (RPM, do termo em inglês *Remote Patient Monitoring*) se destaca pela utilização de sensores biométricos para atender indivíduos acometidos por enfermidades crônicas como apneia, diabetes, pressão alta, além de outras condições cardíacas [2]. O RPM traz consigo um grande potencial para agilizar e aprimorar os tratamentos médicos, assim como pode permitir um aumento da produtividade dos profissionais da área da saúde [3].

Entre os dados biométricos mais comumente aferidos, encontram-se parâmetros como a frequência cardíaca, o nível de oxigênio no sangue (oximetria) e a temperatura corporal [4]. A medição e o armazenamento destas informações podem cumprir um papel fundamental na aprimoração dos tratamentos, permitindo ao profissional de saúde ter um maior nível de conhecimento das condições clínicas de seu paciente.

Também são outras vantagens do RPM, a possibilidade de atender mais facilmente populações localizadas em áreas de difícil acesso e em cidades com pouca infraestrutura hospitalar [3]. Isso se deve a característica remota deste tipo de acompanhamento, possibilitando também em uma redução nas filas para atendimento médico presencial.

No entanto, para desfrutar completamente dos benefícios do monitoramento remoto de pacientes, há de se superar obstáculos como a barreira social da exclusão digital e da falta de conhecimento médico dos pacientes. Vale ressaltar que, a demografia mais afetada por esses problemas sociais é a mesma que, em geral, é acometida por doenças crônicas [5]. Estes desafios representam dificuldades na correta interpretação dos dados por parte dos indivíduos enfermos, o que pode resultar em idas desnecessárias aos ambientes hospitalares, negando completamente um dos benefícios deste tipo de monitoramento.

Dessa forma, se mostra imperativa a necessidade pela criação de ferramentas que facilitem o acesso e a interpretação desses dados biométricos por parte dos pacientes. Isso possibilitaria um maior aproveitamento do potencial do RPM, trazendo vários benefícios à saúde pública e à saúde do próprio paciente.

Assim, a proposta deste trabalho é o desenvolvimento de uma base de dados para o monitoramento remoto de pacientes, juntamente com um aplicativo *web* que busque simplificar o acesso e a visualização dessas informações ao mesmo tempo que garanta a privacidade dos dados biométricos coletados. A base de dados será configurada

utilizando a plataforma *Firebase*, que também será utilizada no desenvolvimento do aplicativo. A obtenção dos dados biométricos ocorrerá por meio da integração de sensores biométricos com um microcontrolador.

2. REFERENCIAL TEÓRICO

O referencial teórico para o desenvolvimento de um aplicativo e base de dados para o monitoramento remoto de pacientes abrange diversas áreas do conhecimento, desde tecnologias de informação e comunicação até conceitos de saúde e cuidados médicos. Este referencial busca embasar o desenvolvimento e a implementação de uma solução tecnológica eficaz e segura para o monitoramento à distância de pacientes.

2.1 Telemedicina e monitoramento remoto de pacientes

A telemedicina representa um avanço significativo na prestação de cuidados de saúde, permitindo a comunicação à distância entre profissionais de saúde e pacientes. No contexto do monitoramento remoto de pacientes, a telemedicina desempenha um papel crucial ao possibilitar o acompanhamento contínuo da saúde dos pacientes fora do ambiente clínico tradicional. A telemedicina envolve o uso de tecnologias de informação e comunicação para permitir consultas médicas à distância, incluindo o acompanhamento contínuo e remoto dos pacientes [2]. Já o monitoramento remoto de pacientes é uma aplicação específica da telemedicina que envolve o acompanhamento contínuo dos sinais vitais e condições de saúde de um paciente sem a necessidade de visitas presenciais ao consultório médico. Isso é possível por meio de dispositivos médicos conectados, sensores e aplicativos que coletam e transmitem informações sobre a saúde do paciente em tempo real [5].

O uso do monitoramento remoto de pacientes enfrenta desafios significativos relacionados à exclusão digital e à falta de conhecimento médico dos pacientes. A exclusão digital representa uma barreira para pacientes que não têm acesso ou familiaridade com tecnologias digitais necessárias para participar ativamente do monitoramento remoto, sendo mais prevalente em populações mais velhas ou em áreas com limitações de acesso à internet e dispositivos tecnológicos. Além disso, a falta de conhecimento médico dos pacientes pode dificultar a interpretação dos dados de saúde coletados por dispositivos remotos, resultando em subutilização ou má interpretação dos resultados, o que pode comprometer a eficácia do monitoramento e a qualidade dos cuidados prestados [3].

Em resumo, a telemedicina e o monitoramento remoto de pacientes representam uma abordagem inovadora para melhorar a qualidade e a eficiência dos cuidados de saúde. Essas tecnologias têm o potencial de revolucionar a forma como os pacientes são monitorados e tratados, proporcionando uma experiência mais conveniente e personalizada, ao mesmo tempo em que enfrentam desafios significativos

2.2 Tecnologias para desenvolvimentos de aplicativos

No desenvolvimento de um aplicativo web, várias tecnologias desempenham papéis fundamentais, desde a construção da interface do usuário até o armazenamento seguro e eficiente dos dados coletados. Uma tecnologia central é o uso de linguagens de programação *web*, como HTML, CSS e JavaScript, para criar uma interface amigável e responsiva que os pacientes possam acessar facilmente em seus dispositivos, como smartphones, tablets e computadores [6].

Além disso, *frameworks* de desenvolvimento *web*, como React.js, Angular.js ou Vue.js, oferecem ferramentas poderosas para a criação de interfaces de usuário dinâmicas e interativas, facilitando a comunicação entre o aplicativo e o usuário. Eles também permitem a construção de componentes reutilizáveis, o que pode agilizar o desenvolvimento e manutenção do aplicativo.

Para o armazenamento seguro e confiável dos dados, é essencial utilizar um sistema de gerenciamento de banco de dados robusto. No caso de um aplicativo *web*, o uso de bancos de dados SQL ou NoSQL pode ser considerado, dependendo dos requisitos específicos do projeto [7]. Bancos de dados SQL oferecem consistência e integridade de dados, enquanto bancos de dados NoSQL são mais flexíveis e escaláveis para lidar com grandes volumes de dados e necessidades de escalabilidade.

A segurança dos dados é outro aspecto crucial. A implementação de práticas como criptografia de dados, autenticação de usuário e controle de acesso é essencial para proteger as informações contra acessos não autorizados e violações de privacidade [8]. O uso de protocolos de comunicação seguros, como HTTPS, e a conformidade com regulamentações de proteção de dados são aspectos críticos durante todo o desenvolvimento.

Em suma, o desenvolvimento de um aplicativo *web* para o monitoramento remoto de pacientes envolve a utilização de uma variedade de tecnologias, desde linguagens de programação até sistemas de gerenciamento de banco de dados e práticas de segurança. Ao escolher e implementar essas tecnologias de forma eficaz, é possível criar uma solução robusta e confiável que atenda às necessidades dos pacientes e profissionais de saúde, proporcionando uma experiência de monitoramento remoto segura e eficiente.

3. MATERIAIS E MÉTODOS

Para atingir os objetivos supracitados, foi adotada uma abordagem estratégica, levando em consideração

critérios como confiabilidade, praticidade e disponibilidade. Nesta seção, serão abordadas as escolhas feitas em relação às escolhas dos materiais, plataformas e as metodologias de desenvolvimento aplicadas para garantir a funcionalidade da integração de todos os elementos utilizados neste projeto.

Os materiais selecionados foram avaliados quanto à sua adequação para coleta de dados biométricos e conectividade, considerando os requisitos especificados nos objetivos da pesquisa. Além disso, os métodos de implementação foram escolhidos com base em práticas recomendadas e tecnologias atuais, visando garantir uma integração harmoniosa entre os diferentes componentes do sistema.

A Figura 1 mostra um diagrama simples dos processos de comunicação que serão realizados para a obtenção e transmissão dos dados, a configuração da base de dados e construção do aplicativo.

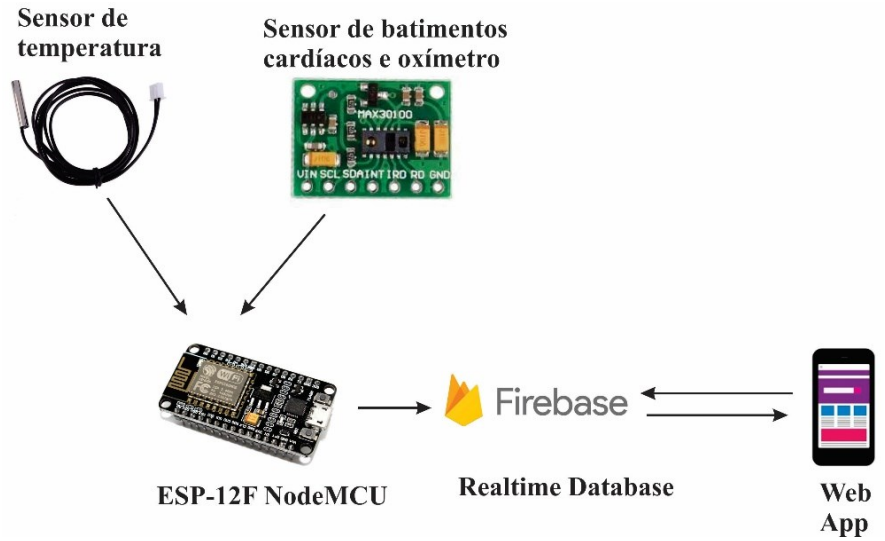

Figura 1. Diagrama dos processos de comunicação (Autoria própria)

3.1 Funcionamento

Neste tópico será apresentada uma breve explicação dos motivos que levaram à adoção das ferramentas utilizadas neste projeto, acompanhada do funcionamento destas ferramentas utilizadas para a captação dos dados biométricos, a construção da base de dados e do aplicativo.

3.1.1 Captação de dados biométricos

Para capturar as leituras dos sensores biométricos e transmiti-las para a base de dados, foi adotado o uso do microcontrolador ESP-12F NodeMCU graças a sua facilidade de programação pois, utiliza a linguagem *Arduino*, baseada em C++, amplamente conhecida e que possui uma vasta gama de desenvolvedores. Essas características facilitam o desenvolvimento de *software* e a implementação de funcionalidades adicionais, além de oferecer acesso a uma ampla gama de bibliotecas e recursos. Além disso, entre outros fatores considerados, é notável que este modelo possui conectividade *Wi-Fi* integrada, simplificando o desenvolvimento de projetos *IoT* com este microcontrolador.

Foram definidos que os dados biométricos a serem coletados serão a temperatura corporal, a frequência cardíaca e o nível de saturação de oxigênio no sangue, já que estão entre as variáveis mais comumente acompanhadas no RPM. Tendo isso em mente, os sensores escolhidos foram o sensor de batimento cardíaco e oxímetro MAX30100 e o sensor de temperatura NTC 10K MF58. Foram escolhidos por seu baixo custo, precisão e compatibilidade com o ESP-12F NodeMCU. Também foi levado em consideração a disponibilidade desses sensores e que este microcontrolador possui apenas uma porta para leituras analógicas, com o sensor de temperatura utilizando esta porta e o sensor MAX30100 utilizando apenas portas digitais.

ESP-12F NodeMCU: Trata-se de um microcontrolador baseado no chip ESP8266, projetado para aplicações de IoT (Internet das Coisas). Ele possui uma interface WiFi integrada, o que o torna ideal para conectar dispositivos à internet e realizar comunicação sem fio. Este microcontrolador pode ser usado para controlar dispositivos, coletar dados de sensores e se comunicar com serviços na nuvem [9], [10].

MAX30100: É um sensor de oximetria de pulso e frequência cardíaca que utiliza a tecnologia de fotopletismografia para medir a absorção de luz infravermelha e vermelha através da pele. Ele emite luzes de LED de comprimentos de onda específicos na pele e mede a quantidade de luz refletida para determinar a quantidade de oxigênio no sangue e a taxa de batimentos cardíacos [10], [11].

NTC 10K MF58: Este sensor é um tipo de termistor que possui uma resistência que diminui conforme a temperatura aumenta, sendo comumente utilizado em circuitos de monitoramento de temperatura. Para medir a temperatura com esse termistor, é comum utilizá-lo em conjunto com um resistor fixo em um divisor de tensão.

Esse circuito gera uma tensão proporcional à temperatura, que pode ser lida por um microcontrolador para calcular a temperatura ambiente ou de um objeto [12].

3.1.2 Base de dados

Para configurar a base de dados, utilizou-se a ferramenta *Realtime Database* da plataforma *Firebase* que pertence ao Google e permite a criação de web apps, dispondo de uma estrutura de *backend* mais facilmente customizável. A escolha por esta opção teve como principal motivador a possibilidade de utilizar outras ferramentas do *Firebase*, como o serviço de *hosting* e autenticação por *e-mail*, aumentando a praticidade na criação do aplicativo. Além disso, para escala deste projeto, pôde ser utilizado o plano gratuito desta plataforma.

O *Firebase* é uma plataforma de desenvolvimento de aplicativos da *Google* que oferece uma variedade de serviços para ajudar os desenvolvedores a criar, melhorar e expandir seus aplicativos. Dois dos principais serviços dessa plataforma são o *Realtime Database* e o *Authentication*.

O *Realtime Database* é um banco de dados hospedado na nuvem que permite armazenar e sincronizar dados em tempo real entre os clientes conectados, oferecendo uma estrutura flexível para facilitar o armazenamento e sincronização em aplicativos web [13]. Enquanto isso, o serviço de autenticação proporciona uma maneira simples e segura de autenticar usuários em aplicativos, permitindo opções de *login* por "e-mail/senha", *login* social (por meio de acesso a contas em outras plataformas, como Google, Facebook, etc.) e métodos de autenticação personalizados, garantindo a segurança e proteção dos dados do usuário e facilitando a integração da autenticação nos aplicativos desenvolvidos [14].

3.1.3 Aplicativo

Já para o desenvolvimento do aplicativo, visando agilizar o processo de programação e melhorar a organização dos códigos, assim como criar um ambiente simples onde os dados podem ser facilmente interpretados, foram utilizadas as ferramentas *Vite*, *React* e *SASS*. O *Firebase Hosting* foi adotado não apenas por sua praticidade, mas também pela utilização de outros serviços do *Firebase*.

Vite: É uma ferramenta de construção de aplicativos *web* baseados em *JavaScript* e *TypeScript*. Ele visa proporcionar um desenvolvimento rápido e eficiente, fornecendo carregamento instantâneo durante o desenvolvimento e *builds* otimizados para produção. O *Vite* é especialmente otimizado para projetos *React*. Além disso, ao construir o projeto para produção, o Vite pode compilar e otimizar todos os arquivos *TypeScript*, gerando um código *JavaScript* otimizado para implantação [15].

React: É uma biblioteca *JavaScript* de código aberto para criar interfaces de usuário, especialmente para aplicativos de página única. Ele permite a criação de componentes que tornam o desenvolvimento de interfaces de usuário mais eficiente e fácil de manter [16].

SASS: O *SASS* (*Syntactically Awesome Style Sheets*) é uma extensão da linguagem *CSS* que adiciona funcionalidades e recursos como variáveis, aninhamento, entre outros. Oferecendo uma forma mais organizada e flexível de escrever estilos para aplicativos, facilitando a manutenção e o desenvolvimento deles [17].

Firebase Hosting: É um serviço de hospedagem fornecido pelo *Firebase*. Com esta ferramenta é possível implantar web apps estáticos e dinâmicos, incluindo sites e aplicativos de página única. [18].

3.2 Métodos

Com essas ferramentas à disposição, o projeto foi desenvolvido em quatro etapas principais. A montagem do circuito com os sensores e o microcontrolador para a captação dos dados biométricos, a criação da base de dados, estabelecimento da comunicação entre o ESP e a base de dados para a transmissão e armazenamento dos dados biométricos e o desenvolvimento do aplicativo, já com a capacidade de realizar buscas pelas informações na base de dados e, por fim, o estabelecimento da comunicação do microcontrolador e do aplicativo com a base de dados.

Nesta parte serão discutidos com mais detalhes como foram realizadas cada uma dessas etapas, incluindo a programação dos diferentes códigos utilizados, a conexão dos sensores MAX30100 e NTC 10K MF58 com o ESP-12F NodeMCU e os passos para a configuração da base de dados no *Firebase*.

3.2.1 Montagem do Circuito

Como representado na Figura 2, ambos os sensores são alimentados pelas saídas de 3V3 do microcontrolador que fornece uma tensão de 3,3V, enquanto este é alimentada por uma fonte externa de 5V, neste projeto foi utilizada a conexão Micro USB do ESP para esta alimentação. Para realizar o divisor de tensão do sensor de temperatura, conecta-se a saída 3V3 do microcontrolador com uma das portas do sensor de temperatura, a outra porta deve ser conectada a um resistor de 10kΩ. Deve-se conectar a porta A0 do ESP a este mesmo ponto do circuito para realizar a leitura do divisor de tensão. Por fim, o outro terminal do resistor é conectado a um pino GND do microcontrolador. Já o MAX30100 deve ter suas portas VIN, SCL, SDA, INT e GND ligadas aos pinos 3V3, D1, D2, D0 e GND do ESP, respectivamente. Já a figura 3 apresenta o esquemático do circuito.

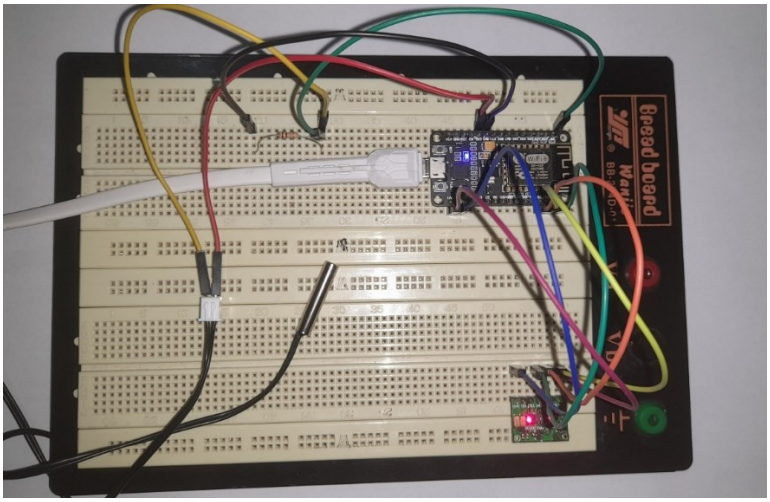

Figura 2. Circuito para aferição de dados biométricos (Autoria própria)

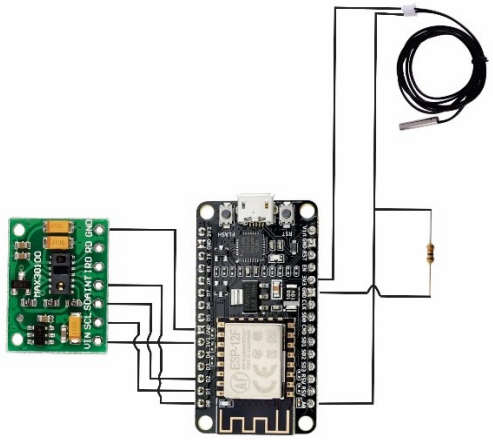
Figura 3. Esquemático de conexões do circuito (Autoria própria)

De forma simplificada, o código para a leitura destes sensores pode ser feito seguindo o fluxograma da Figura 4. Seguindo o diagrama, o algoritmo deve iniciar declarando a utilização das bibliotecas citadas anteriormente, seguida pela declaração das variáveis, como o tempo estabelecido entre as leituras e os parâmetros para o cálculo da temperatura.

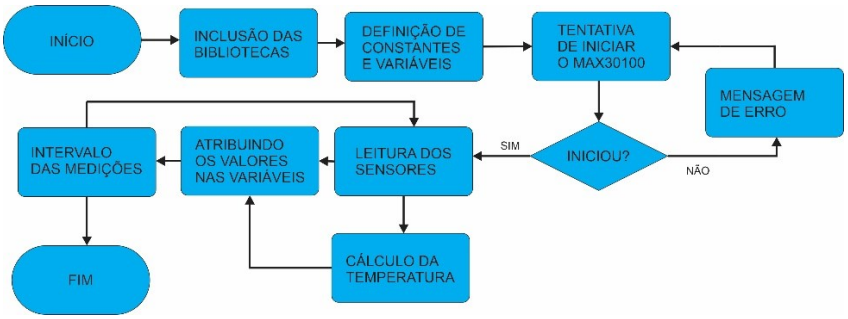

Figura 4. Fluxograma do código para aferição dos dados biométricos (Autoria própria)

O cálculo para a temperatura é realizado com base na equação do parâmetro B, uma forma simplificada da equação de Steinhart-Hart [19], utilizando os valores nominais contidos na sua folha de dados e uma média de valores obtidos em um curto período de tempo para atingir uma maior precisão.

Na programação do MAX30100 foram utilizadas as bibliotecas "*Wire*" e "*MAX30100_PulseOximeter*". A primeira é utilizada para estabelecer a comunicação serial entre o sensor e o microcontrolador [20]. A segunda encapsula funções para configurar o sensor, iniciar a medição e obter os dados de frequência cardíaca e oximetria de forma simplificada [21]. Objetivando um melhor funcionamento do sensor, as funções *pox.update*, *pox.getHeartRate* e *pox.getSpO2* devem ser realizadas continuamente para que o sensor apresente um comportamento estável durante seu funcionamento.

3.2.2 Base de dados

Tendo definido como os dados serão captados, a etapa seguinte a ser realizada é a configuração de um local para armazenar estas informações. Para criar a base de dados com o *Firebase*, foi criado um projeto no console da plataforma, seguindo o passo-a-passo disponibilizado no seu guia oficial [22] e mostrado também durante o processo de criação. O método escolhido para adicionar o *kit* de desenvolvimento de software (SDK) do *Firebase* foi por meio do *npm*, que é um gerenciador de pacotes para a linguagem *JavaScript*. Decidiu-se, também, a utilização de um aplicativo *web* para o trabalho.

Tendo um projeto no *Firebase*, foi possível utilizar os serviços disponibilizados por esta plataforma. Começando pelo *Realtime Database*, foram utilizados os passos disponibilizados também disponibilizados pela própria plataforma [23]. O único desvio nesta etapa aconteceu nas configurações de segurança onde optou-se por definir, após a configuração inicial, de que apenas dois *e-mails*, um para a transmissão do controlador e outro para controle externo, poderiam ter acesso às funções de leitura e edição de dados na base de dados.

Por fim, estes *e-mails* foram definidos no serviço *Authentication*. Seguindo novamente o guia oficial da plataforma, foram definidos os métodos de login "e-mail/senha" e "Google". A autenticação por meio de *gmail* será utilizada para autenticar os usuários do *web app*, permitindo ou bloqueando o seu acesso à visualização dos dados. O método "E-mail/senha" será utilizado para permitir que o microcontrolador tenha acesso à leitura e edição da base de dados. Na seção de usuários, foram adicionados os *e-mails* definidos anteriormente.

3.2.3 Transmissão dos dados do microcontrolador para a base de dados

Para fazer com que o microcontrolador possa transmitir os dados biométricos para a base de dados. É necessário buscar no *Firebase* a URL do *Realtime Database* e a chave API do projeto. Também é necessário configurar o *e-mail* e a senha que o microcontrolador utilizará para armazenar os dados, além das credencias da rede *Wi-Fi* a qual o ESP deve se conectar.

Além disso, é importante mencionar as bibliotecas que serão adicionadas ao código dos sensores. A biblioteca "*ESP8266WiFi*" permite a comunicação do ESP-12F via *Wi-fi*. O "*Firebase_ESP_Client*" permite o envio e o recebimento de dados entre o ESP e a base de dados [24] e a biblioteca "time" permite, entre outras coisas, obter informações acerca da data e hora no momento em que os dados são transmitidos.

Em primeiro momento, serão enviados números aleatórios dentro de um intervalo arbitrário, para ter um maior controle sobre os valores e testar o estabelecimento dessa comunicação antes de integrar os sensores ao restante do código.

Como os algoritmos dos sensores já foram explicados com maior detalhe anteriormente, o fluxograma da Figura 5 apresenta as etapas do código desenvolvido para a transmissão de dados do ESP para a base de dados. Primeiramente, além das bibliotecas e das credenciais supracitadas, também deve-se definir um objeto para o *Firebase* e declarar as variáveis necessárias. Por melhor organização, os dados serão armazenados dentro de um nó denominado "HESP" na base de dados, com um marcador que contem a data e o horário em que a leitura que foi enviada. Essa transmissão ocorrerá a cada 15 segundos, de forma a limitar problemas devido a uma conexão lenta com a *internet*. Também será enviado o *e-mail* do usuário que usará o aplicativo, que servirá como uma forma de limitar o acesso aos dados.

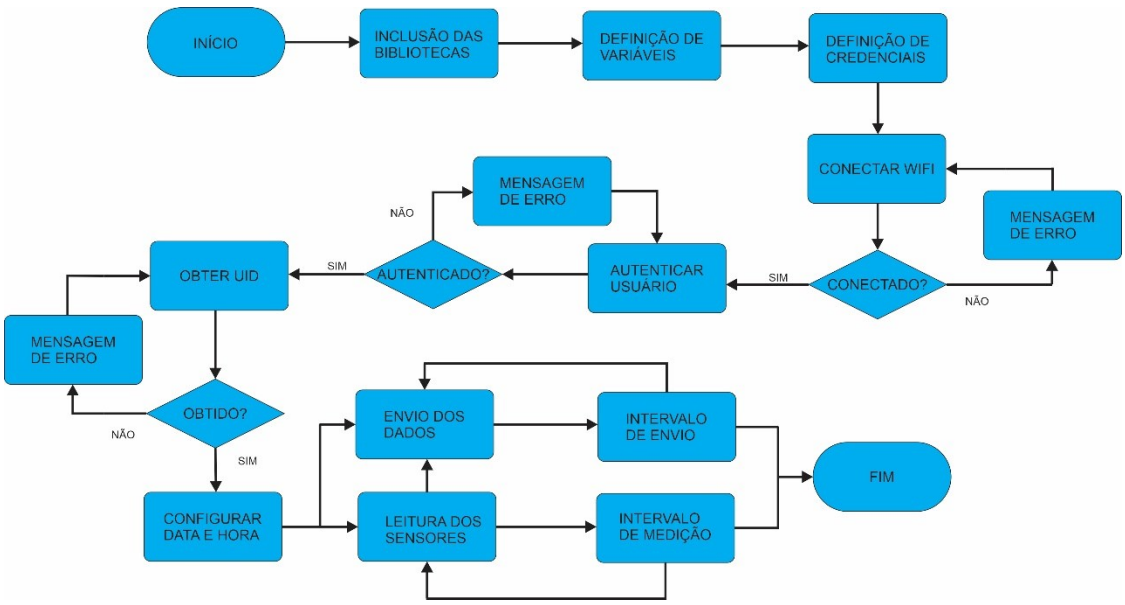

Figura 5. Fluxograma do código para a transmissão de dados para o *Realtime Database* (Autoria própria)

Assim, é definido no código a inicialização da conexão com a *internet* e com a base de dados, também é obtido o horário local para ser enviados juntamente aos dados para o *Realtime Database*. Como os *e-mails* são constantes, o endereço de *e-mail* é enviado apenas uma vez, ao estabelecer a conexão com a plataforma, não sendo atualizado após este momento. Além disso, também é obtido o *UID* (Identificador de usuário, do termo em inglês: *User Identification*) para verificar se a conexão ocorreu com as credenciais pré-determinadas.

Será testado, também, o envio de números aleatórios dentro de um intervalo arbitrário, para permitir uma melhor avaliação das transmissões à base de dados.

3.2.4 Desenvolvimento do aplicativo e a comunicação com a base de dados

Estabelecido o armazenamento dos dados, resta apenas configurar o aplicativo para que ele proporcione uma visualização simples e intuitiva dos dados coletados e limite o acesso, permitindo o acesso apenas para o *gmail* definido na configuração do serviço *Authentication* do *Firebase*.

A autenticação do usuário para verificar se o usuário deve ter acesso aos dados, será exibida uma tela inicial, solicitando ao usuário que entre em sua conta *Google*. Se o *e-mail* utilizado pelo usuário for igual ao que está armazenado na base dados, ele terá acesso a uma página que constará com um mostrador e um gráfico de linhas para cada um dos dados biométricos coletados. Senão, o usuário será mandado de volta para a página inicial.

Os mostradores citados anteriormente exibirão indicações dos níveis de frequência cardíaca, nível de saturação de oxigênio do sangue e temperatura corporal consideradas normais, moderadas e ruins como definidas em [25], [26] e [27]. Os gráficos de linha contarão com o registro das últimas 40 amostras, que correspondem às leituras dos últimos 10 minutos em que o microcontrolador estava ligado. Esta quantidade foi definida arbitrariamente para o teste do aplicativo, podendo ser aumentada ou diminuída na programação.

O diagrama de blocos da Figura 6 demonstra a lógica de programação para a comunicação do *web app* com a base de dados. Como grande parte do algoritmo do aplicativo é dedicado à personalização da página e devido a sua grande extensão, serão descritos em maior detalhe nesse tópico somente os trechos correspondentes a interlocução entre o aplicativo e a base de dados.

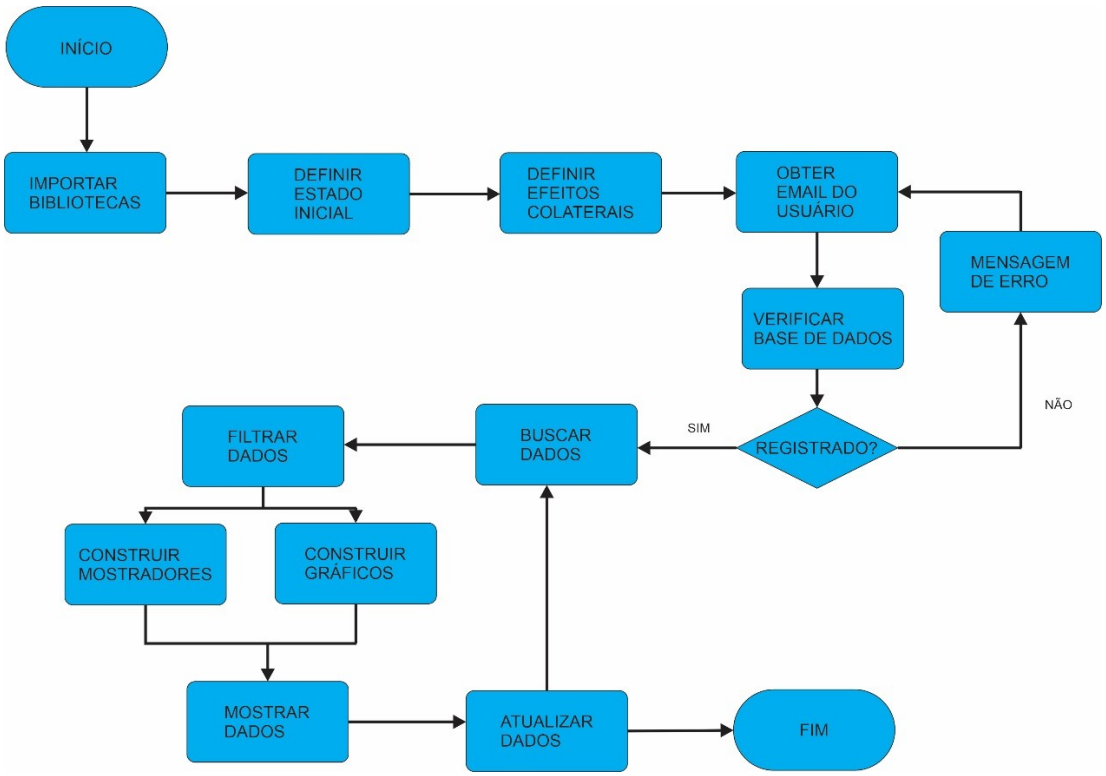

Figura 6. Fluxograma do código de comunicação do aplicativo com o Realtime Database (Autoria própria)

Inicialmente, são importados da biblioteca "Firebase Auth" alguns métodos para a autenticação de usuários. Foi criada uma função para monitorar o resultado da autenticação do usuário, certificando se o *e-mail* utilizado é válido, permitindo ou bloqueando a visualização dos dados. Caso o acesso aos dados seja bloqueado, o usuário é redirecionado à página inicial, que contará com uma mensagem de erro, informando-o da razão do bloqueio e indicando-o a realizar o *login* com uma conta válida.

As informações da base de dados são acessadas após criar uma referência aos nós presentes dentro da chave "HESP", criada no *Realtime Database* pelo microcontrolador. É utilizada uma função para observar e armazenar

atualizações dos dados biométricos localizados na base de dados, para a atualização em tempo real no aplicativo.

O código utiliza o *React* para construir a interface do usuário, aproveitando recursos como "useState" e "useEffect", para gerenciar o estado e os efeitos colaterais da aplicação. Como o código foi escrito em *TypeScript* e *SASS*, que oferecem alguns recursos a mais do que a linguagem *JavaScript* e *CSS*, respectivamente, a compilação feita pelo *Vite* também tem a função de converter o código para arquivos em *JavaScript* e *CSS* que possam ser utilizados para tornar o aplicativo público.

Para fazer esta publicação do aplicativo, foi configurado o serviço *Hosting* do *Firebase*, seguindo o passo-a-passo disponível oficialmente pela plataforma em [28].

A princípio, os valores buscados na base de dados serão aqueles definidos arbitrariamente na seção anterior e, somente em segundo momento, serão utilizados os valores obtidos pelos sensores. Isto foi feito para garantir uma maior praticidade na verificação de que os valores mostrados eram iguais aos armazenados na base de dados.

4. ANÁLISE DOS RESULTADOS

Levando em consideração toda a metodologia e os objetivos supracitados, esta seção dedica-se à análise dos resultados obtidos no desenvolver deste projeto, avaliando-os em relação às metas estabelecidas.

A transmissão de dados do ESP para o *Realtime Database* foi configurada com sucesso, no primeiro momento. Com o envio criando automaticamente os nós destinados às informações de frequência cardíaca (denominada bpm), nível de oxigenação do sangue (ox), temperatura (temp) e o *e-mail* (email) dentro do nó HESP. Os envios ocorreram no tempo pré-definido de quinze segundos e os números foram enviados e armazenados corretamente, com a data desse armazenamento no formato "YYYYMMDDhhmmss" (ano, mês, dia, horas, minutos e segundos) sendo utilizadas como marcadores desses dados para mantê-los em ordem cronológica dentro da base de dados, como demonstrado na Figura 7.

Figura 7. Captura de tela com a forma com que os dados foram armazenados (Autoria própria)

A captação dos dados para visualização no aplicativo também ocorreu de forma bem-sucedida, com os mostradores referentes à cada um desses dados, assim como os seus gráficos de linha mostrando os valores armazenados na base de dados, como indicam as Figuras 8 e 9. Vale destacar que, os valores dos mostradores e dos gráficos de linha são atualizados em tempo real, à medida com que os valores são armazenados na base de dados. O *hosting* do aplicativo também foi bem sucedido, com o *web app* podendo ser acessado pelo *link* gerado automaticamente pelo serviço.

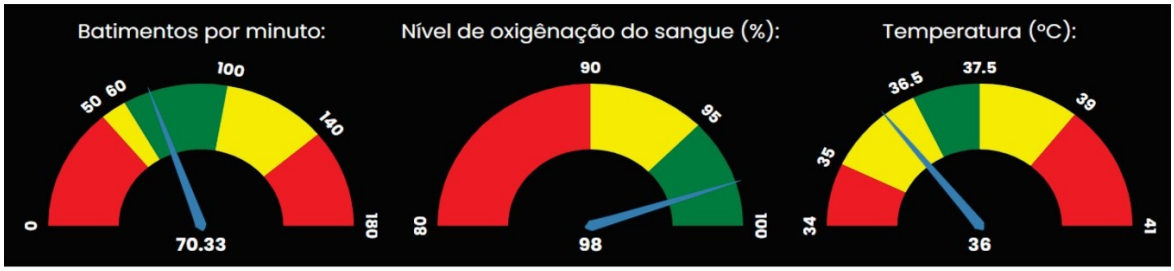

Figura 8. Mostradores do aplicativo (Autoria própria)

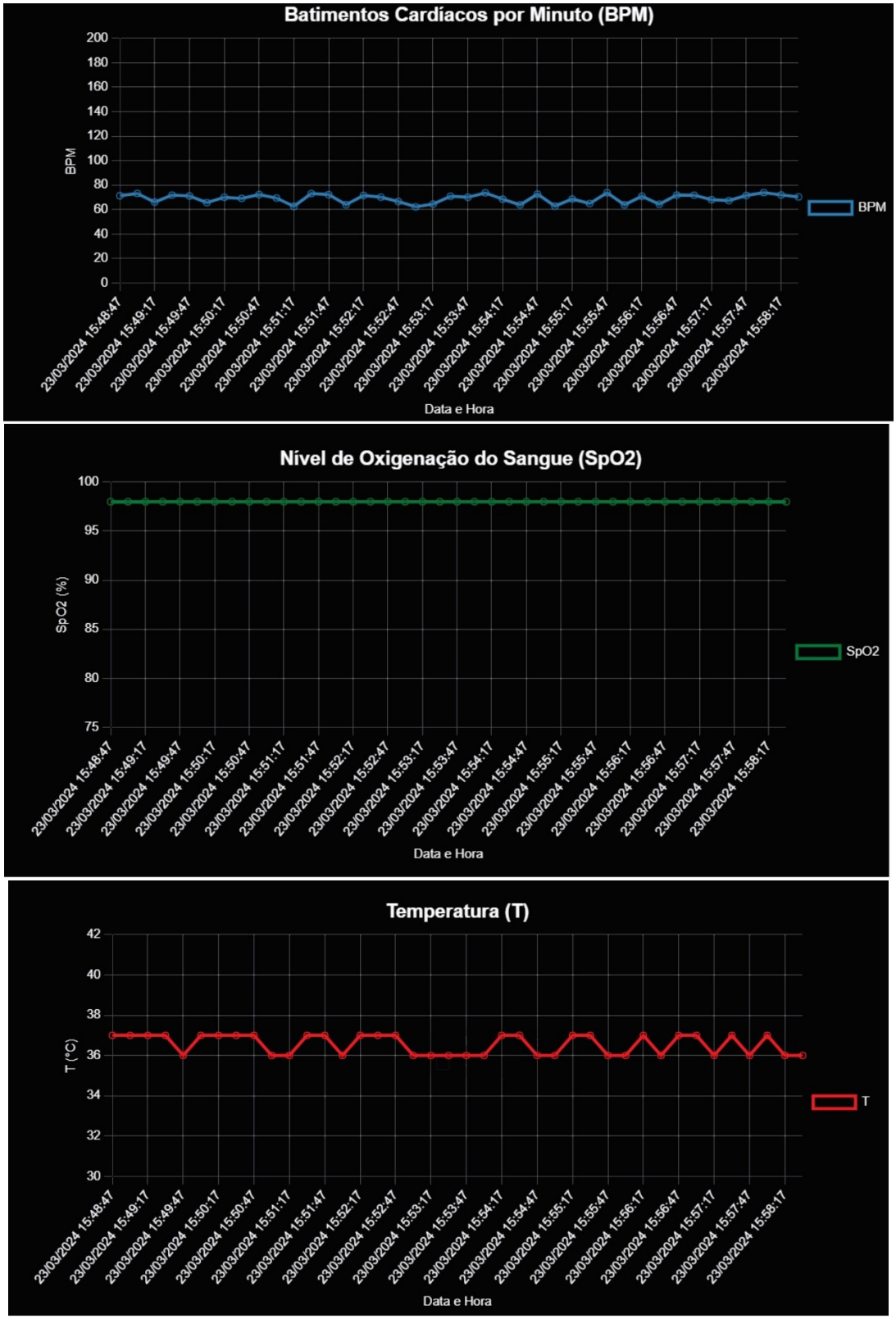

Figura 9. Gráficos do aplicativo (Autoria própria)

Na Figura 8, também é possível ver que os marcadores de nível dos mostradores foram implementados corretamente, indicando por meio das cores verde, amarela e vermelha, os níveis aceitos como normais, moderados e ruins, respectivamente. Os gráficos de linha também representam adequadamente as variações dos dados armazenados no intervalo definido, com cada ponto do gráfico contendo os valores das medições transmitidas nas datas e horários indicados. Dessa forma, o aplicativo oferece ao usuário uma interface enxuta e intuitiva, possibilitando uma mais rápida interpretação das informações coletadas em relação a uma interface que

apresentasse apenas os valores obtidos, sem a utilização de recursos visuais como os mostradores e os gráficos.

O método de autenticação teve sucesso em limitar o acesso aos dados, fazendo com que apenas os usuários cadastrados no serviço *Authentication* tenham acesso às informações coletadas. A Figura 10 mostra a mensagem de erro vista pelo usuário ao tentar acessar o aplicativo com um *e-mail* não autorizado e a mensagem de erro da conexão do ESP no monitor serial do *Arduino IDE* ao tentar acessar os dados com credenciais diferentes das definidas no *Firebase*.

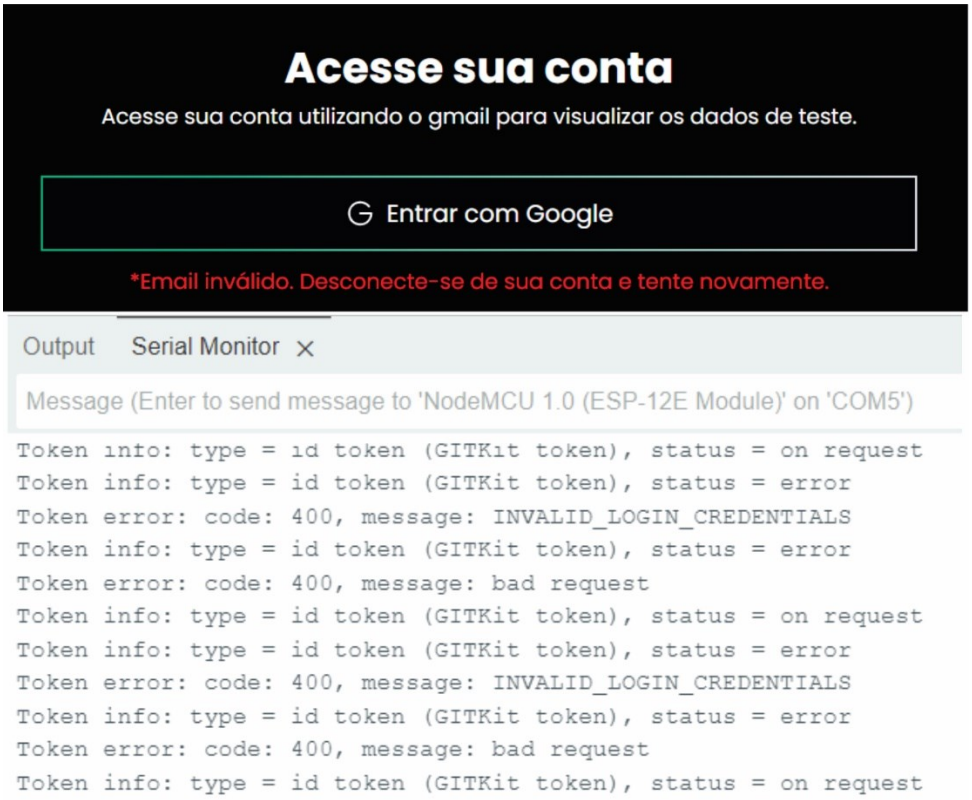

Figura 10. Mensagens de erro no aplicativo e no monitor serial (Autoria própria)

As leituras dos dados foram bem sucedidas em primeiro momento, quando estavam sendo executadas de forma isolada da transmissão para a base de dados. Porém, ao integrar essa comunicação com o código dos sensores, a leitura dos dados de frequência cardíaca e de oximetria por parte do sensor MAX30100 foi prejudicada, não atualizando esses dados corretamente. A Figura 11 mostra as leituras feitas com cada código.

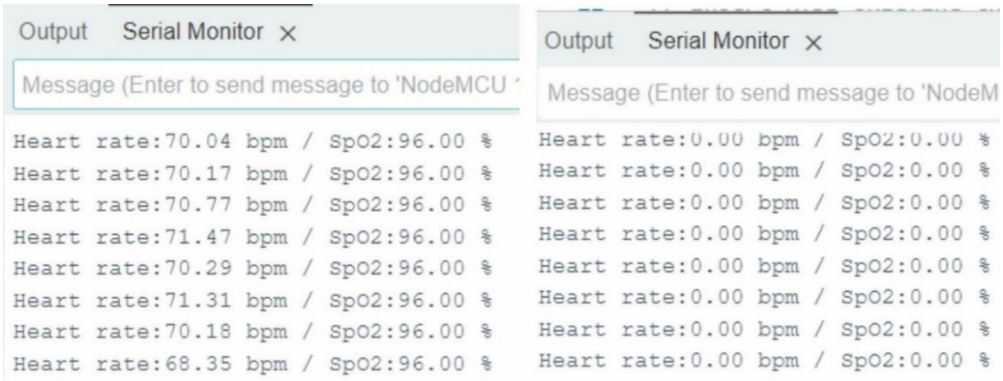

Figura 11. Leituras feitas com e sem a transmissão para o *Realtime Database* (Autoria própria)

Isto pode ter sido causado devido ao modelo ESP-12F NodeMCU ser capaz de realizar apenas uma instrução por vez, a função *pox.update* deve ser realizada continuamente para o bom funcionamento do sensor e o envio de dados pode demandar um período de tempo prolongado pela conexão à *internet*. Ao que indicam os testes realizados, esta demora do microcontrolador em realizar o envio dos dados prejudica o funcionamento do sensor MAX30100 pela natureza do processo em que este realiza suas leituras.

No entanto, vale ressaltar que o processo de transmissão não foi prejudicado, com os envios dos dados acontecendo regularmente e com os valores indicados pela medição de temperatura não sendo afetados pelo problema com o outro sensor.

5. CONCLUSÃO

Destarte, após todo o processo de desenvolvimento da base de dados e do aplicativo, desde a definição dos materiais e métodos a serem utilizados, a montagem do circuito, passando pela programação do web app e pela configuração do *Realtime Database* e culminando com a realização dos testes e a análise dos seus resultados, conclui-se que o objetivo central da criação de uma base de dados e de um aplicativo para o armazenamento e apresentação de dados biométricos para o monitoramento remoto de pacientes foi atingido, mesmo com a ressalva de que a transmissão dos dados de frequência cardíaca e oxigenação do sangue não tenha ocorrido de forma bem sucedida.

Com a conclusão deste projeto e acompanhando as tendências recentes, espera-se que haja uma intensificação no número de pesquisas acerca do tema de monitoramento remoto de pacientes, buscando explorar este de tipo de acompanhamento médico como um método alternativo ao modelo tradicional de atendimento e que a utilização de plataformas como o *Firebase* para o desenvolvimento de plataformas que facilitem o acesso o acesso e entendimento de dados biométricos que podem ser cruciais no acompanhamento médico.

Para trabalhos futuros que desejem aprimorar os métodos utilizados, pode ser buscada a utilização de outro sensor para a obtenção dos dados de oximetria e frequência cardíaca, a utilização de diferentes microcontroladores, assim como a implementação de mais sensores, para tornar possível o acompanhamento de uma gama maior de dados biométricos, atendendo à demanda de pacientes acometidos por diferentes enfermidades, que necessitariam a aferição de dados biométricos distintos aos utilizados neste projeto.

6. REFERÊNCIAS

[1] J. Shaver, "The State of Telehealth Before and After the COVID-19 Pandemic", *Prim. Care*, vol. 49, nº 4, p. 517–530, dez. 2022, doi: 10.1016/j.pop.2022.04.002.

[2] "Telehealth and remote patient monitoring | Telehealth.HHS.gov". Acesso em: 24 de março de 2024. [Online]. Disponível em: https://telehealth.hhs.gov/providers/preparing-patients-for-telehealth/telehealth-and-remote-patient-monitoring

[3] R. Pearl e B. Wayling, "The Telehealth Era Is Just Beginning", *Harvard Business Review*, 1º de maio de 2022. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://hbr.org/2022/05/the-telehealth-era-is-just-beginning

[4] P. J. Pronovost, M. D. Cole, e R. M. Hughes, "Remote Patient Monitoring During COVID-19: An Unexpected Patient Safety Benefit", *JAMA*, vol. 327, nº 12, p. 1125–1126, mar. 2022, doi: 10.1001/jama.2022.2040.

[5] "Remote Patient Monitoring", mar. 2023, Acesso em: 24 de março de 2024. [Online]. Disponível em: https://psnet.ahrq.gov/perspective/remote-patient-monitoring

[6] "Web Application Development: Process, Tools, & Examples", BrowserStack. Acesso em: 24 de abril de 2024. [Online]. Disponível em: https://browserstack.wpengine.com/guide/web-application-development-guide/

[7] B. Anderson, "SQL vs. NoSQL Databases: What's the Difference?", IBM Blog. Acesso em: 24 de abril de 2024. [Online]. Disponível em: https://www.ibm.com/blog/sql-vs-nosql/www.ibm.com/blog/sql-vs-nosql

[8] J. Schneider, "Cryptography use cases: From secure communication to data security", IBM Blog. Acesso em: 24 de abril de 2024. [Online]. Disponível em: https://www.ibm.com/blog/cryptography-use-cases/www.ibm.com/blog/cryptography-use-cases

[9] "Ai-Thinker Remote AT Command Transceiver Wireless esp8266 Chip IOT esp8266 12f Wifi Module". Acesso em: 24 de março de 2024. [Online]. Disponível em: http://www.ai-thinker.com/pro_view-58.html

[10] G. Preetham, "IOT BASED PULSE OXIMETER USING ESP8266".

[11] A. Onubeze, "Developing a Wireless Heart Rate Monitor with MAX30100 and nRF51822".

[12] J. Kim e J. D. Kim, "Voltage divider resistance for high-resolution of the thermistor temperature measurement", *Measurement*, vol. 44, nº 10, p. 2054–2059, dez. 2011, doi: 10.1016/j.measurement.2011.08.004.

[13] "Firebase Realtime Database". Acesso em: 24 de março de 2024. [Online]. Disponível em: https://firebase.google.com/docs/database?hl=pt

[14] "Firebase Authentication". Acesso em: 24 de março de 2024. [Online]. Disponível em: https://firebase.google.com/docs/auth?hl=pt

[15] "Introducing Vite JS - Next-Gen Frontend Tooling", Radixweb. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://radixweb.com/vite-js-Informative

[16] "Visão geral da referência do React – React". Acesso em: 24 de março de 2024. [Online]. Disponível em: https://pt-br.react.dev/reference/react

[17] "Sass: Documentation". Acesso em: 24 de março de 2024. [Online]. Disponível em: https://sass-lang.com/documentation/

[18] “Firebase Hosting”. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://firebase.google.com/docs/hosting?hl=pt
[19] “Thermistor”. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://www.chemeurope.com/en/encyclopedia/Thermistor.html
[20] “Wire - Arduino Reference”. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://www.arduino.cc/reference/en/language/functions/communication/wire/
[21] “oxullo/Arduino-MAX30100: Arduino library for MAX30100, integrated oximeter and heart rate sensor”. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://github.com/oxullo/Arduino-MAX30100
[22] “Adicionar o Firebase ao seu projeto JavaScript”. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://firebase.google.com/docs/web/setup?hl=pt
[23] “Documentação de construção | Documentação do Firebase”. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://firebase.google.com/docs/build?hl=pt
[24] S. K, “mobizt/Firebase-ESP-Client”. 24 de março de 2024. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://github.com/mobizt/Firebase-ESP-Client
[25] “What your heart rate is telling you”, Harvard Health. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://www.health.harvard.edu/heart-health/what-your-heart-rate-is-telling-you
[26] “O que é importante saber sobre o uso do oxímetro de pulso para monitorar a COVID-19 em casa”, UNA-SUS - Especial COVID-19. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://www.unasus.gov.br/especial/covid19/markdown/502
[27] “Body temperature norms: MedlinePlus Medical Encyclopedia”. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://medlineplus.gov/ency/article/001982.htm
[28] “Comece a usar o Firebase Hosting”, Firebase. Acesso em: 24 de março de 2024. [Online]. Disponível em: https://firebase.google.com/docs/hosting/quickstart?hl=pt