UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIA E TECNOLOGIA
CURSO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

ISAAC KEVIN DE SOUZA LIMA

DISPOSITIVO BLUETOOTH PARA ACOMPANHAMENTO DE PACIENTES ATRAVÉS DA FREQUÊNCIA CARDÍACA

MOSSORÓ
2025

ISAAC KEVIN DE SOUZA LIMA

DISPOSITIVO BLUETOOTH PARA ACOMPANHAMENTO DE PACIENTES ATRAVÉS DA FREQUÊNCIA CARDÍACA

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência e Tecnologia.

Orientador: Prof. Dr. Idalmir de Souza Queiroz Júnior.

MOSSORÓ

2025

L732d Lima, Isaac Kevin de Souza.
Dispositivo Bluetooth para acompanhamento de pacientes através da frequência cardiaca / Isaac Kevin de Souza Lima. - 2025.
48 f. : il.

Orientador: Idalmir de Souza Queiroz Júnior.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência e Tecnologia, 2025.

1. Arduino. 2. Programação. 3. Monitor Cardíaco. 4. Frequência Cardíaca. 5. Acessível. I. Queiroz Júnior, Idalmir de Souza, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Mossoró / Setor de Informação e Referência
Bibliotecária: Keina Cristina Santos Sousa e Silva
CRB: 15/120

ISAAC KEVIN DE SOUZA LIMA

DISPOSITIVO BLUETOOTH PARA ACOMPANHAMENTO DE PACIENTES ATRAVÉS DA FREQUÊNCIA CARDÍACA

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência e Tecnologia.

Defendida em: 04 / 08 / 2025 .

BANCA EXAMINADORA

_____
Idalmir de Souza Queiroz Júnior, Prof. Dr. (UFERSA)
Presidente

_____
Elias Samuel Soares Cesário, Prof. (UFERSA)
Membro Examinador

_____
Francisco Magno Monteiro Sobrinho, Prof. Dr.
Centro de Educação e Tecnologias (CET) Ítalo Bologna – SENAI-RN
Membro Examinador

AGRADECIMENTOS

Ao meu pai, por ser o alicerce fundamental no meu desenvolvimento pessoal e profissional. Agradeço por todos os ensinamentos e pelo apoio incondicional que me transformaram no homem que sou hoje, e a quem espero sempre poder dar orgulho.

À minha mãe, cujo cuidado e amor me moldaram desde os primeiros anos de vida. Sua presença constante e seu incentivo foram indispensáveis para o meu crescimento e para que eu chegasse até aqui.

Às minhas irmãs, companheiras de jornada que, com seu apoio e carinho, sempre foram parte essencial das minhas conquistas. Agradeço por estarem sempre presentes em todos os momentos.

Ao meu orientador, Professor Dr. Idalmir de Souza Queiroz Júnior, a quem expresso minha profunda gratidão não apenas pela orientação, mas pela confiança em mim depositada. A oportunidade de explorar e apresentar um tema tão complexo sobre sua tutela foi gratificante.

Aos membros da Banca Examinadora, por dedicarem seu tempo e conhecimento à avaliação deste trabalho. Agradeço pela oportunidade de apresentar esta monografia e por contribuírem de forma decisiva para a conclusão desta importante etapa acadêmica, que abre as portas para o meu futuro.

Aos meus amigos, em especial Francisco Victor, que com lealdade e companheirismo tornaram a jornada mais tranquila. Agradeço por todo o apoio, pelas conversas e pela ajuda sempre disposta nos momentos em que mais precisei.

RESUMO

Os sistemas de monitoramento cardíaco são utilizados a anos no meio hospitalar para acompanhamento de pacientes com condições cardíacas. Porém a pratica desse monitoramento em domicilio é incomum, devido ao alto custo e complexidade dos equipamentos. Este TCC aborda a necessidade de sistemas de monitoramento cardíaco acessíveis para uso domiciliar, visto que os equipamentos hospitalares são de alto custo e complexidade. Baseado nisso se propõe a criação de um protótipo de monitor cardíaco com o Arduino, utilizando dois sensores, um eletrocardiograma, o AD8232, e um sensor de fotopletismografia, o MAX30102, onde ambos enviarão através de números e gráficos dados pertinentes sobre condição cardíaca do usuário pelo *Bluetooth* com o modulo HC-05, que possa ser visualizada por meio da interface do app *Bluetooth Electronics*, fazendo com que haja uma opção viável tanto economicamente como em quesito de portabilidade e compreensão, auxiliando no tratamento domiciliar e ao alcance inclusive de comunidades rurais. O objetivo do dispositivo é fornecer uma opção viável em termos econômicos, de portabilidade e de compreensão, auxiliando no tratamento domiciliar e alcançando comunidades rurais. Este projeto visa complementar, e não substituir, os equipamentos hospitalares, oferecendo ao usuário uma medida acessível para compreender suas condições e tomar as precauções necessárias.

Palavras-chave: Arduino, Programação, Monitor Cardíaco, Frequência Cardíaca, Acessível.

ABSTRACT

Cardiac monitoring systems have been used for years in hospitals to monitor patients with heart conditions. However, home monitoring is uncommon due to the high cost and complexity of the equipment. This thesis addresses the need for affordable cardiac monitoring systems for home use, given the high cost and complexity of hospital equipment. Based on this, we propose the creation of a prototype cardiac monitor with Arduino, utilizing two sensors: an electrocardiogram (AD8232) and a photoplethysmography (MAX30102). Both will send relevant data about the user's heart condition via *Bluetooth* using the HC-05 module, via numbers and graphs. This data can be viewed through the *Bluetooth Electronics* app interface. This makes it a viable option both economically and in terms of portability and comprehension, aiding in home treatment and reaching even rural communities. The device's goal is to provide a viable option in terms of cost, portability, and comprehensibility, aiding in home treatment and reaching rural communities. This project aims to complement, not replace, hospital equipment, offering users an accessible way to understand their condition and take necessary precautions.

Keywords: Arduino. Programming, Heart Monitor, Heart Rate, Accessible.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AC	Corrente Alternada (Componente Alternado do Sinal)
ADC	Conversor Analógico-Digital
BPM	Batimentos por Minuto
DC	Corrente Contínua (Componente Contínuo do Sinal)
Dr.	Doutor
ECG	Eletrocardiograma
GND	Terra (Referência de 0 Volts)
I2C	Circuito Inter-Integrado (Protocolo de Comunicação)
IDE	Ambiente de Desenvolvimento Integrado
IR	Infravermelho
LA	*Left Arm* (Braço Esquerdo)
LED	Diodo Emissor de Luz
LO+	*Lead-Off Positive* (Detecção de Eletrodo Positivo)
LO-	*Lead-Off Negative* (Detecção de Eletrodo Negativo)
Me.	Mestre
PPG	Fotopletismografia
PWM	Modulação por Largura de Pulso
RA	*Right Arm* (Braço Direito)
RL	*Right Leg* (Perna Direita)
RX	Recepção de Dados (Pino Receptor)
SCL	*Serial Clock* (Linha de Relógio I2C)
SDA	*Serial Data* (Linha de Dados I2C)
SpO_2	Saturação Periférica de Oxigênio
TX	Transmissão de Dados (Pino Transmissor)
UFERSA	Universidade Federal Rural do Semi-Árido
VIN	Tensão de Entrada

LISTA DE SÍMBOLOS

%	Porcentagem
°C	Graus Celsius
Hz	Hertz (Ciclos por segundo)
ms	Milissegundos
mV	Milivolts
R	Razão das Razões (Oximetria)
V	Volts
µs	Microssegundos
*	Asterisco

Sumário

1. INTRODUÇÃO

No meio hospitalar, existem métodos de monitoramento cardíaco amplamente utilizados para o acompanhamento de pacientes. Dentre os principais, destacam-se o eletrocardiograma (ECG), que monitora a atividade elétrica do coração para determinar seu funcionamento (GUYTON; HALL, 2011), e a fotopletismografia (PPG), uma técnica óptica que avalia o fluxo sanguíneo e permite aferir, entre outros parâmetros, a frequência cardíaca e a saturação de oxigênio (ALLEN, 2007).

Apesar da sua eficácia, o maior problema desses sistemas de monitoramento reside na sua complexidade e alto custo, o que os torna, em grande parte, inacessíveis para o tratamento domiciliar contínuo, também conhecido como *home care*. Para pessoas de baixa renda, a aquisição e utilização de equipamentos portáteis se torna inviável, já que o valor de um eletrocardiograma portátil em 2025 pode variar entre R$ 1.000 e R$ 10.000, a depender do modelo e das funções implementadas. Um problema comum, portanto, é a escassez de equipamentos para monitoramento cardíaco que sejam acessíveis para o uso fora de hospitais de alta patente, cenário onde a busca por sensores vestíveis e de baixo custo tem crescido como uma alternativa viável (CASTANEDA et al., 2018).

Diante dessa realidade, este trabalho tem como objetivo principal desenvolver um protótipo funcional de um monitor cardíaco de baixo custo e portátil, destinado ao uso domiciliar para acompanhamento complementar. Para alcançar essa meta, o projeto consiste na montagem de um circuito com a plataforma Arduino (FERRONI et al., 2015), que integra um sensor de eletrocardiograma (AD8232) e um de fotopletismografia (MAX30102). Os dados coletados, como frequência cardíaca e saturação de oxigênio, serão transmitidos via módulo *Bluetooth* HC-05 para um aplicativo móvel, onde serão exibidos de forma clara e em tempo real. Com isso, busca-se oferecer e validar uma ferramenta que seja não apenas economicamente viável, mas também portátil e de fácil compreensão, reforçando que sua função é complementar, e não substituir, os equipamentos médicos profissionais, fornecendo ao usuário uma medida acessível para entender sua condição e tomar as devidas precauções.

2. REFERÊNCIAL TEÓRICO

Anteriormente à construção e aplicação prática do projeto, é fundamental obter um sólido entendimento das tecnologias que o compõem. Essa compreensão abrange tanto a plataforma Arduino e seu funcionamento, quanto os princípios dos dois métodos clínicos utilizados: o eletrocardiograma (ECG) e a fotopletismografia (PPG).

2.1 O Arduino

O Arduíno é um sistema de eletrônica simples baseado no microcontrolador ATMEL ATMEGA328 que surgiu em 2005, na Itália, por iniciativa do professor Massimo Banzi e de David Cuartielles, que desejava ensinar eletrônica e programação a seus alunos de design. Com a ajuda do aluno David Mellis, que desenvolveu a linguagem de programação, a placa foi criada para superar duas dificuldades principais: a complexidade de ensinar o tema para pessoas que não são da área e a "disponibilidade limitada de placas potentes a um custo acessível" para projetos de arte, interatividade e robótica (RIOS et al., 2012, p. 3).

A partir dessa origem, seu objetivo principal se consolidou em mostrar as diferentes formas de aplicação da eletrônica no meio cotidiano, mostrando sua importância na evolução tecnológica. Com o Arduino, é possível criar diversos projetos, desde ligar um simples LED até criar sistemas de trava automática, sistemas com comando de voz e robôs.

O Arduíno é definido como uma placa de controle que faz o gerenciamento de dados de entrada e saída — como sensores e LEDs, respectivamente. Ele é, essencialmente, um sistema de controle que utiliza programação para comandar outros dispositivos, utilizando um sistema conhecido como Mestre e Escravo. Os microcontroladores possuem uma faixa de tensão de funcionamento recomendada entre 7V e 12V, por possuírem um regulador de tensão embutido que transforma a tensão recebida em uma tensão funcional de 5V. Os microcontroladores, em sua maioria, possuem pelo menos duas portas de alimentação, 5V (padrão) e o terra (GND), mas também podem possuir um pino com tensão de 3,3V para equipamentos que necessitem dessa faixa de tensão (RIOS et al., 2012, p. 3-8).

Os microcontroladores do Arduino possuem dois tipos de pinos de controle: digitais e analógicos. Os pinos digitais são como interruptores que entendem apenas dois tipos de lógica, HIGH (Alta) e LOW (Baixa). Essa lógica serve para ligar ou desligar dispositivos diversos. O Arduino em si só trabalha com lógica digital, o que impossibilitaria a leitura de valores intermediários. Por isso, os microcontroladores Arduino possuem um Conversor Analógico-

Digital, que funciona como um “ajuste fino” aos valores de tensão de entrada de 0V a 5V, convertendo-os em valores entre 0 e 1023. O Arduino também é capaz de simular pinos analógicos de saída devido a uma técnica chamada PWM (Modulação por Largura de Pulso) (RIOS et al., 2012, p. 23).

O potencial de um hardware só é plenamente realizado por meio do software que o instrui. A programação é, portanto, a disciplina que estabelece a ponte entre os componentes físicos e a execução de algoritmos e tarefas lógicas. Conforme Ferroni et al. (2015, p. 8), "tudo o que será realizado pelo Arduino, seja através do próprio microcontrolador ou de periféricos a ele conectados, deverá ser programado em *scripts* inseridos por intermédio da IDE".

Para isso, o Arduino possui seu próprio Ambiente de Desenvolvimento Integrado (IDE), que é a ferramenta utilizada para escrever, alterar e "convertê-los em instruções que o Arduino consiga compreender e executar". A linguagem de programação utilizada se baseia em C/C++. Embora o ambiente da IDE seja considerado intuitivo, ele exige do usuário conhecimento em algoritmos e programação. Sua interface oferece funções essenciais para o desenvolvimento, como um compilador que verifica o corpo do código em busca de erros, um botão para gravar o código na placa (que envia todos os comandos e instruções ao microcontrolador), o monitor serial e o plotter (FERRONI et al., 2015). Dessa forma, a IDE é a ferramenta que permite ao programador projetar, alterar, compilar e gravar os programas, dando à placa e a todos os componentes que dependem dela um objetivo claro a ser cumprido.

2.2 Eletrocardiograma

O eletrocardiograma (ECG) é um exame que, segundo Beraldo (1997, p. 1), é "o registro da atividade elétrica do coração", sendo uma representação do comportamento cardíaco por meio de pulsos elétricos. O coração gera esses pulsos para causar as contrações cardíacas que bombeiam o sangue pelo corpo. O ECG detecta essa atividade elétrica através de eletrodos conectados à pele, os quais traduzem os pulsos em um gráfico de ondas. Para essa detecção, são utilizados no mínimo três eletrodos: RA (*Right Arm*) para o potencial negativo, LA (*Left Arm*) para o potencial positivo e RL (*Right Leg*) como referência. Esse registro gráfico é dividido em onda P, complexo QRS, segmento S-T e onda T, que correspondem a eventos elétricos específicos do ciclo cardíaco (BERALDO, 1997).

Conforme citado, a onda de ECG é dividida em três partes primordiais. A onda P representa a primeira contração interna do coração, na qual o sangue é bombeado do átrio para os ventrículos. O complexo QRS corresponde à atividade elétrica que inicia a contração do

coração e o bombeamento geral do sangue para o corpo. As ondas P e o complexo QRS são de despolarização. Por fim, a onda T é a de repolarização, momento em que o coração relaxa e se prepara para reiniciar o ciclo. Entre o complexo QRS e a onda T, há o segmento ST, durante o qual o coração se mantém contraído enquanto o sangue é liberado (GUYTON; HALL, 2011). A Figura 1 exibe uma representação da onda de ECG com as ondas P, QRS e T, e indica também que a atividade elétrica se aproxima de 1 mV.

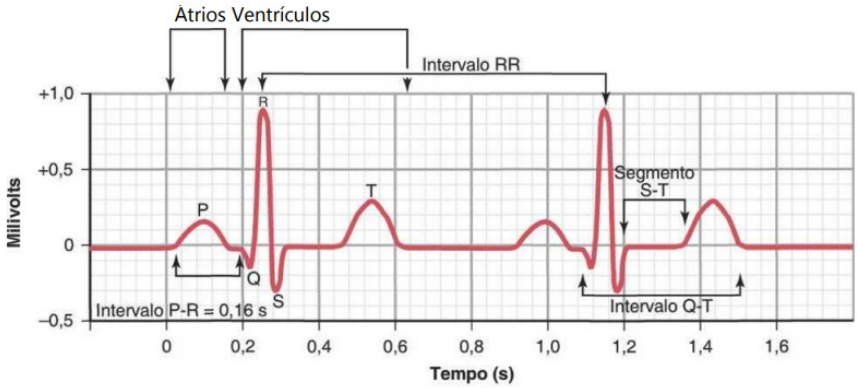

Figura 1 – Onda ECG padrão.
Fonte: Guyton e Hall (2011)

O funcionamento do exame é complexo, pois os pulsos elétricos medidos na pele são extremamente pequenos. Segundo Beraldo (1997, p. 17), "os picos do sinal de ECG estão geralmente abaixo de 5mV". Por essa razão, o equipamento de ECG precisa amplificar o sinal centenas de vezes para permitir uma análise adequada.

Além da baixa amplitude, o sinal de ECG é suscetível a diversas interferências. Guimarães et al. (1999) alertam para a importância de minimizar "interferências ambientais, como as provocadas por aparelhos de ondas curtas, fios de alta tensão, motores e outros aparelhos elétricos". Os mesmos autores também apontam interferências de origem biológica, como "tremores musculares", e problemas técnicos, como o mau contato dos eletrodos com o corpo. Devido a isso, o equipamento de ECG possui meios próprios para garantir a qualidade do sinal. A normatização exige, por exemplo, que o aparelho possua uma "entrada para o condutor de proteção" (aterramento) e que os filtros estejam ativos para que o gráfico elétrico mostrado seja o mais preciso possível (GUIMARÃES et al., 1999).

2.3 Fotopletismografia

A Fotopletismografia (PPG) é uma técnica óptica, simples e não invasiva desenvolvida para detectar alterações no volume de sangue nos tecidos (ALLEN, 2007). Embora possa ser usada para monitoramento venoso, sua aplicação se expandiu principalmente para o

monitoramento cardíaco, pois o pulso sanguíneo, gerado pela variação constante de sangue nos vasos, permite a extração de informações sobre a frequência cardíaca (ALIAN; SHELLEY, 2014).

O princípio de funcionamento do PPG baseia-se na emissão de luz por um LED, que penetra em uma parte da superfície da pele, como o dedo ou o pulso. Partes do tecido corporal, como pele, ossos e sangue, absorvem essa luz de forma distinta (TAMURA et al., 2014). A luz não absorvida é então captada por um fotodetector posicionado próximo ao LED, e a variação na intensidade luminosa registrada por ele é o que forma o gráfico da onda PPG (ALLEN, 2007).

O ciclo cardíaco é o que modula esse sinal. Na sístole, o coração se contrai e ejeta sangue, levando a um aumento do volume arterial. Com mais sangue, mais luz é absorvida, o que leva a uma queda na intensidade detectada, formando o vale do gráfico PPG. Em seguida, ocorre a diástole, fase em que o coração relaxa e o volume de sangue nas artérias diminui. Isso leva a uma menor absorção e, consequentemente, a um aumento da intensidade luminosa, configurando o pico do PPG. Contudo, para uma melhor análise clínica, considera-se o inverso, no qual a sístole corresponde ao pico do gráfico e a diástole, ao vale (ALIAN; SHELLEY, 2014).

O sinal é composto por dois componentes principais: Componente DC (Corrente Contínua): Representa a linha de base do sinal, detectada pela absorção de luz de tecidos não pulsáteis, como ossos, pele e o volume constante de sangue venoso; Componente AC (Corrente Alternada)**:** É a variação pulsátil que se sobrepõe ao sinal DC. Este componente é o de maior interesse, pois sua frequência é síncrona com a frequência cardíaca (BPM) e está diretamente relacionada à pulsação do sangue nas artérias (TAMURA et al., 2014).

A Figura 2 apresenta o gráfico de onda PPG, destacando o componente AC (CA), que define a amplitude da onda, e o componente DC (CC), que representa a linha de base constante.

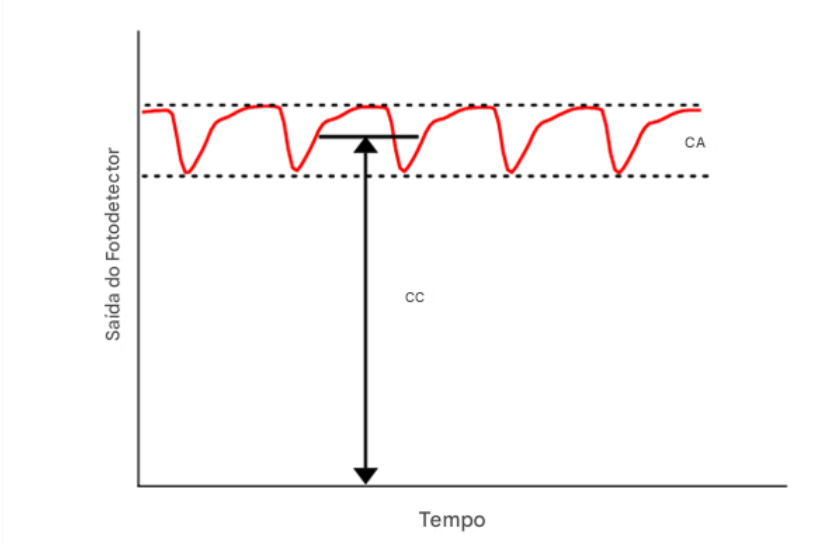

Figura 2 – Onda PPG com os componentes CA e CC.
Fonte: Alian e Shelley. (2014)

Por fim, a medição da saturação de oxigênio (SpO2) é uma extensão da técnica PPG que exige a utilização de duas fontes luminosas com comprimentos de onda diferentes: a luz vermelha e a infravermelha (ALLEN, 2007). O princípio de funcionamento é que a hemoglobina oxigenada (HbO2) e a hemoglobina desoxigenada (Hb) possuem coeficientes de absorção distintos para essas duas luzes. Devido a essa interação diferenciada, o fotodetector pode analisar a razão da intensidade luminosa absorvida no componente pulsátil (AC) do sangue. A partir dessa razão, é possível estimar com alta precisão a porcentagem de hemoglobina que está transportando oxigênio (ALLEN, 2007; CASTANEDA et al., 2018).

A partir da análise do pulso sanguíneo, pode-se determinar a frequência cardíaca em batimentos por minuto (BPM). Para isso, utiliza-se um cálculo baseado no intervalo de tempo entre pulsos consecutivos. Segundo Guyton e Hall (2017), se o intervalo entre dois batimentos for de 1 segundo, a frequência cardíaca será de 60 batimentos por minuto. Seguindo esta lógica, a equação para o cálculo dos BPM é apresentada na Equação 1:

$$BPM = \frac{60}{Intervalo\ de\ Tempo\ de\ um\ Batimento} \quad (1)$$

Para um adulto em repouso, a frequência cardíaca deve situar-se entre 60 e 100 batimentos por minuto (BPM). Valores acima desse intervalo são classificados como taquicardia e, abaixo, como bradicardia (GUYTON; HALL, 2011).

O cálculo da saturação de oxigênio (SpO2), por sua vez, utiliza a razão R, mencionada anteriormente. Segundo Oak e Aroul (2015), essa métrica é conhecida como "Razão das Razões" e é calculada por meio da Equação 2:

$$R = \frac{\frac{AC\ do\ LED\ vermelho}{DC\ do\ LED\ vermelho}}{\frac{AC\ do\ LED\ IR}{DC\ do\ LED\ IR}} \quad (2)$$

Esse cálculo baseia-se nos dois componentes do sinal de fotopletismografia: o componente pulsátil (AC) e o não pulsátil (DC). Após a obtenção de R, utiliza-se uma fórmula empírica para determinar a SpO2, conforme apresentado a seguir (OAK; AROUL, 2015):

$$\%\, SpO2 = 110 - 25 \times R \quad (3)$$

Por meio da Equação 3, é possível obter um valor aproximado para a porcentagem de saturação de oxigênio.

Na Figura 3, podem-se observar as diferenças entre uma onda de ECG e uma de PPG. O ECG corresponde a um gráfico da atividade elétrica, enquanto o PPG representa um gráfico do pulso sanguíneo, composto pelo componente pulsátil (AC) e pelo componente de linha de base (DC).

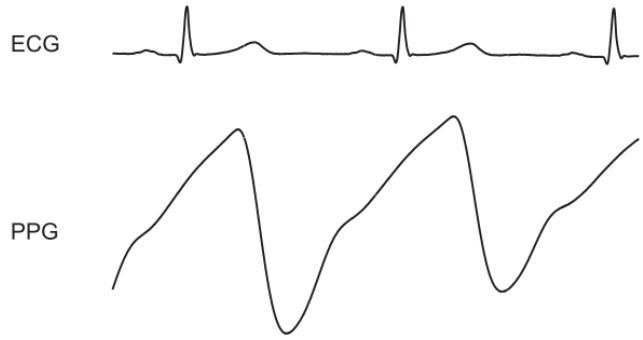

Figura 3 – Onda ECG e PPG.
Fonte: Allen (2007)

3 MATERIAIS E MÉTODOS

Diante do exposto, define-se o objetivo principal do projeto: desenvolver um monitor cardíaco com a plataforma Arduino e interface via *Bluetooth*. Para tal, os capítulos seguintes apresentarão os materiais utilizados, a metodologia de montagem do protótipo, o software desenvolvido, o aplicativo que atua como interface, os testes práticos realizados e, por fim, os resultados e desafios encontrados durante o desenvolvimento.

3.1 Materiais

Nesta seção, serão detalhados os componentes de hardware utilizados para o desenvolvimento e a construção do protótipo. Cada subseção descreverá um componente específico, suas principais características e sua função no dispositivo.

3.1.1 Arduíno MEGA 2560

A plataforma Arduino foi utilizada como base para o projeto, como já comentado, devido à sua versatilidade e simplicidade de programação e também pelo seu baixo custo, alguns modelos estão ilustrados na Figura 4.

Idealmente, a placa indicada para o protótipo seria o Arduino Nano, um dos menores modelos disponíveis. Devido à sua portabilidade, esse modelo seria mais adequado para o formato final do projeto. Contudo, devido à sua indisponibilidade, optou-se pela placa Arduino MEGA 2560 para o desenvolvimento.

Este modelo, embora maior em tamanho, atendeu plenamente às necessidades do protótipo. Por possuir uma grande quantidade de portas, é uma das placas mais indicadas para projetos grandes e complexos. Especificamente, o Arduino MEGA 2560 oferece 54 pinos de entrada/saída digital e 16 entradas analógicas, recursos que foram suficientes para atender às necessidades dos equipamentos utilizados.

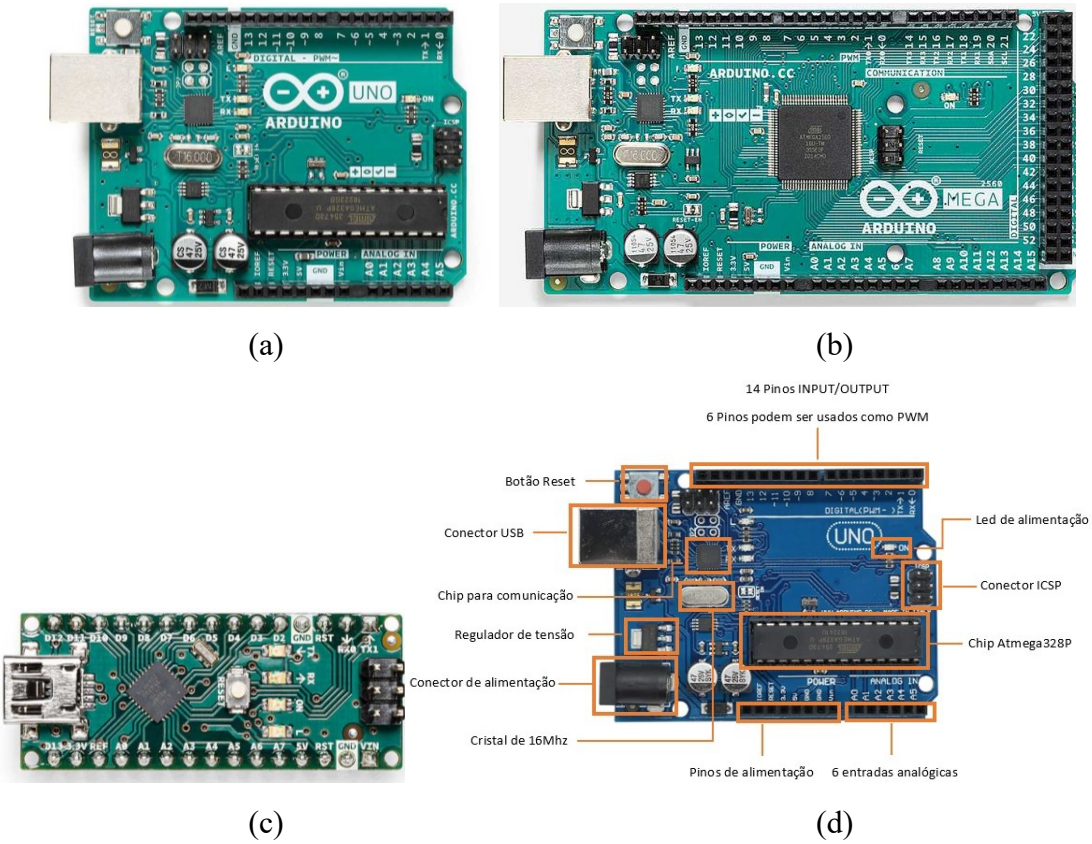

Figura 4 – Placa Arduíno (a) Uno, (b) Mega 2560, (c) Nano e (d) pinagem básica do Arduíno Uno
Fonte: Arduino (2025)

3.1.2 Sensores

Uma vez definida a placa de desenvolvimento, a seleção dos sensores foi a etapa seguinte. Para compor o projeto, foram escolhidos dois sensores principais que, embora ambos destinados à medição da atividade cardíaca, desempenham funções distintas e complementares: o sensor de eletrocardiograma (ECG) AD8232 e o sensor de fotopletismografia (PPG) MAX30102.

3.1.2.1 AD8232

O AD8232 é um circuito integrado de baixo custo projetado para medir a atividade elétrica do coração, funcionando como um ECG dedicado. O módulo utiliza três eletrodos para captar os sinais biopotenciais do corpo.

As conexões dos eletrodos são identificadas da seguinte forma: RA (*Right Arm*): Eletrodo para o braço direito. LA (*Left Arm*): Eletrodo para o braço esquerdo. RL (*Right Leg*): Eletrodo para a perna direita, que atua como referência.

Para facilitar a correta aplicação, os cabos que conectam os eletrodos ao módulo são codificados por cores. O padrão original da SparkFun é: Preto: RA. Azul: LA. Vermelho: RL.

Porém, devido à disponibilidade de modelos no mercado brasileiro, é comum encontrar um padrão de cores diferente, conforme ilustrado na Figura 5: Vermelho: RA; Amarelo: LA; e Verde: RL.

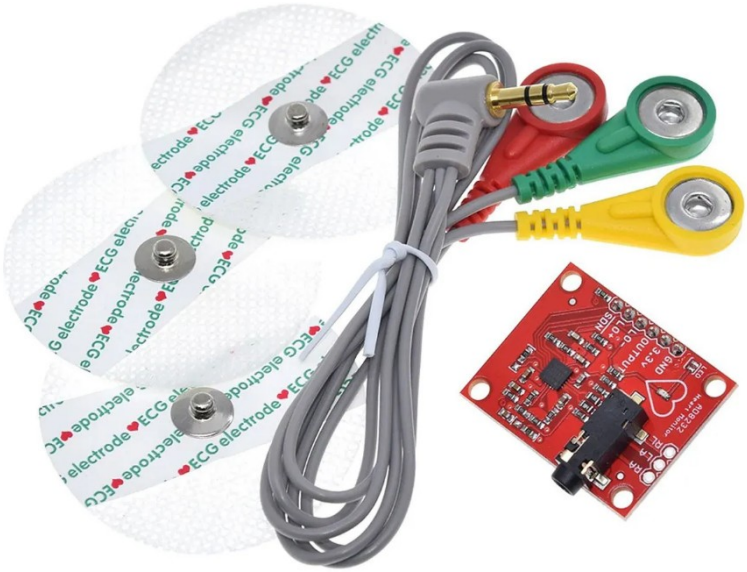

Figura 5 – Cabos e placa AD8232 com eletrodos.
Fonte: Casa da Robótica (2021)

O AD8232 possui diversas conexões específicas para seu pleno funcionamento, descritas a seguir: 3.3V e GND: Pinos para a alimentação do sensor, sendo GND a referência de terra (*Ground*).

OUTPUT: Saída do sinal analógico da atividade cardíaca, já amplificado, para ser lido pelo microcontrolador.

LO+ e LO-: Pinos da função de detecção de eletrodo desconectado (*Leads-Off Detection*). Essa função verifica a integridade da conexão, sendo que o pino LO+ está associado ao eletrodo positivo (LA) e o LO- ao negativo (RA).

SDN (*Shutdown*): Pino que permite desativar o sensor via *software* para economizar energia, útil em situações onde não é possível cortar a alimentação diretamente.

RA, LA e RL (Pinos): Conexões alternativas que permitem ligar os eletrodos diretamente na placa, sem a necessidade de usar o conector P2 de 3,5 mm.

Adicionalmente, o módulo possui um LED que pulsa em sincronia com os batimentos cardíacos, fornecendo uma indicação visual imediata do seu funcionamento. O esquema completo dos pinos é apresentado na Figura 6 (SPARKFUN, 2013).

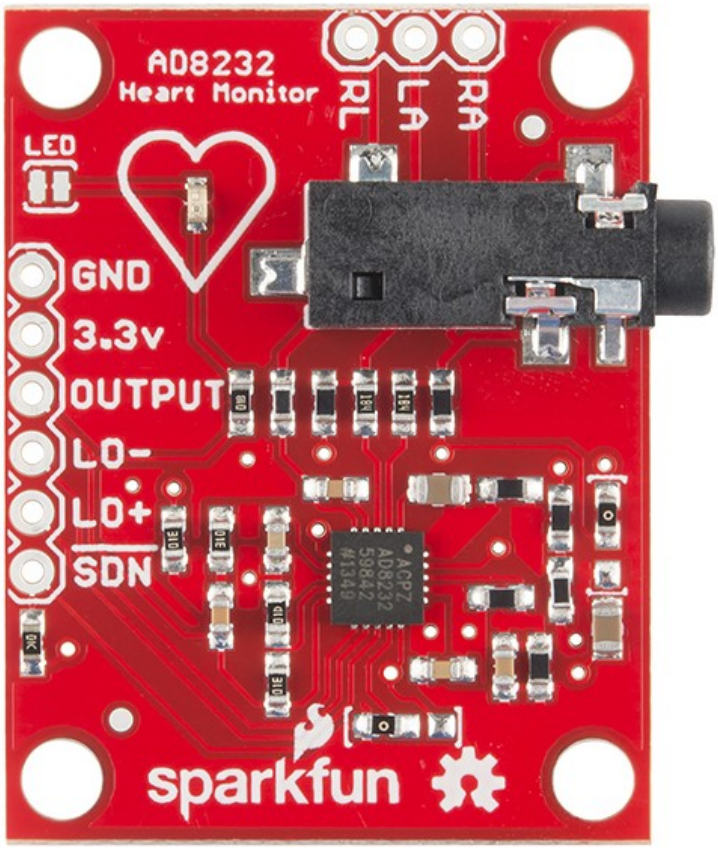

Figura 6 – Placa AD8232 e sua composição eletrônica.
Fonte: SparkFun (2013)

Para obter a melhor qualidade de sinal, os eletrodos devem ser posicionados o mais próximo possível do coração. Existem duas configurações principais recomendadas: Nos membros: Os eletrodos RA e LA são posicionados nos antebraços (próximo aos punhos), e o eletrodo RL é posicionado na perna direita (próximo ao joelho). No torso: Os eletrodos RA e LA são posicionados no peito, próximos aos braços (abaixo da clavícula), e o eletrodo RL é posicionado no abdômen inferior direito (acima do quadril). As principais configurações de posicionamento estão ilustradas na Figura 7.

É fundamental garantir uma boa conexão dos eletrodos com a pele. Para isso, recomenda-se limpar a área de aplicação para remover oleosidade e pelos, bem como utilizar eletrodos novos a cada medição para assegurar a melhor condução do sinal.

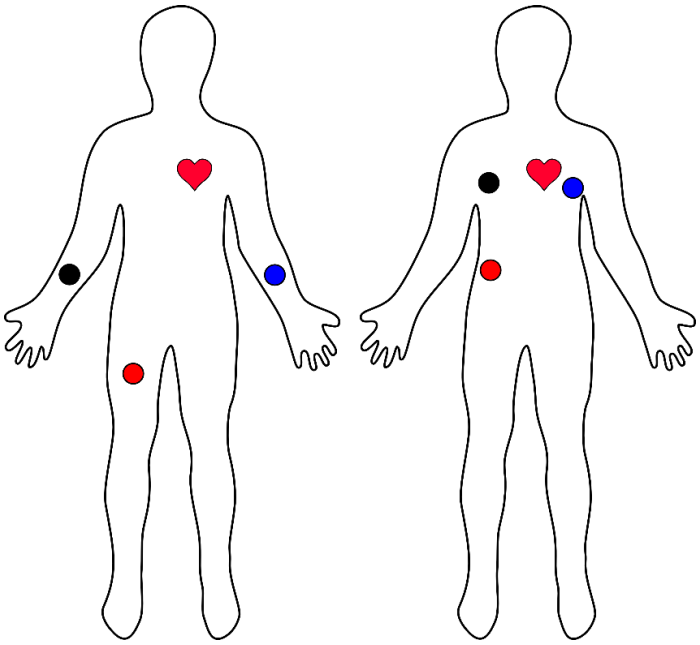

Figura 7 – Esquemas de Conexão dos eletrodos.
Fonte: SparkFun (2013)

3.1.2.2 MAX30102

O MAX30102 opera de forma distinta de um ECG, utilizando a fotopletismografia (PPG) para medir os sinais vitais. Conforme descrito no *datasheet* do componente (ANALOG DEVICES, 2018), este método consiste em emitir luz de LEDs integrados sobre um conjunto de vasos sanguíneos, como a ponta de um dedo. Essa luz interage com o fluxo sanguíneo e é captada por um fotossensor, o qual converte a intensidade luminosa recebida em um sinal elétrico, normalmente representado na forma de uma onda.

Conforme mencionado anteriormente, a função principal da PPG é monitorar a frequência cardíaca (BPM) e a saturação de oxigênio no sangue (SpO2). O módulo possui dois LEDs de alta precisão com comprimentos de onda distintos: um infravermelho (IR) e um vermelho. O princípio de medição da SpO2 baseia-se na interação dessas duas fontes luminosas com o sangue; o sangue rico em oxigênio absorve mais luz infravermelha e permite que mais luz vermelha passe, enquanto o sangue com pouco oxigênio faz o oposto. A Figura 8 apresenta o sensor MAX30102 (ANALOG DEVICES, 2018).

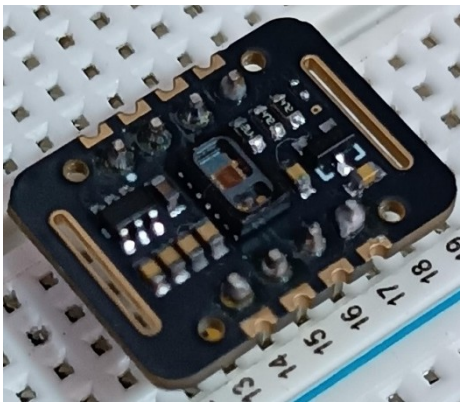

Figura 8 – Módulo MAX30102.
Fonte: O autor (2025)

A pinagem do sensor MAX30102 é relativamente simples, possuindo um total de oito pinos: VIN, GND, SCL, SDA, INT, IRD, RD e um GND adicional, conforme a configuração ilustrada nas Figuras 9 e 10.

No entanto, para a operação essencial do módulo, utilizam-se apenas quatro desses pinos. Os pinos VIN e GND são dedicados à alimentação, e o módulo aceita tensões de 3,3 V e 5 V, pois possui reguladores internos que fornecem 1,8 V para o sensor e 3,3 V para os LEDs. A comunicação é realizada pelos pinos SCL (*Serial Clock*), responsável pela sincronização, e SDA (*Serial Data*), utilizado para a transferência de dados. Ambos operam com base no protocolo I2C.

Os demais pinos oferecem funcionalidades adicionais. O pino INT (*Interrupt*) é uma saída opcional que alerta o microcontrolador quando novos dados estão prontos para serem lidos. Os pinos RD e IRD são saídas que permitem controlar LEDs externos para indicar o funcionamento dos emissores de luz vermelho e infravermelho, respectivamente. Por fim, o módulo também possui um pino GND adicional (ANALOG DEVICES, 2018).

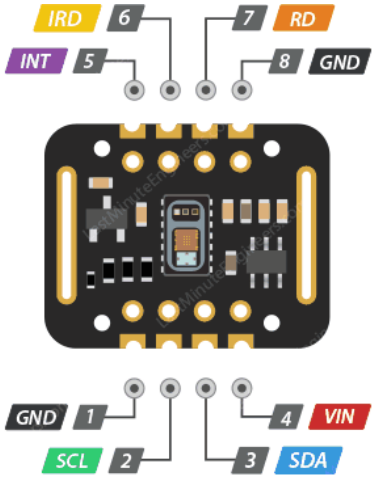

Figura 9- Pinagem do MAX30102 com parte frontal.
Fonte: Last Minute Engineers (2022)

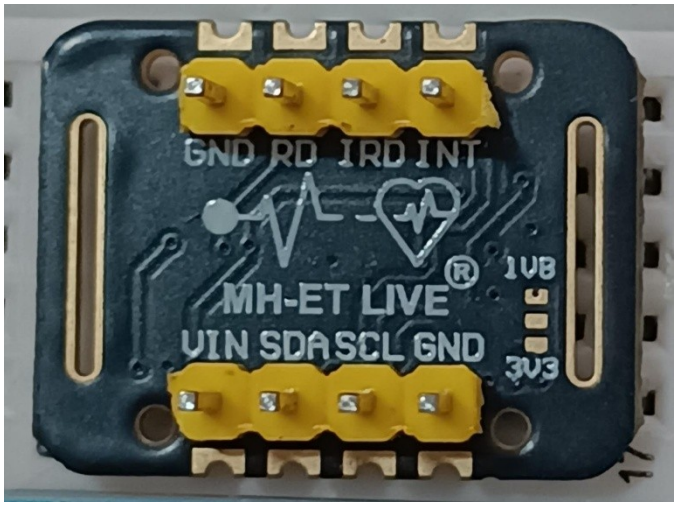

Figura 10 – Pinagem MAX30102 parte traseira.
Fonte: O autor (2025)

3.1.2.3 HC-05

Por fim, o quarto componente eletrônico central do projeto é o módulo *Bluetooth* HC-05, ilustrado na Figura 11. Este módulo pode operar de duas formas principais: como Mestre (*Master*) ou como Escravo (*Slave*).

No modo Mestre, o dispositivo é capaz de enviar comandos e informações para controlar outros periféricos. No modo Escravo, por outro lado, ele é configurado para receber ordens e dados de um dispositivo mestre. Para este projeto, o módulo foi utilizado em seu modo Escravo, sendo esta a sua função principal no sistema (ITEAD STUDIOS, 2010).

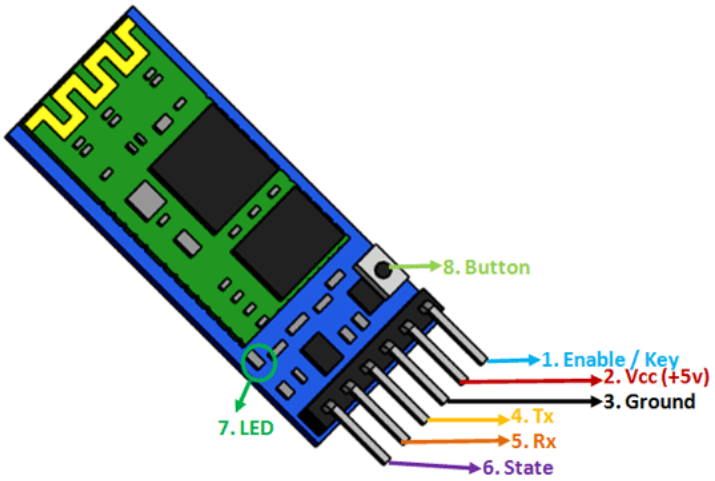

Figura 11 – Placa bluetooth com HC-05 integrado.
Fonte: Components101 (2021)

O módulo HC-05 é alimentado com uma tensão de +5V e GND. Embora seu chip principal opere com 3,3 V, a placa do módulo inclui reguladores de tensão internos para fornecer a alimentação correta tanto ao chip quanto a outros componentes, como o LED indicador (Figura 12).

Os pinos principais para a comunicação de dados são o RX e o TX. O pino RX (*Reception*) é o responsável por receber dados e comandos do microcontrolador via comunicação serial. Por sua vez, o pino TX (*Transmission*) realiza a função oposta, transmitindo dados do módulo para o microcontrolador.

Além desses, o HC-05 possui pinos com funções específicas. O pino EN (*Enable*) é utilizado para alternar o módulo entre seu modo de operação padrão (modo de dados) e o modo de comandos AT. Neste último, é possível modificar configurações internas do módulo, como nome, senha e taxa de comunicação (*Baud Rate*). Finalmente, o pino *STATE* serve para indicar o status da conexão *Bluetooth*, informando se o módulo está pareado ou aguardando conexão (ITEAD STUDIOS, 2010).

Figura 12 – Módulo bluetooth limpo sem placas de conexão, LED ou botão.
Fonte: Itead Studio (2010)

3.1.2.4 Materiais Gerais

Além desses quatro componentes principais, também são necessários outros materiais de uso geral para o pleno funcionamento do projeto, tais como *protoboard* e fios *jumpers* apresentados na Figura 13, resistores (caso necessário) e uma fonte de alimentação para o Arduino.

(a)

(b)

Figura 13 – (a) Fios jumper e (b) mini protoboard.
Fonte: MakerHero (2025)

Lista de materiais utilizados:

- Arduino Mega 2560
- Protoboard Mini
- AD8232
- MAX30102

- 15/20 Jumpers Macho-Macho
- HC-05
- Resistores (Opcional)
- Fonte de Alimentação (Notebook)

3.2 Montagem do Protótipo

Com os materiais definidos, iniciou-se a montagem do protótipo para os testes. Nesta etapa, foram estabelecidas as conexões elétricas ao microcontrolador, tanto as de alimentação quanto as de controle. A *protoboard* e a placa Arduino MEGA 2560 serviram como base para a montagem, e os sensores foram dispostos sobre a *protoboard* de modo a facilitar as conexões. Embora o Arduino MEGA 2560 possuísse pinos suficientes para uma ligação direta, a indisponibilidade de *jumpers* do tipo macho-fêmea tornou necessária a utilização da *protoboard*, uma vez que se dispunha apenas de *jumpers* macho-macho.

Para a montagem, as conexões entre os sensores e o Arduino Mega foram estabelecidas por meio de *jumpers*. A alimentação dos módulos foi a primeira etapa: o pino VIN do sensor MAX30102 foi conectado à saída de 5V do Arduino, enquanto o AD8232 foi alimentado pela saída de 3,3V, respeitando a tensão de operação de cada componente. Como tanto o MAX30102 quanto o HC-05 operam com 5V, criou-se uma linha de alimentação comum na *protoboard* para atendê-los. De forma similar, todos os módulos foram conectados a uma linha de aterramento comum, ligada ao pino GND do Arduino.

Em seguida, foram feitas as conexões de dados. Para o sensor MAX30102, que opera via comunicação I2C, os pinos SCL e SDA foram ligados às suas respectivas portas no microcontrolador. O pino *OUTPUT* do módulo AD8232, por sua vez, foi conectado à porta analógica A0, e os pinos LO- e LO+ foram conectados às portas digitais 22 e 24, respectivamente.

Por fim, realizou-se a conexão do módulo *Bluetooth* HC-05. A comunicação serial entre o módulo e o microcontrolador exige uma conexão cruzada: o pino de transmissão (TX) de um dispositivo deve ser ligado ao pino de recepção (RX) do outro, e vice-versa. Seguindo esse princípio, o pino TX do HC-05 foi conectado ao pino RX1 do Arduino, e o pino RX do módulo foi conectado ao pino TX1 do microcontrolador. Essa configuração garante que os dados enviados por um dispositivo sejam corretamente recebidos pelo outro.

Com estas conexões estabelecidas, a disposição final do circuito é a que está ilustrada na Figura 14.

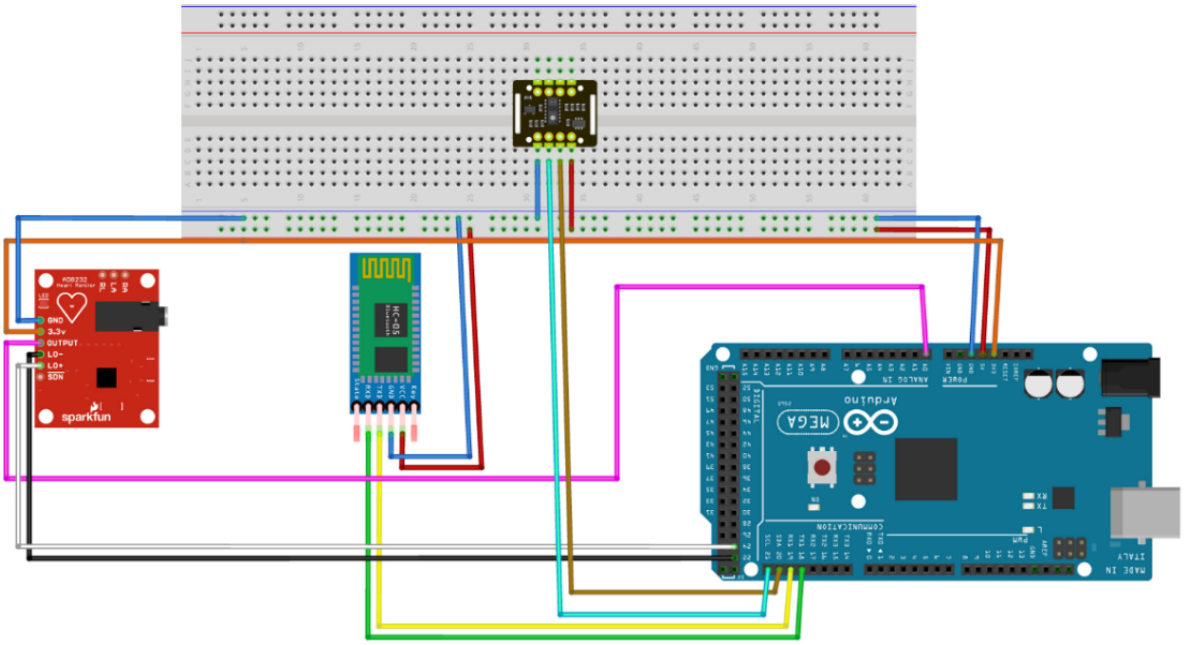

Figura 14 – Esquema do Circuito feito no Fritzing.
Fonte: O autor (2025)

3.3 Código-fonte

O código-fonte do projeto é o detentor real de todas as capacidades realizadas pelo monitor cardíaco. Os sensores, como já explicado, possuem muitos equipamentos embutidos para melhoria de sinal, tanto em recepção como em transmissão de dados, mas são basicamente inúteis sem um guia. O guia desses sensores é o microcontrolador, que, por sua vez, precisa receber comandos para iniciar as comunicações. A ideia central do *software* é permitir que os dois módulos utilizados, AD8232 e MAX30102, enviem informações como onda ECG, BPM e SpO2, garantindo que um sensor não atrapalhe o outro durante as leituras e evitando interferências devido à alta sensibilidade dos componentes. O código-fonte completo do projeto pode ser consultado no Apêndice A.

O *software* foi desenvolvido na IDE do Arduino, que se estrutura em duas funções principais: *Setup*, que executa os comandos de inicialização uma única vez, e *Loop*, que repete indefinidamente os comandos. Para o pleno funcionamento do código, foram implementadas as seguintes etapas: inclusão de bibliotecas, definição de variáveis globais, ativação de sensores e da comunicação serial, configuração dos parâmetros dos sensores, implementação de um sistema de alternância de modos, e pôr fim a coleta, processamento e envio de dados via *Bluetooth*.

Inicialmente, no início do código-fonte, apresentado no Apêndice A, demonstra as bibliotecas utilizadas. Para este projeto, apenas o MAX30102 requer uma biblioteca específica,

pois sua comunicação I2C pelos pinos SCL e SDA é complexa, e a biblioteca facilita a interação sem a necessidade de escrever extensas linhas de código para cada função básica. A biblioteca utilizada, "MAX30105.h" da SparkFun, embora nomeada para outro modelo, é compatível com o MAX30102. A comunicação com o sensor é então encapsulada pela instância particleSensor do objeto MAX30105. Já o AD8232 não necessita de biblioteca, pois seu pino de comunicação *OUTPUT* envia valores analógicos brutos diretamente ao Arduino, simplificando a programação. Adicionalmente, a biblioteca <math.h> foi incluída para acesso a funções matemáticas prontas.

Após as bibliotecas, o código declara um conjunto de variáveis globais (conforme visto no Apêndice A) que servem como base para o funcionamento do sistema. São criadas diversas variáveis constantes (const int) para definir os pinos dos sensores e os limites fisiológicos de BPM e SpO_2, evitando que dados fora da realidade sejam processados. Também são definidos buffers de histórico para suavizar as leituras de BPM e SpO_2, além de variáveis para armazenar os dados intermediários do cálculo de SpO_2. Além disso, é definida a estrutura enum SensorMode para gerenciar a alternância entre os modos de operação, um pilar fundamental da arquitetura do software. Por fim, são declaradas as variáveis de estado para monitorar a conexão dos sensores, controlar o tempo das leituras e armazenar os dados brutos do sinal.

A próxima etapa executada é a função setup(), onde são feitas definições importantes. Nela, inicia-se a comunicação serial tanto com a IDE quanto com o módulo *Bluetooth*, configuram-se os pinos dos sensores, e realiza-se a verificação e ativação do MAX30102. Caso o sensor não seja encontrado, uma mensagem de aviso é exibida. Além dessas funções, também se definem os comandos de alternância dos dois sensores, sendo o modo ECG o número '1', e o modo BPM/SpO2 (Oxímetro) o número '2'. O modo padrão foi definido para ser o modo ECG, conforme pode ser verificado na função setup() no Apêndice A.

Sendo uma função tão importante, a loop() serve para manter os módulos atualizados, permitindo que o Arduino saiba qual sensor ler a cada momento. Essa rotina se inicia com a verificação de comandos de alternância via *Bluetooth*. Caso uma troca de modo seja solicitada, o tipo de leitura é alterado entre a onda ECG e os dados de BPM e SpO_2. As leituras são atualizadas e exibidas a cada segundo, valor definido na variável PRINT_INTERVAL de 1000 ms, cuja implementação pode ser vista na função loop() no Apêndice A. A função loop() apenas delega as tarefas de leitura e exibição, não contendo os cálculos em si, o que ajuda a evitar erros de repetição e mantém o código organizado.

Um ponto crucial do código é a alternância de modos, que constitui a base para o sistema operar em harmonia. Para evitar que a sensibilidade do AD8232 a pulsos elétricos captasse ruídos do MAX30102, foi implementado um sistema de alternância em que apenas um sensor fica ativo por vez, enquanto o outro entra em estado de "hibernação" (shutDown()). Para essa alternância acontecer, utiliza-se a estrutura enum SensorMode, declarada no início do código (Apêndice A). A implementação dessa lógica é centralizada nas funções switchToEcgMode() e switchToOximeterMode(), detalhadas no Apêndice A. A primeira ativa o modo ECG e garante que o MAX30102 seja desativado. A segunda ativa o modo de oximetria e configura o MAX30102 com parâmetros otimizados, como intensidade do LED, número de amostras e taxa de amostragem, então ele inicia a comunicação Serial com o MAX30102 e corta a comunicação com o AD8232. Por fim, a função checkModeSwitch() é responsável por receber os comandos '1' e '2' via *Bluetooth*, sendo chamada continuamente pelo loop() para verificar a necessidade de alterar o modo atual do monitor.

Por último, tem-se a parte de recepção e processamento de dados. Para o AD8232, o processo é simples: a função readECG() lê o valor analógico bruto e o envia para a porta serial, cuja implementação simples pode ser vista na função readECG() no Apêndice A.

Já o MAX30102 necessita de funções mais elaboradas, pois seu objetivo é fornecer os valores processados de BPM e SpO_2. O processo, iniciado na função readOximeter() (Apêndice A), começa com a aquisição dos valores dos LEDs IR e vermelho. O sinal passa por um filtro passa-baixa, implementado no início da função processSignals() (Apêndice A), que suaviza a leitura somando 95% do valor atual com 5% do novo. Em seguida, uma verificação pico-vale é realizada para identificar o batimento cardíaco, validando um pico apenas quando o valor seguinte do sinal é menor, o que garante precisão e impede a detecção dupla.

Com um batimento válido detectado, a função processSignals() inicia os cálculos. O BPM é obtido a partir da Equação 1, medindo-se o intervalo de tempo entre os picos. Para evitar erros, o algoritmo exige que o intervalo seja maior que 250 ms, o que corresponde a um limite máximo de 240 BPM, descartando valores irreais que poderiam ser causados por ruído, conforme a lógica implementada na função processSignals() (Apêndice A).

Para o cálculo do SpO_2, aplica-se o método empírico da Equação 3. Inicialmente, o algoritmo coleta os componentes AC e DC de cada LED, calculando a amplitude do pulso (AC) pela diferença entre os valores máximo e mínimo, e o nível médio do sinal (DC) pela média desses mesmos pontos. Em seguida, calcula-se a Razão das Razões (Equação 2) para obter o valor de R, que é então aplicado na fórmula empírica para se obter a porcentagem de SpO_2. O

valor final é verificado para garantir que esteja dentro de uma faixa fisiologicamente possível e, então, é salvo em um buffer para suavização.

Ao fim dos cálculos, o código reseta os valores coletados dos LEDs para evitar inconsistências na leitura e inicia um novo ciclo de coleta, conforme visto ao final do bloco de cálculo de SpO2 na função processSignals() no Apêndice A. Os valores de BPM e SpO_2 são então filtrados por uma média móvel, implementada pela função calculateAverage() (Apêndice A), que ignora valores nulos para maior precisão.

Por fim, o código envia, através da função displayResults() (Apêndice A), *strings* com as informações pertinentes ao usuário. No modo oxímetro, o sistema aguarda um buffer de cinco leituras válidas para estabilizar a medição antes de exibir os resultados. No caso do ECG, os valores já são enviados com o endereçamento correto dentro da função readECG() (Apêndice A). Como recurso final, o sistema classifica o BPM (Bradicardia/Taquicardia) e verifica se os valores estão dentro da normalidade. Caso uma anormalidade seja detectada (BPM fora da faixa ou SpO_2 abaixo de 95%), uma mensagem de alerta é exibida e um indicador LED no aplicativo muda de verde para vermelho. Os comandos dos LEDs foram definidos como constantes no início do código (Apêndice A) e utilizam o sistema de endereçamento do aplicativo. Além dessas, uma função de reset, resetAll() (Apêndice A), também foi criada para limpar todos os valores durante a troca de modos, garantindo a estabilidade das leituras.

3.4 Bluetooth Electronics

Conforme apresentado anteriormente, o objetivo deste projeto é exibir os dados de frequência cardíaca e de saturação de oxigênio, tanto numericamente quanto em gráficos, por meio de uma conexão *Bluetooth*. Para tal, foi necessário selecionar um aplicativo que fosse de fácil acesso e manuseio, visto que a praticidade é um dos requisitos do sistema. Diante disso, optou-se pelo aplicativo *Bluetooth Electronics* como ferramenta principal. Adicionalmente, essa escolha eliminou a necessidade de desenvolver um *software* específico para o projeto.

O *Bluetooth Electronics* é um aplicativo gratuito desenvolvido pela Keuwlsoft com o intuito de fornecer uma plataforma intuitiva para a criação de sistemas de controle e coleta de dados via *wireless*. Ele se destaca por sua compatibilidade com uma vasta gama de microcontroladores (KEUWLSOFT, 2019).

O aplicativo opera com um sistema de painéis que contêm uma *grid*. Nessa interface, o usuário pode selecionar componentes e posicioná-los livremente por meio de uma função de arrastar e soltar (*drag and drop*), conforme ilustrado na Figura 15. Esses componentes dividem-

se em duas categorias: de exibição (como gráficos, campos de texto e terminais) e de controle (como botões, interruptores e barras deslizantes).

A comunicação entre os componentes do aplicativo e o microcontrolador é realizada por meio de um protocolo específico. Cada mensagem é formatada como uma *string* que se inicia com um *. Em seguida, um caractere único atua como endereço, identificando o componente de destino. Após o endereço, o valor a ser exibido é enviado, e a *string* é finalizada com um segundo asterisco. Esse método de endereçamento permite que o aplicativo filtre e direcione os dados recebidos via *Bluetooth* apenas para os componentes de interesse, otimizando a comunicação (KEUWLSOFT, 2019).

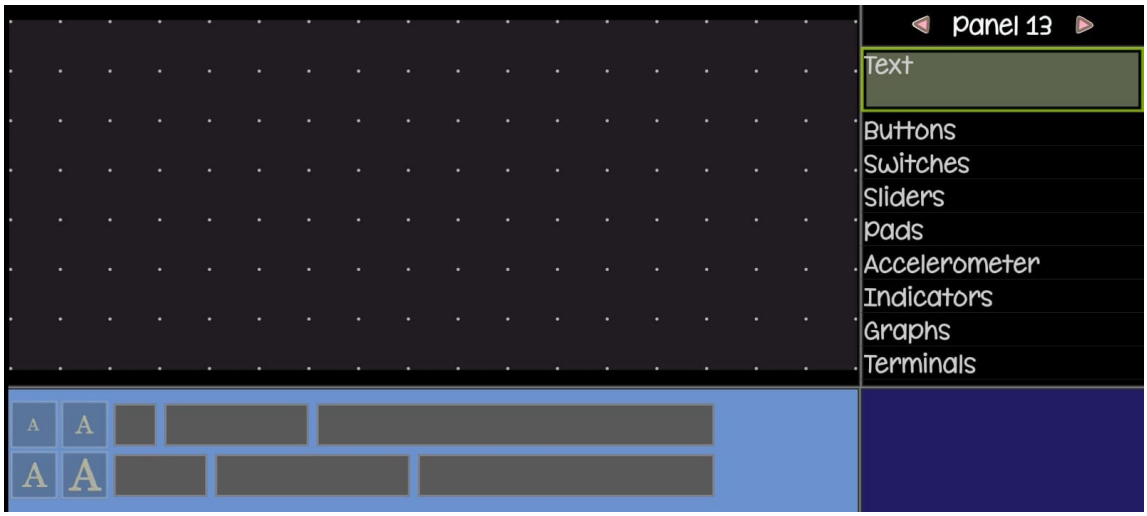

Figura 15 – Grid do Painel 13 do Bluetooth Electronics com a grade de componentes a direita.
Fonte: O autor (2025)

Para a interface de visualização de dados deste projeto, foram utilizados dois componentes do aplicativo: o *Terminal Monitor* na Figura 16 e o *Rolling Graph* na Figura 17. Cada um desses componentes opera de maneira distinta.

O *Terminal Monitor* funciona como um espelho da porta serial do microcontrolador, exibindo todos os dados brutos que são transmitidos por ela. Embora possa ser configurado para filtrar informações específicas, sua função principal neste projeto foi servir como uma ferramenta de *debug*, permitindo verificar a integridade e o formato dos dados enviados pelo Arduino.

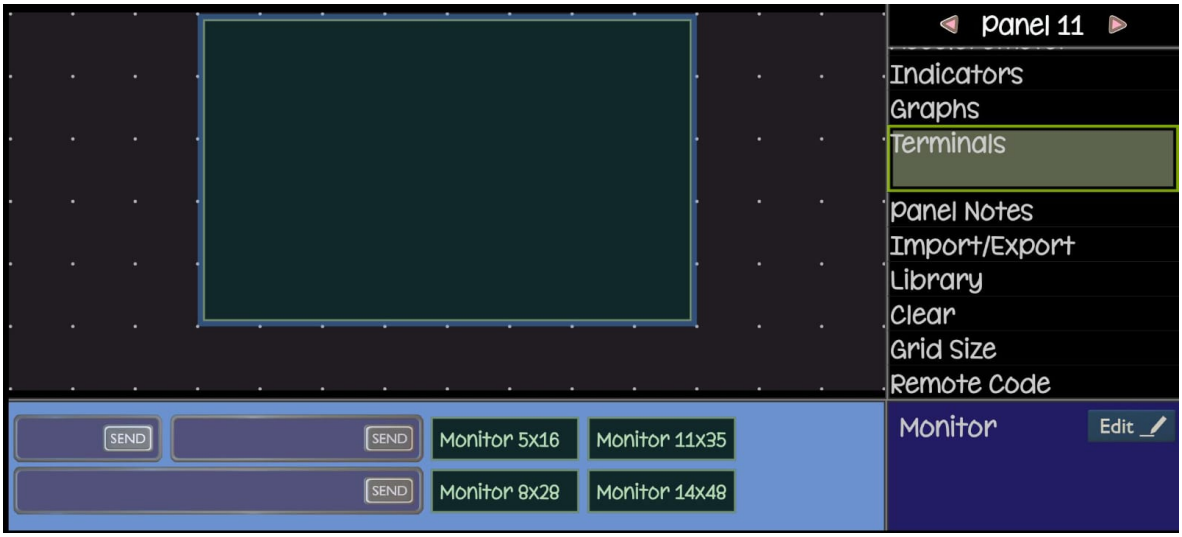

Figura 16 – Terminal Monitor 14x48
Fonte: O autor (2025)

O componente *Rolling Graph*, ao contrário do *Terminal Monitor*, exige a utilização do protocolo de endereçamento para operar. Isso é necessário para que o aplicativo possa identificar, dentre os dados recebidos via *Bluetooth*, quais devem ser especificamente representados no gráfico.

No entanto, na implementação deste projeto, observou-se uma particularidade no envio dos dados. O uso do comando println no código do Arduino, que automaticamente adiciona um caractere de nova linha ao final de cada transmissão, foi suficiente para que o aplicativo processasse o dado e avançasse para o próximo ponto no gráfico. Essa abordagem tornou desnecessário o envio do caractere de * para finalizar cada envio de dados, simplificando a comunicação.

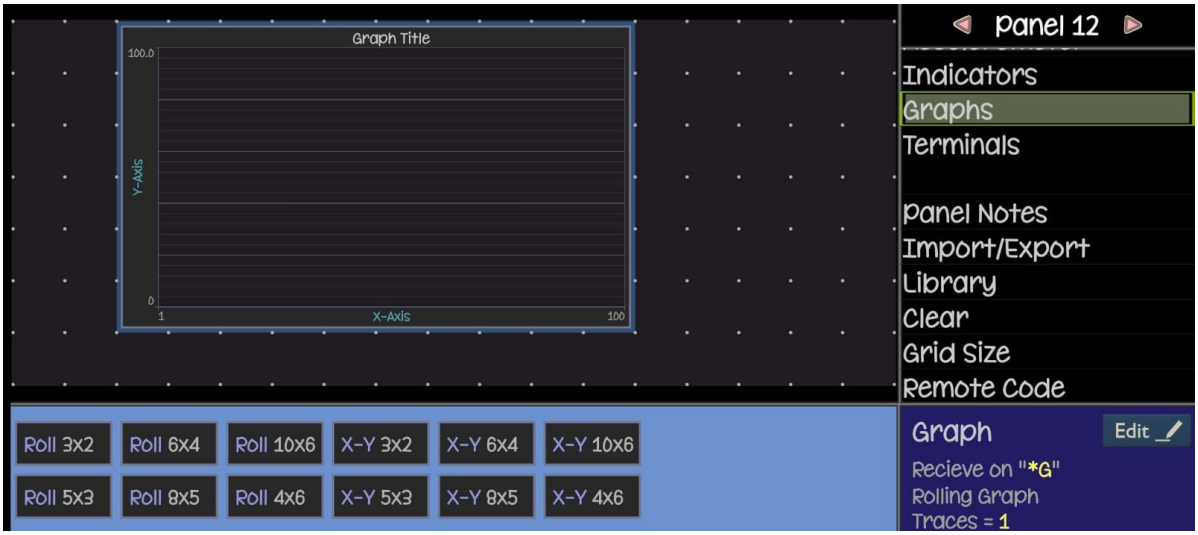

Figura 17 – Gráfico tipo Roll 10x6
Fonte: O autor (2025)

Além da configuração básica para a recepção de dados, o componente de gráfico oferece diversas opções de personalização. Entre elas, destacam-se: Configuração dos eixos: É possível definir os valores máximo e mínimo do eixo Y, bem como nomear os eixos X e Y e o título do gráfico; Aparência da linha: O usuário pode customizar a cor e a espessura da linha representada, além de definir a quantidade de pontos a serem exibidos simultaneamente; Funcionalidades adicionais: O componente inclui também uma opção de autoajuste da escala, que adapta dinamicamente o eixo Y aos valores recebidos (KEUWLSOFT, 2019).

4. Resultados

A Figura 18 exibe o sinal de onda PPG capturado pelo sensor MAX30102. No gráfico, a linha azul representa o sinal do LED infravermelho (IR), enquanto a linha vermelha representa o sinal do LED vermelho (*Red*).

Ambos os sinais ilustram o ciclo dos batimentos cardíacos, exibindo o pico duplo característico de cada ciclo. Nota-se que o sinal infravermelho apresenta maior amplitude, característica que pode ser atribuída ao seu maior comprimento de onda, o que geralmente resulta em uma maior profundidade de penetração nos tecidos.

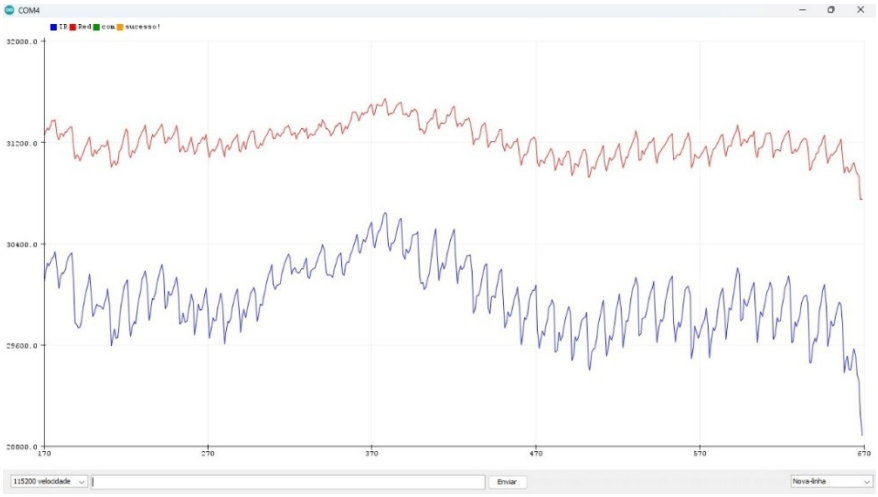

Figura 18 – Onda de batimentos PPG do MAX30102 via Monitor Serial.
Fonte: O autor (2025)

As Figuras 19 e 20 apresentam os valores de frequência cardíaca (BPM) e de saturação de oxigênio no sangue (SpO2). Os dados foram coletados em um único indivíduo em estado de repouso. As imagens exibem os resultados dos cálculos realizados pelo software, conforme são transmitidos via comunicação serial pelo Arduino e pelo módulo *Bluetooth*. Adicionalmente, o LED verde aceso sinaliza o funcionamento estável das leituras, indicando a ausência de anomalias.

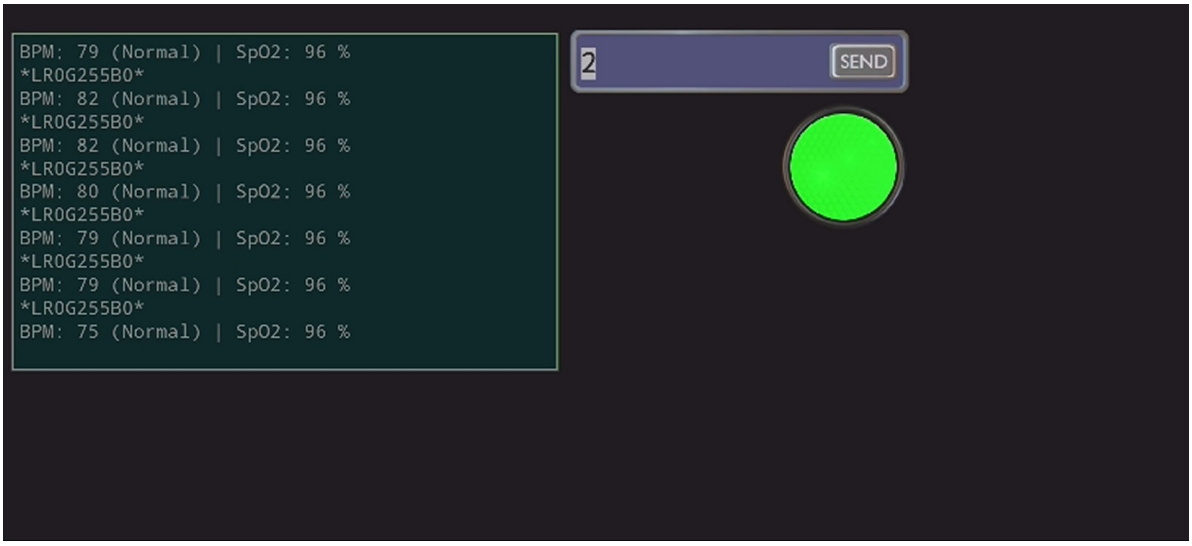

Figura 19 – Monitor Serial do Bluetooth Electronics mostrando BPM e SpO2 do MAX30102 com LED verde para valores corretos.
Fonte: O autor (2025)

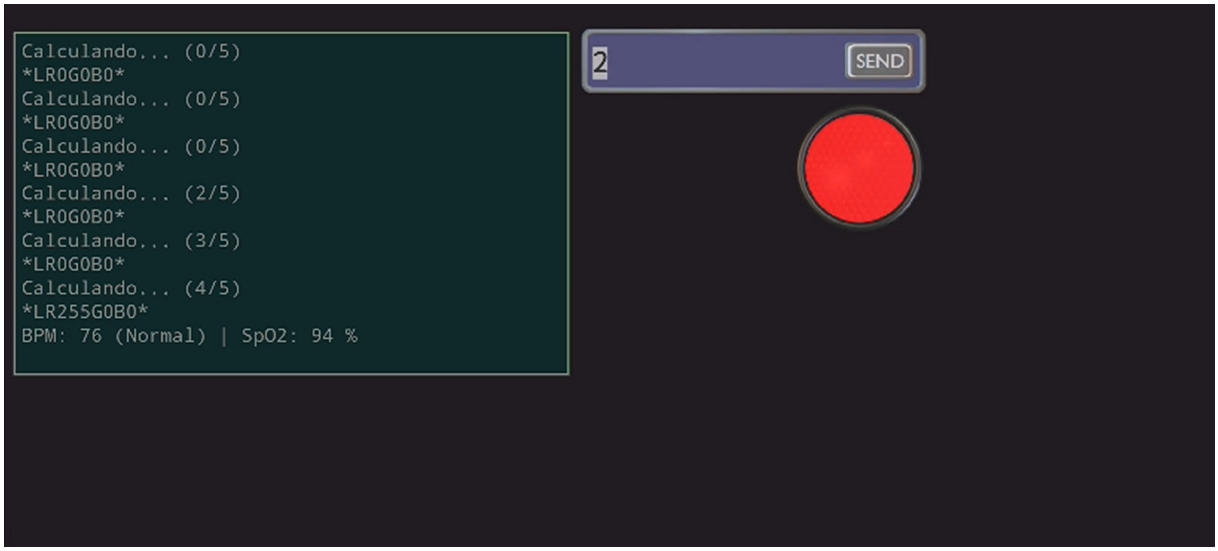

Figura 20 – LED Vermelho ativado devido ao SpO2 estar menor que 94%.
Fonte: O autor (2025)

Conforme ilustrado na Figura 20, o LED de status altera sua cor para vermelho quando os parâmetros vitais se encontram fora da faixa de normalidade (BPM < 60 ou > 100; SpO2 < 95%). Essa alteração funciona como um alerta visual para indicar uma potencial anomalia nos dados coletados.

Na Figura 21, é apresentado o sinal de ECG capturado pelo módulo AD8232 e transmitido para a interface gráfica via *Bluetooth*. O gráfico resultante demonstra um sinal íntegro e completo, no qual são visíveis todos os elementos fundamentais para uma análise morfológica, incluindo a onda P, o complexo QRS e a onda T.

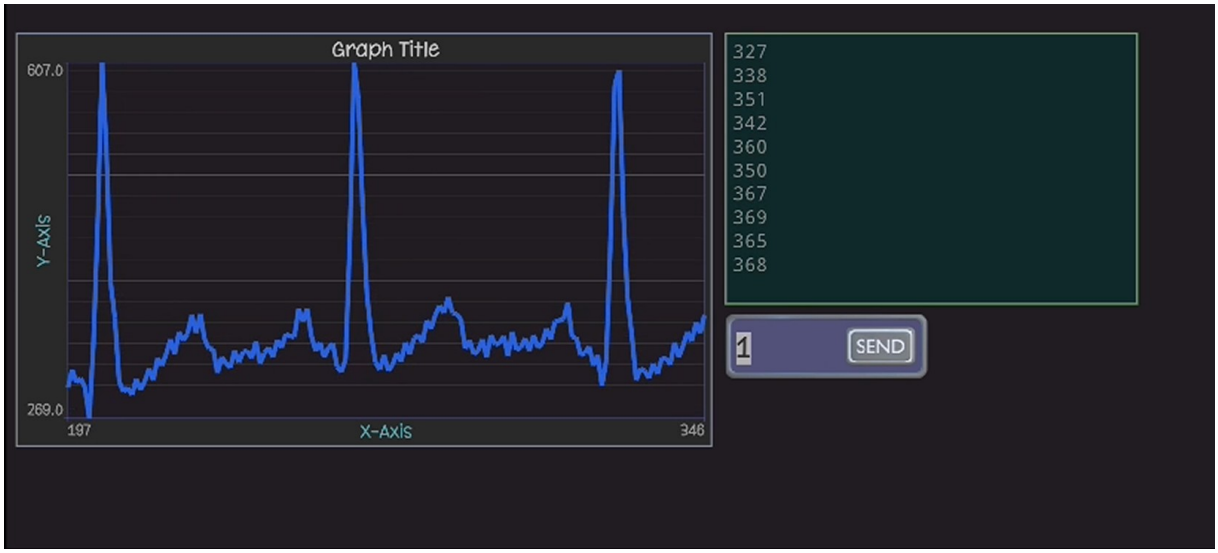

Figura 21 – Gráfico ECG do AD8232.
Fonte: O autor (2025)

Houve alguns desafios ao longo do desenvolvimento que dificultaram sua conclusão e eficiência. O primeiro e mais evidente desafio foi o ruído no sinal de eletrocardiograma (ECG) captado pelo sensor AD8232. Conforme observado na Figura 21, o sinal apresentava uma linha de base instável em vez de reta. Isso ocorre devido à alta sensibilidade do sensor a interferências eletromagnéticas, incluindo a da rede elétrica, que possui uma frequência de 60 Hz. Para corrigir esse erro, filtros foram adicionados no código-fonte para mitigar essa interferência, o que funcionou. Porém, o filtro acabou interferindo com as leituras de onda ECG, gerando uma onda inconsistente. Como a onda plotada no gráfico é visível e precisa, o ruído pode ser desprezado.

No caso do MAX, por ser um sensor que depende de luminosidade, fontes externas de luz podem interferir na leitura. Isso pode levar a leituras falsas; por exemplo, os valores podem iniciar corretamente, mas mudarem bruscamente para valores muito altos de BPM ou muito baixos. Mesmo com muitos filtros presentes no código-fonte para mitigar erros, fatores externos ainda são influentes nas leituras, causando erros.

Outro ponto a ser considerado, que pode se tornar uma limitação, é a capacidade de processamento do microcontrolador. Este tipo de dispositivo não é projetado para cálculos computacionalmente intensivos, e a base de todo o projeto foi a coleta e o processamento de dados. Para o Arduino MEGA utilizado, que possui recursos robustos (8 KB de SRAM e 256 KB de Flash), isso não representou um problema. Contudo, para o Arduino Nano, sugerido para melhorar a portabilidade do monitor, é provável que a menor capacidade de memória se torne uma limitação de hardware. Tal restrição não poderia ser mitigada por outros métodos, exigindo

o uso de uma plataforma mais potente. Essa limitação também impede que o código possua filtros e algoritmos mais complexos, pois a carga de processamento excederia os recursos do microcontrolador, e enviar leituras ao mesmo tempo ao aplicativo, necessitando do modo de alternância.

5. CONSIDERAÇÕES FINAIS

Este trabalho foi iniciado com o objetivo de desenvolver um dispositivo de monitoramento cardíaco de baixo custo através do Arduino, com interface e controle pelo *Bluetooth*. Conclui-se que este objetivo foi atingido, já que o dispositivo foi desenvolvido com materiais de baixo custo e uso livre, obtendo resultados satisfatórios e cumprindo sua função como monitor cardíaco.

Apesar dos resultados funcionais e satisfatórios, o projeto não está isento de limitações, como os ruídos gerados por fontes externas que podem interferir na precisão dos sensores e as limitações do próprio microcontrolador. Devido a isso, o projeto pode receber melhorias futuras, tais como: Bloqueio de interferências externas, tanto para pulsos elétricos como para luminosidade; Desenvolvimento de uma interface própria para o sistema, que não dependa de um aplicativo de terceiros; Adição de conexão via Wi-Fi, com um possível site para monitoramento; Melhoria dos materiais utilizados, sempre respeitando as margens de baixo custo. Um exemplo prático da melhoria de materiais seria utilizar o microcontrolador ESP32. Ele possui mais memória e poder de processamento em comparação com os modelos de Arduino, além de já ter módulos *Bluetooth* e Wi-Fi integrados, facilitando a aplicabilidade do sistema nas duas modalidades. A compatibilidade do ESP32 com a IDE do Arduino o torna uma “evolução natural” para este dispositivo. Também pode-se integrar uma placa PCB para mitigar mal contato que é gerado pela alta quantidade de *jumpers* conectados a *protoboard*.

Por fim, vale ressaltar que a função principal deste monitor cardíaco é ser um complemento para equipamentos médicos, e não um substituto.

REFERÊNCIAS

ALIAN, Aymen A.; SHELLEY, Kirk H. Photoplethysmography. **Best Practice & Research Clinical Anaesthesiology**, v. 28, n. 4, p. 395-406, 2014. DOI: 10.1016/j.bpa.2014.08.006.

ALLEN, John. Photoplethysmography and its application in clinical physiological measurement. **Physiological Measurement**, v. 28, n. 3, p. R1-R39, 2007. DOI: 10.1088/0967-3334/28/3/R01

ANALOG DEVICES. **MAX30102: High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health**. Norwood, MA, 2018. Disponível em: https://www.analog.com/media/en/technical-documentation/data-sheets/max30102.pdf. Acesso em: 25 jun. 2025.

ARDUINO. **Hardware.** 2025. Disponível em: https://www.arduino.cc/en/hardware. Acesso em: 7 ago. 2025.

BERALDO, O. A. **Processamento Digital do Sinal de Eletrocardiograma para Aplicação em Experimentos de Fisiologia Cardíaca**. 1997. Dissertação (Mestrado em Engenharia Elétrica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 1997. Disponível em: https://www.teses.usp.br/teses/disponiveis/18/18133/tde-27112017-142134/publico/Dissert_Beraldo_OswaldoA.pdf. Acesso em: 23 jun. 2025.

CASA DA ROBÓTICA. **Módulo Pulso Frequência Cardíaca ECG AD8232**. 2021. Disponível em: https://www.casadarobotica.com/sensores-modulos/modulos/outros/modulo-pulso-frequencia-cardiaca-ecg-ad8232. Acesso em: 25 jun. 2025.

CASTANEDA, Denisse *et al*. A review on wearable photoplethysmography sensors and their potential future applications in health care. **International Journal of Biosensors & Bioelectronics**, v. 4, n. 4, p. 195-202, 2018. DOI: 10.15406/ijbsbe.2018.04.00125.

COMPONENTS101. **HC-05 Bluetooth Module Pinout, Configuration, Interfacing**. Components101, 16 jul. 2021. Disponível em: https://components101.com/wireless/hc-05-bluetooth-module. Acesso em: 29 jun. 2025.

FERRONI, Eduardo *et al*. **A PLATAFORMA ARDUÍNO E SUAS APLICAÇÕES**. Revista Da UI_IPSantarém, v. 3, n. 2, p. 133-148, 2015. DOI: https://doi.org/10.25746/ruiips.v3.i2.14354. Acesso em: 15 jun. 2025.

GUIMARÃES, J. I. *et al*. Normatização dos equipamentos e técnicas para a realização de exames de eletrocardiografia e eletrocardiografia de alta resolução. **Arquivos Brasileiros de Cardiologia**, v. 73, n. 4, p. 327-336, out. 1999. Disponível em: https://www.scielo.br/j/abc/a/Srtw3DdFHyRCPK7JfQbWx6f/?lang=pt. Acesso em: 23 jun. 2025.

GUYTON, Arthur C ; HALL, John E. **Tratado de Fisiologia Médica**. 13. ed. Rio de Janeiro: Elsevier, 2011. Disponível em:

https://cssjd.org.br/imagens/editor/files/2019/Abril/Tratado%20de%20Fisiologia%20Médica.pdf. Acesso em: 16 jul. 2025.

ITEAD STUDIO. **HC-05 - Bluetooth to Serial Port Module**. 18 jun. 2010. Disponível em: https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf. Acesso em: 29 jun. 2025.

KEUWLSOFT. **Bluetooth Electronics: User Guide**. [S. l.]: Keuwlsoft, **2019**. Disponível em: https://www.keuwl.com/apps/bluetoothelectronics/userguide/. Acesso em: 1 jul. 2025.

LAST MINUTE ENGINEERS. **MAX30102 Pulse Oximeter & Heart Rate Sensor with Arduino**. [S. l.], 2022. Disponível em: https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/. Acesso em: 25 jun. 2025.

MAKERHERO. **MakerHero - Componentes Eletrônicos e Impressão 3D**. 2025. Disponível em: https://www.makerhero.com. Acesso em: 8 ago. 2025.

OAK, Sang-Soo; AROUL, Praveen. **How to Design Peripheral Oxygen Saturation (SpO_2) and Optical Heart Rate Monitoring (OHRM) Systems Using the AFE4403**. Dallas: Texas Instruments, mar. 2015. (Application Report; SLAA655). Disponível em: https://www.ti.com/lit/an/slaa655/slaa655.pdf. Acesso em: 1 jul. 2025.

RIOS, Jefferson *et al*. **Introdução ao Arduino**. Fundação Universidade Federal de Mato Grosso do Sul Campo Grande - UFMS, 2012. https://www.academia.edu/5229813/Introdução_ao_Arduino?auto=download. Acesso em: 15 jun. 2025.

SOUZA, Fábio. **Arduino Mega 2560**. Embarcados, 28 abr. 2014. Disponível em: https://embarcados.com.br/arduino-mega-2560/. Acesso em: 2 jul. 2025.

SPARKFUN. **AD8232 Heart Rate Monitor Hookup Guide**. SparkFun Learn, 2013. Disponível em: https://learn.sparkfun.com/tutorials/ad8232-heart-rate-monitor-hookup-guide. Acesso em: 25 jun. 2025.

SPARKFUN. **MAX30105 Particle and Pulse Ox Sensor Hookup Guide**. SparkFun, 1 dez. 2016. Disponível em: https://learn.sparkfun.com/tutorials/max30105-particle-and-pulse-ox-sensor-hookup-guide/using-the-sparkfun-max30105-arduino-library. Acesso em: 29 jun. 2025.

TAMURA, Toshiyo *et al*. Wearable Photoplethysmographic Sensors—Past and Present. **Electronics**, v. 3, n. 2, p. 282-302, 2014. DOI: 10.3390/electronics3020282.

APÊNDICE A – CÓDIGO-FONTE COMPLETO DO PROJETO

```
#include "MAX30105.h"
#include <math.h>

// Definição do modo de depuração. Mude para 'true' para ver valores brutos no Monitor Serial.
#define DEBUG_MODE false

// Instância do objeto para o sensor MAX30102
MAX30105 particleSensor;

// Enum para controlar os modos de operação de forma legível
enum SensorMode {
  MODE_ECG,
  MODE_BPM_SPO2
};
SensorMode currentMode = MODE_ECG; // O modo inicial é ECG

// ================== DEFINIÇÕES PARA OS COMANDOS DOS LEDS ==================
// Comandos de cor RGB para o aplicativo Bluetooth Electronics, caractere 'L'
const char* LED_COLOR_NORMAL = "*LR0G255B0*";   // Cor Verde para estado normal
const char* LED_COLOR_ALERT  = "*LR255G0B0*";   // Cor Vermelha para estado de alerta
const char* LED_COLOR_OFF    = "*LR0G0B0*";     // Comando para desligar os LEDs

// ================== CONFIGURAÇÕES GERAIS E DE SENSORES ==================
// Pinos do hardware
const int LO_PLUS_PIN = 24;  // Pino de detecção de eletrodo desconectado (LA)
const int LO_MINUS_PIN = 22; // Pino de detecção de eletrodo desconectado (RA)
const int ECG_PIN = A0;      // Pino de entrada analógica para o sinal do ECG

// Limites clínicos para classificação e alertas
const int MIN_BPM_NORMAL = 60;
const int MAX_BPM_NORMAL = 100;
const int SPO2_ALERT_THRESHOLD = 95;

// Limites do sensor para filtragem de ruído
const int MIN_BPM_SENSOR = 40;
const int MAX_BPM_SENSOR = 165;
const int HISTORY_SIZE = 5; // Tamanho do buffer para suavização de BPM e SpO2
```

```
// Buffers para suavização dos dados
float bpmHistory[HISTORY_SIZE];
int bpmHistoryIndex = 0;
float filteredBPM = 0;
int validBeatCount = 0;

float spo2History[HISTORY_SIZE];
int spo2HistoryIndex = 0;
float filteredSpO2 = 0;
bool spo2Ready = false;

// ================== VARIÁVEIS DE ESTADO E TIMING ==================
unsigned long lastPrint = 0;
unsigned long lastRead = 0;
const int PRINT_INTERVAL = 1000; // Intervalo para enviar dados (1 segundo)
const int READ_INTERVAL = 10;    // Intervalo de leitura (100 Hz)

// Flags de estado do sistema
bool max30102Connected = false;
bool ecgLeadOff = true;
bool fingerDetected = false;

// ================== VARIÁVEIS DE DETECÇÃO DE SINAL ==================
float smoothedIrValue = 0; // Valor do IR suavizado para detecção de BPM
long ir_max_since_last_beat = 0;
long ir_min_since_last_beat = 100000;
long red_max_since_last_beat = 0;
long red_min_since_last_beat = 100000;

long previous_ir_for_bpm = 0;
bool signalIsRising = false;
unsigned long lastBeatTime = 0;


// ================== SETUP ==================
void setup() {
  // Inicia as comunicações seriais
  Serial.begin(115200); // Comunicação com o PC
  Serial1.begin(9600);  // Comunicação com o módulo Bluetooth HC-05
```

```
  // Configura os pinos de detecção de eletrodos do AD8232
  pinMode(LO_PLUS_PIN, INPUT_PULLUP);
  pinMode(LO_MINUS_PIN, INPUT_PULLUP);

  Serial.println("=== Monitor Cardíaco Híbrido vFinal (4.2) ===");

  // Tenta inicializar o sensor MAX30102
  if (!particleSensor.begin()) {
    Serial.println("AVISO: MAX30102 não encontrado!");
    max30102Connected = false;
  } else {
    Serial.println("MAX30102 encontrado.");
    max30102Connected = true;
    particleSensor.setup();
  }

  // Imprime as instruções de uso no monitor serial do PC
  Serial.println("\n=== INSTRUÇÕES ===");
  Serial.println("Envie '1' (via Bluetooth) para MODO ECG.");
  Serial.println("Envie '2' (via Bluetooth) para MODO BPM/SpO2.");

  // Inicia o sistema no modo ECG por padrão
  switchToEcgMode();
  delay(2000); // Pequena pausa para estabilização inicial
}

// ================== LOOP ==================
void loop() {
  unsigned long currentTime = millis();

  // Verifica constantemente se há um comando de troca de modo vindo do Bluetooth
  checkModeSwitch();

  // Executa a lógica correspondente ao modo atual
  if (currentMode == MODE_ECG) {
    if (currentTime - lastRead >= READ_INTERVAL) {
      ecgLeadOff = (digitalRead(LO_PLUS_PIN) == HIGH || digitalRead(LO_MINUS_PIN) == HIGH);
      if (ecgLeadOff) {
        Serial1.println("*A0"); // Limpa o gráfico do ECG no app
      } else {
```

```
        readECG(); // Lê e envia os dados do ECG
      }
      lastRead = currentTime;
    }
  } else { // MODO_BPM_SPO2
    if (max30102Connected) {
      readOximeter(); // A leitura do oxímetro já controla seu próprio timing
    }
  }

  // Envia os resultados para o usuário a cada segundo
  if (currentTime - lastPrint >= PRINT_INTERVAL) {
    displayResults();
    lastPrint = currentTime;
  }
}

// ================== FUNÇÕES DE CONTROLE DE MODO ==================
void switchToEcgMode() {
  currentMode = MODE_ECG;
  String msg = "\n>>> MODO ECG ATIVADO <<<";
  Serial.println(msg);
  Serial1.println(msg);

  if (max30102Connected) {
    particleSensor.shutDown(); // Desliga o MAX30102 para evitar interferência
  }
  resetAll(); // Limpa todos os dados antigos
}

void switchToOximeterMode() {
  currentMode = MODE_BPM_SPO2;
  String msg = "\n>>> MODO BPM/SpO2 ATIVADO <<<";
  Serial.println(msg);
  Serial1.println(msg);

  if (max30102Connected) {
    particleSensor.wakeUp(); // "Acorda" o sensor MAX30102
    // Configura o sensor com os parâmetros otimizados para SpO2
    particleSensor.setup(0x1F, 4, 3, 100, 411, 4096);
```

```
  }
  resetAll(); // Limpa todos os dados antigos
  Serial1.println("*A0"); // Limpa o gráfico do app para a nova leitura
}

// Escuta por comandos ('1' ou '2') na porta serial do Bluetooth
void checkModeSwitch() {
  if (Serial1.available() > 0) {
    char command = Serial1.read();
    if (command == '1' && currentMode != MODE_ECG) {
      switchToEcgMode();
    } else if (command == '2' && currentMode != MODE_BPM_SPO2) {
      switchToOximeterMode();
    }
  }
}

// ================== FUNÇÕES DE LEITURA E PROCESSAMENTO ==================
void readECG() {
  int rawECGValue = analogRead(ECG_PIN);
  Serial.println(rawECGValue); // Envia para o monitor do PC
  // Envia para o app com o endereçamento do gráfico ('*A...')
  Serial1.println("*A" + String(rawECGValue));
}

void readOximeter() {
  // A biblioteca do sensor possui um buffer interno. Verificamos se há amostra nova.
  if (!particleSensor.available()) {
    particleSensor.check();
    return;
  }

  // Coleta os valores brutos de luz infravermelha e vermelha
  long irValue = particleSensor.getIR();
  long redValue = particleSensor.getRed();
  particleSensor.nextSample(); // Avisa o sensor para preparar a próxima amostra

  // Lógica de detecção de dedo corrigida para o range do ADC de 4096
  if (irValue < 2000) {
    if (fingerDetected) resetAll(); // Se o dedo acabou de ser removido, reseta tudo
```

```
    fingerDetected = false;
    return;
  }

  if (!fingerDetected) {
    fingerDetected = true;
  }

  // Delega o processamento dos sinais para a função especialista
  processSignals(irValue, redValue);
}

// Função principal do algoritmo: processa os sinais brutos para obter BPM e SpO2
void processSignals(long irValue, long redValue) {
  // PARTE 1: DETECÇÃO DE BPM
  // Aplica um filtro passa-baixa para suavizar o sinal IR e facilitar a detecção de picos
  if (smoothedIrValue == 0) smoothedIrValue = irValue;
  smoothedIrValue = (smoothedIrValue * 0.95) + (irValue * 0.05);

  // Detecção de pico: Ocorre quando o sinal estava subindo e agora começa a cair
  if (smoothedIrValue < previous_ir_for_bpm && signalIsRising) {
    unsigned long currentTime = millis();
    // Filtro de tempo para rejeitar ruídos (limita a ~240 BPM)
    if (currentTime - lastBeatTime > 250) {
      // Ignora o primeiro batimento, pois não há intervalo para calcular
      if (lastBeatTime != 0) {
        long interval = currentTime - lastBeatTime;
        float bpm = 60000.0 / interval;

        // Filtra para garantir que o BPM está dentro de uma faixa fisiologicamente possível
        if (bpm > MIN_BPM_SENSOR && bpm < MAX_BPM_SENSOR) {
          // Adiciona BPM ao histórico para suavização
          bpmHistory[bpmHistoryIndex] = bpm;
          bpmHistoryIndex = (bpmHistoryIndex + 1) % HISTORY_SIZE;
          validBeatCount++;

          // PARTE 2: CÁLCULO DE SPO2
          // Agora que temos um batimento válido, usamos os valores min/max coletados
          long irAC = ir_max_since_last_beat - ir_min_since_last_beat;
          long redAC = red_max_since_last_beat - red_min_since_last_beat;
```

```
        long irDC = (ir_max_since_last_beat + ir_min_since_last_beat) / 2;
        long redDC = (red_max_since_last_beat + red_min_since_last_beat) / 2;

        // Checa se há um pulso detectável (componente AC forte o suficiente)
        if (irAC > 100 && redAC > 100) {
          // Calcula a Razão das Razões (R)
          float R = ( (float)redAC / (float)redDC ) / ( (float)irAC / (float)irDC );
          // Aplica a fórmula empírica para obter o SpO2
          float currentSpO2 = 110 - 25 * R;

          if (currentSpO2 > 100) currentSpO2 = 100;
          if (currentSpO2 >= 70) {
            spo2History[spo2HistoryIndex] = currentSpO2;
            spo2HistoryIndex = (spo2HistoryIndex + 1) % HISTORY_SIZE;
            spo2Ready = true;
          }
        }
        // Reseta os valores min/max para o próximo ciclo cardíaco
        ir_max_since_last_beat = 0; ir_min_since_last_beat = 100000;
        red_max_since_last_beat = 0; red_min_since_last_beat = 100000;
      }
    }
    lastBeatTime = currentTime; // Atualiza o tempo do último batimento válido
  }
 }

 // Atualiza o estado de subida/descida do sinal para a próxima iteração
 signalIsRising = (smoothedIrValue > previous_ir_for_bpm);
 previous_ir_for_bpm = smoothedIrValue;

 // PARTE 3: COLETA DE DADOS PARA SPO2
 // Coleta continuamente os valores de pico e vale do sinal bruto entre os batimentos
 if(irValue > ir_max_since_last_beat) ir_max_since_last_beat = irValue;
 if(irValue < ir_min_since_last_beat) ir_min_since_last_beat = irValue;
 if(redValue > red_max_since_last_beat) red_max_since_last_beat = redValue;
 if(redValue < red_min_since_last_beat) red_min_since_last_beat = redValue;
}

// Função auxiliar para calcular a média de um buffer, ignorando valores nulos
float calculateAverage(float* buffer, int size) {
```

```
  float sum = 0;
  int count = 0;
  for (int i = 0; i < size; i++) {
    if (buffer[i] > 0) {
      sum += buffer[i];
      count++;
    }
  }
  // Evita divisão por zero
  return (count > 0) ? (sum / count) : 0;
}

// ================== FUNÇÃO DE EXIBIÇÃO DE DADOS ==================
void displayResults() {
  String message = "";

  if (currentMode == MODE_ECG) {
    if (ecgLeadOff) {
      message = "MODO ECG: Eletrodos Desconectados!";
      Serial.println(message);
    }
  } else { // MODO_BPM_SPO2 - ÚNICO LUGAR QUE CONTROLA O LED
    if (!fingerDetected) {
      message = "Posicione o dedo no sensor";
      Serial1.println(LED_COLOR_OFF);
    } else if (validBeatCount < HISTORY_SIZE) {
      message = "Calculando... (" + String(validBeatCount) + "/" + String(HISTORY_SIZE) + ")";
      Serial1.println(LED_COLOR_OFF);
    } else {
      // Calcula os valores finais suavizados
      filteredBPM = calculateAverage(bpmHistory, HISTORY_SIZE);
      filteredSpO2 = calculateAverage(spo2History, HISTORY_SIZE);

      // Classifica o BPM
      String bpmClassification = " (Normal)";
      if (filteredBPM < MIN_BPM_NORMAL) {
        bpmClassification = " (Bradicardia)";
      } else if (filteredBPM > MAX_BPM_NORMAL) {
        bpmClassification = " (Taquicardia)";
      }
```

```
    // Monta a mensagem de texto para o usuário
    message += "BPM: " + String((int)round(filteredBPM)) + bpmClassification;
    message += " | ";

    if (spo2Ready && filteredSpO2 > 0) {
      message += "SpO2: " + String((int)round(filteredSpO2)) + " %";
    } else {
      message += "SpO2: -- %";
    }

    // Lógica de Alerta para o LED RGB
    bool alertState = false;
    if ((filteredBPM < MIN_BPM_NORMAL && filteredBPM > 0) ||
        (filteredBPM > MAX_BPM_NORMAL) ||
        (filteredSpO2 < SPO2_ALERT_THRESHOLD && filteredSpO2 > 0)) {
      alertState = true;
    }

    if (alertState) {
      Serial1.println(LED_COLOR_ALERT); // Envia comando para LED Vermelho
    } else {
      Serial1.println(LED_COLOR_NORMAL); // Envia comando para LED Verde
    }
   }
   // Envia a mensagem de texto para ambos os seriais
   Serial.println(message);
   Serial1.println(message);
  }
}

// Reseta todas as variáveis para uma nova medição limpa
void resetAll() {
  Serial.println("Resetando dados...");

  for (int i = 0; i < HISTORY_SIZE; i++) {
    bpmHistory[i] = 0;
    spo2History[i] = 0;
  }
  bpmHistoryIndex = 0;
```

```
  spo2HistoryIndex = 0;
  validBeatCount = 0;
  filteredBPM = 0;
  filteredSpO2 = 0;
  spo2Ready = false;

  fingerDetected = false;
  smoothedIrValue = 0;
  previous_ir_for_bpm = 0;
  signalIsRising = false;
  lastBeatTime = 0;

  ir_max_since_last_beat = 0; ir_min_since_last_beat = 100000;
  red_max_since_last_beat = 0; red_min_since_last_beat = 100000;
}
```