UNIVERSIDADE PRESBITERIANA MACKENZIE
– Faculdade de Computação e Informática –

ALTERNATIVA ECONÔMICA PARA MONITORAMENTO DE FREQUÊNCIA CARDÍACA

Henrique Yuji Murasaki

Professor: André Luiz de Oliveira

Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie (UPM) – São Paulo, SP – Brazil

10330549@mackenzista.com.br

Abstract.

The project aims to develop a heartbeat sensor using the NodeMCU ESP8266 module platform. Using a pulse sensor, the device can monitor heartbeats in real time. Subsequently, the project processes this information and presents it on a remote display that must not be connected via the network. This approach offers an affordable solution for cardiac monitoring.

Resumo.

O projeto visa desenvolver um sensor de batimentos cardíacos utilizando a plataforma o módulo NodeMCU ESP8266. Através de um sensor de pulso, o dispositivo é capaz de monitorar os batimentos cardíacos em tempo real. Posteriormente, o projeto processa essas informações e as apresenta em um display remoto no qual deve ser conectado pela rede. Essa abordagem oferece uma solução acessível para o monitoramento cardíaco.

1. Introdução

O monitoramento da frequência cardíaca é fundamental para o bem-estar humano, pois é um dos principais indicadores da saúde do coração. A frequência cardíaca normal varia entre 50 e 90 batimentos por minuto. Mudanças no ritmo e na frequência podem indicar a presença de doenças ou até variações emocionais. No dia a dia, os médicos usam equipamentos especializados para essa finalidade, que, embora sejam altamente eficazes, podem ter custos elevados e exigir infraestrutura técnica específica.

Nesses casos, a busca por alternativas de baixo custo e fácil acesso é muito procurada. Soluções inovadoras que ajudem a superar a barreira dos recursos financeiros e ofereçam monitoramento preciso da frequência cardíaca têm grande potencial para revolucionar a forma como a saúde cardiovascular é gerenciada em comunidades com recursos limitados.

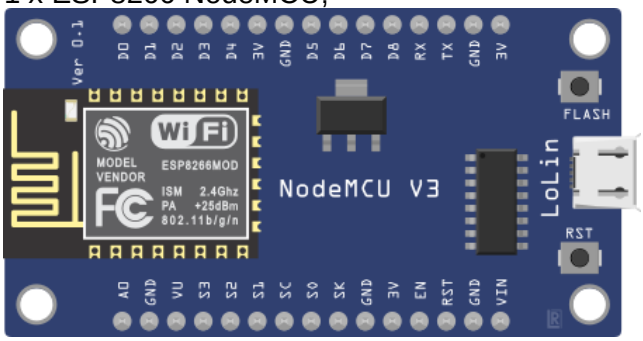
O projeto mencionado tem como objetivo desenvolver um sensor de frequência cardíaca usando o Arduino como base. Usando um sensor de pulso, o dispositivo é capaz de monitorar os batimentos cardíacos em tempo real, apresentando uma alternativa diferente aos dispositivos convencionais, sendo uma solução de baixo custo que pode ser replicada e implementada em outros contextos.

1.1. Subseções

2. Materiais e métodos

MATERIAIS

- 1 x ESP8266 NodeMCU;



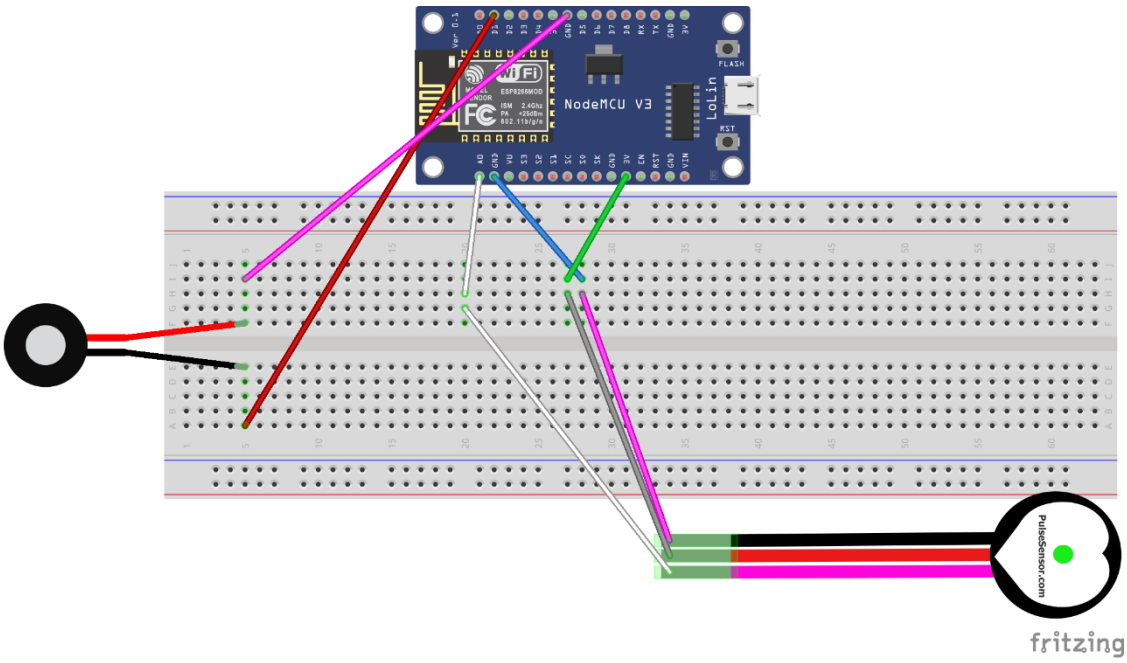
- 1 x Cabo USB Tipo A-B compatível com Arduino UNO;

- 1 x Sensor de batimento cardíaco / Monitor de pulso;

- Jumpers macho-fêmea

- Buzzer

MODELO DE MONTAGEM



MÉTODOS

setup() : Configura os pinos, inicializa a serial, tenta conectar ao WiFi e configura o servidor MQTT.

loop() : Verifica a conexão MQTT, lê o sensor, detecta batimentos, calcula o BPM, imprime e publica o BPM.

publicarBatimentosNoTopico() : Publica os dados de BPM no tópico MQTT.

reconnectMQTT() : Tenta reconectar ao servidor MQTT em caso de desconexão.

FUNCIONAMENTO

Este código Arduino é escrito para um microcontrolador ESP8266 e realiza as seguintes tarefas principais: conecta-se a uma rede WiFi, lê dados de um sensor de pulso, aciona um buzzer em resposta aos batimentos cardíacos detectados, calcula a frequência

cardíaca (BPM) e publica esses dados em um broker MQTT. Aqui está uma explicação detalhada de como o código funciona:

Configurações Iniciais

Inclusão de Bibliotecas e Declaração de Variáveis

1- Bibliotecas:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

- ESP8266WiFi.h: permite a conexão à rede WiFi
- PubSubClient.h: Facilita a comunicação com o broker MQTT

2- Credenciais WiFi:

```
const char *ssid = "LIVE TIM_DFB8_2G";
const char *password = "wlanb52047";
```

3- Configuração do MQTT:

```
const char *mqtt_broker = "mqtt.eclipseprojects.io";
const char *mqtt_topic = "sensor/bpm";
const char *mqtt_client = "clientsensor";
```

4- Pinos e Variáveis do Sensor e Atuador:

```
int PulseSensorPin = A0;
int BuzzerPin = D1;
int Signal;
int Threshold = 572;
unsigned long lastBeatTime = 0;
unsigned int bpm = 0;
```

Função 'setup()'

1- Configuração Inicial:

```
pinMode(BuzzerPin, OUTPUT); // Configura o pino do buzzer como saída.
    Serial.begin(115200); // Inicializa a comunicação serial.
    WiFi.disconnect(true); // Desconecta qualquer conexão WiFi existente.
    delay(1000);
    WiFi.begin(ssid, password); // Tenta conectar à rede WiFi.
    client.setServer(mqtt_broker, 1883); // Configura o servidor MQTT.
```

2- Tentativa de Conexão WiFi:

```
Serial.println("Conectando ao wifi.....");
int retries = 0;
while (WiFi.status() != WL_CONNECTED && retries < 20) {
  delay(500);
  Serial.print(".");
  retries++;
}
if (WiFi.status() == WL_CONNECTED) {
  Serial.println("Wifi conectado");
} else {
  Serial.println("Falha ao conectar ao Wifi");
}
```

Função 'loop()'

1- Verificação da Conexão MQTT:

```
if (!client.connected()) {
  reconnectMQTT();
}
client.loop();
```

2- Leitura do Sensor de Pulso:

```
Signal = analogRead(PulseSensorPin);
```

3- Detecção de Batimentos e Cálculo de BPM:

```
if (Signal > Threshold) {
  digitalWrite(BuzzerPin, HIGH); // Ativa o buzzer.
  delay(100); // Duração ON.
  digitalWrite(BuzzerPin, LOW); // Desativa o buzzer.
  bpm = 15000 / (millis() - lastBeatTime); // Calcula BPM.
  lastBeatTime = millis();
} else {
  digitalWrite(BuzzerPin, LOW); // Desativa o buzzer.
  bpm = 0;
}
```

4- Publicação e Impressão de BPM:

```
static unsigned long lastPrintTime = 0;
if (millis() - lastPrintTime >= 1000) {
  Serial.print("BPM: ");
  Serial.println(bpm * 6);
  lastPrintTime = millis();
  publicarBatimentosNoTopico();
}
delay(1000); // Aguarda 1 segundo.
```

Funções Auxiliares

- ***'publicarBatimentosNoTopico()'***

```
void publicarBatimentosNoTopico() {
  client.publish(mqtt_topic, String(bpm * 6).c_str(), true);
}
```

Publica o valor do BPM no tópico MQTT especificado.

- **'reconnectMQTT()'**

```
void reconnectMQTT() {
  while (!client.connected()) {
    Serial.print("Tentando conectar ao MQTT...");
    if (client.connect(mqtt_client)) {
      Serial.println("Conectado ao MQTT");
    } else {
      Serial.print("Falha na conexão ao MQTT, rc=");
      Serial.print(client.state());
      Serial.println(" Tentando novamente em 5 segundos...");
      delay(5000);
    }
  }
}
```

Tenta reconectar ao servidor MQTT até obter sucesso.

RESUMO DO FUNCIONAMENTO

1- **Configuração e Conexão WiFi:** O ESP8266 se conecta à rede Wifi especificada no '*setup()'.*
2- **Configuração e Conexão MQTT:** Configura o cliente MQTT e tenta conectar ao broker MQTT.
3- **Leitura e Processamento so Sinal do Sensor:**

- No loop principal, o sinal do sensor de pulso é lido.
- Se o sinal estiver acima do limiar, o buzzer é ativado brevemente e a frequência cardíaca é calculada.

4- **Publicação de Dados:** A cada Segundo, o BPM é publicado no tópico MQTT e impresso no console serial.
5- **Reconexão Automática:** Se a conexão MQTT for perdida, o Código tenta reconectar automaticamente.

PSEUDOCÓDIGO

INÍCIO

// Definições de WIFI

Defina ssid como "LIVE TIM_DFB8_2G"

Defina password como "wlanb52047"

Crie um cliente WiFi

// Definições de MQTT

Defina mqtt_broker como "mqtt.eclipseprojects.io"

Crie um cliente MQTT usando o cliente WiFi

Defina mqtt_topic como "sensor/bpm"

Defina mqtt_client como "clientsensor"

// Definições de Sensor e Atuador

Defina PulseSensorPin como A0 (pino de entrada para o sensor de pulso)

Defina BuzzerPin como D1 (pino de saída para o buzzer)

Defina Signal (variável para armazenar o sinal do sensor de pulso)

Defina Threshold como 572 (limiar para detecção de batimento)

Defina lastBeatTime como 0 (tempo do último batimento detectado)

Defina bpm como 0 (batimentos por minuto)

// SETUP

Defina BuzzerPin como saída

Inicie a comunicação serial com uma taxa de transmissão de 115200 bits por segundo

Desconecte qualquer conexão WiFi existente

Aguarde 1 segundo

Tente conectar à rede WiFi usando o ssid e a senha definidos anteriormente

Configure o servidor MQTT usando o broker e a porta definidos anteriormente

Imprima "Conectando ao wifi....."

Faça um loop de tentativas de conexão WiFi:

  Se WiFi estiver conectado, imprima "Wifi conectado"

  Senão, aguarde 500 milissegundos, imprima ".", e tente novamente (até 20 tentativas)

Se não conseguir conectar ao WiFi após 20 tentativas, imprima "Falha ao conectar ao Wifi"

// LOOP

Se o cliente MQTT não estiver conectado, chame a função reconnectMQTT para tentar reconectar

Execute o loop do cliente MQTT para processar qualquer mensagem recebida

Leia o valor do sensor de pulso e armazene em Signal

// Detecção dos batimentos

Se Signal for maior que Threshold, então

  Ative o Buzzer

  Aguarde 100 milissegundos

  Desative o Buzzer

  Calcule a frequência cardíaca em batimentos por minuto (BPM) usando a fórmula bpm = 15000 / (tempo atual - tempo do último batimento)

  Atualize lastBeatTime com o tempo atual

Senão

  Desative o Buzzer

  Defina bpm como 0

Verifique o tempo para imprimir e publicar BPM:

  Se passou 1 segundo desde a última impressão, então

    Imprima "BPM: " e o valor de bpm*6

    Atualize lastPrintTime com o tempo atual

    Chame a função publicarBatimentosNoTopico para publicar os batimentos no tópico MQTT

Aguarde 1 segundo

// Publicar MQTT

Função publicarBatimentosNoTopico:

  Publique bpm*6 no tópico MQTT

// Reconectar MQTT

Função reconnectMQTT:

  Enquanto o cliente MQTT não estiver conectado, faça

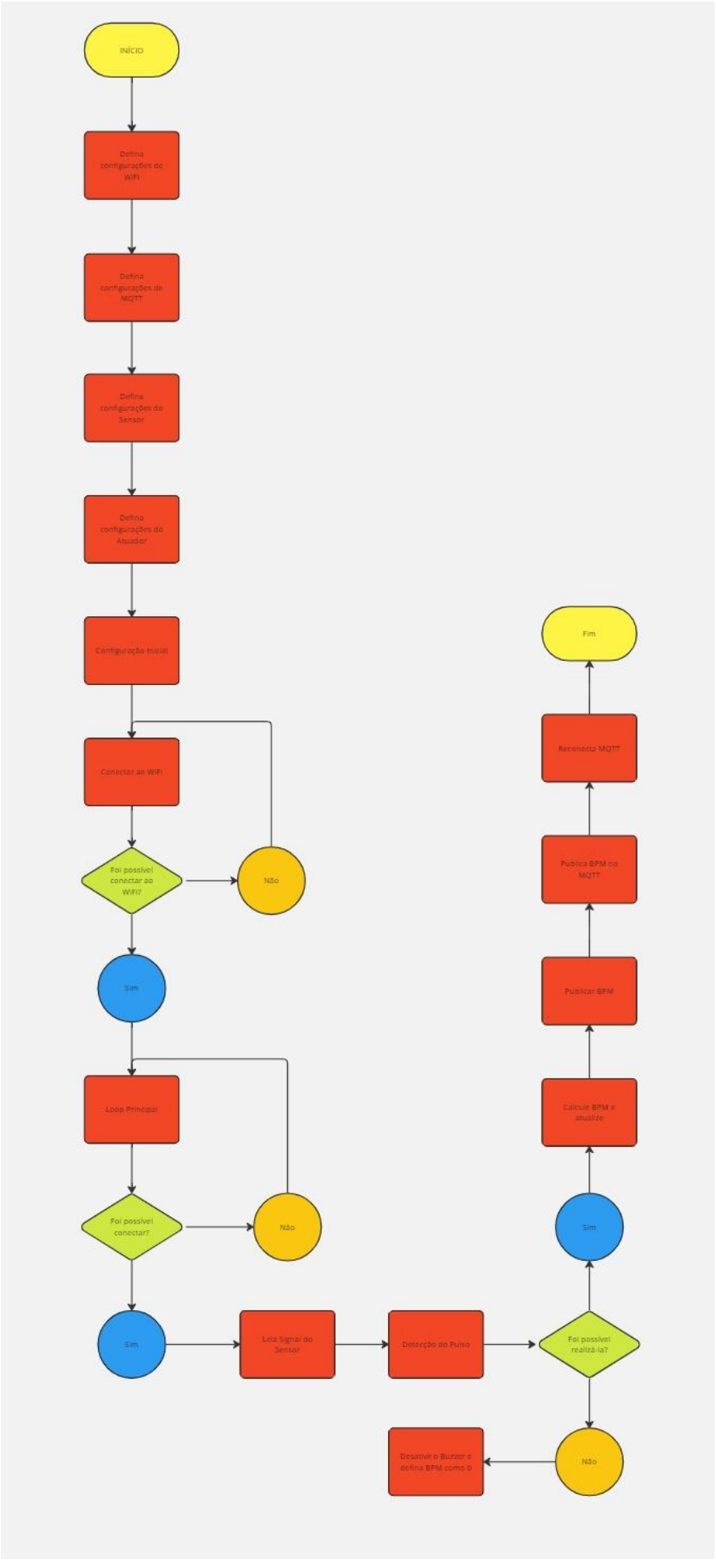
    Imprima "Tentando conectar ao MQTT..."

    Se conseguir conectar ao MQTT, imprima "Conectado ao MQTT"

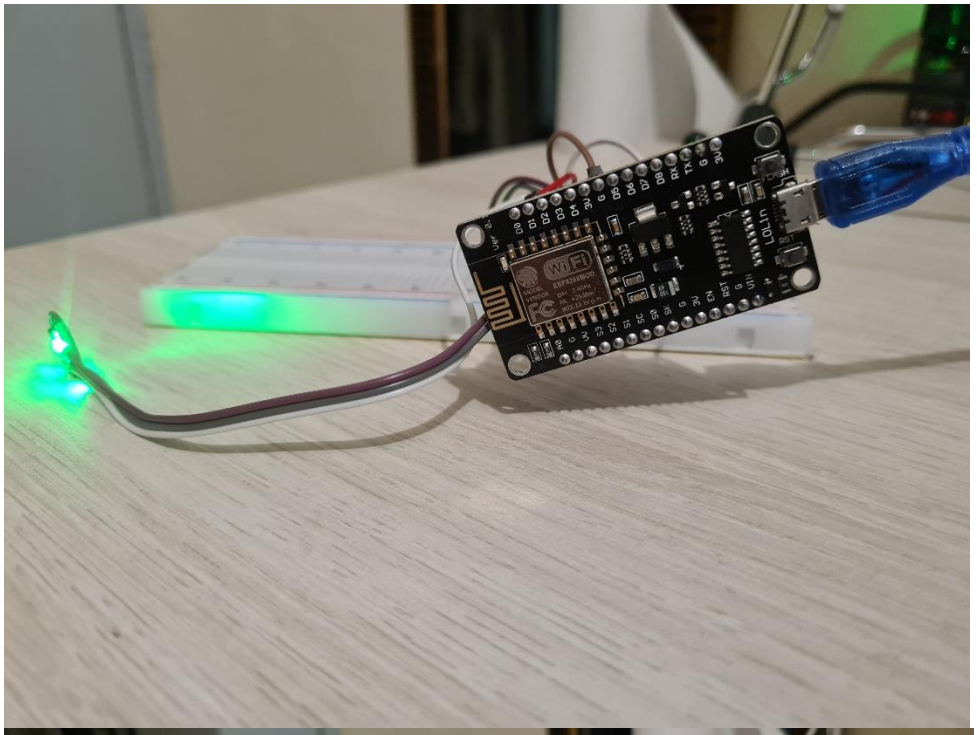
    Senão, imprima "Falha na conexão ao MQTT, rc=" e o estado do cliente, aguarde 5 segundos e tente novamente

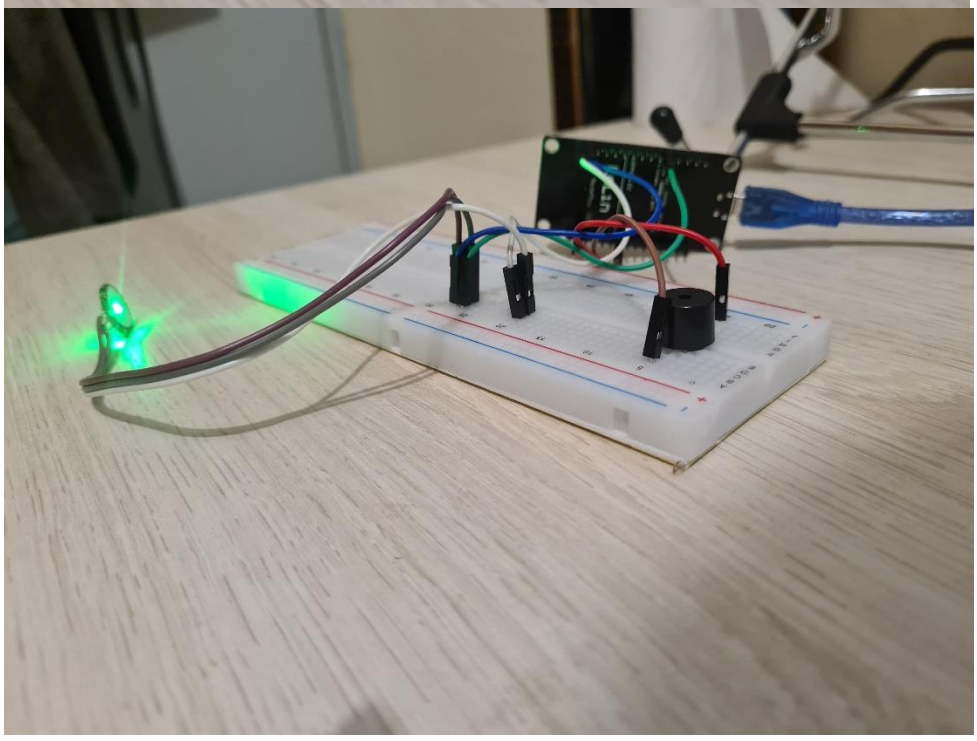
FIM

FLUXOGRAMA


INÍCIO
Defina configurações do WiFi
Defina configurações do MQTT
Defina configurações do Sensor
Defina configurações do Atuador
Configuração Inicial
Conectar ao WiFi
Foi possível conectar ao WiFi?
Não
Sim
Loop Principal
Foi possível conectar?
Não
Sim
Leia Sinal do Sensor
Detecção do Pulso
Foi possível realizá-la?
Não
Desative o Buzzer e defina BPM como 0
Sim
Calcule BPM e atualize
Publicar BPM
Publica BPM no MQTT
Reconecta MQTT
Fim

IMAGENS DO PROTÓTIPO MONTADO





Nas é possível identificar o sensor de batimento cardíaco, o modulo NodeMCU ESP8266, cabos jumpers macho-fêmea e o buzzer que foi utilizado para o retorno sonoro da leitura do batimento cardíaco.

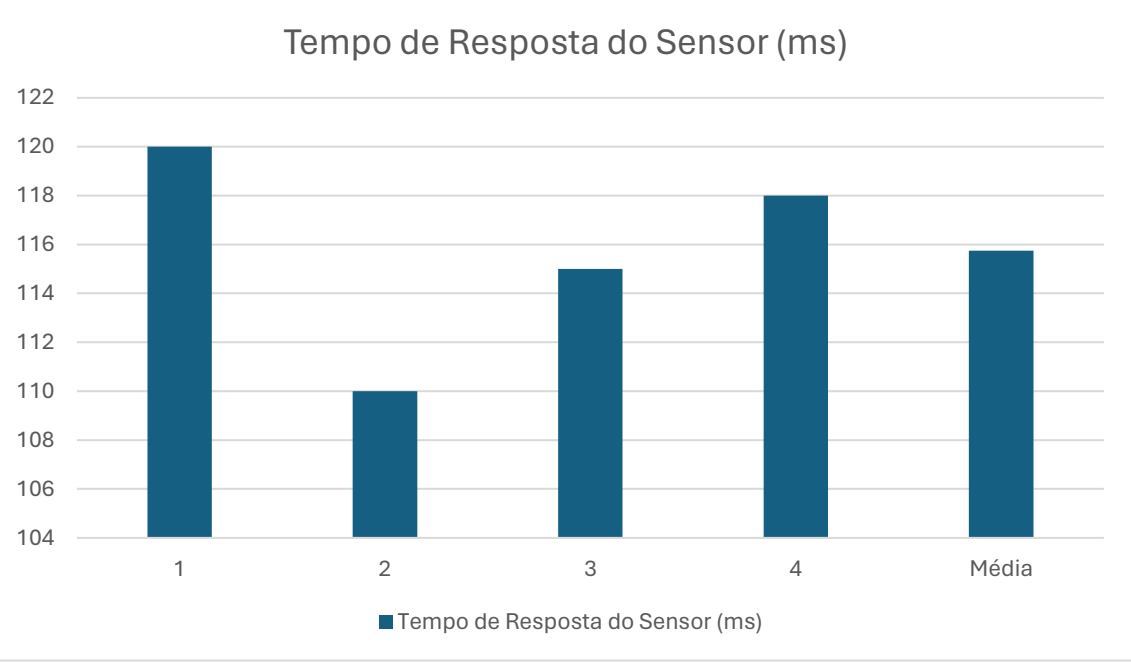
VÍDEO-DEMONSTRAÇÃO

Aqui está o vídeo-demonstração upado no Youtube da apresentação do funcionamento do projeto:

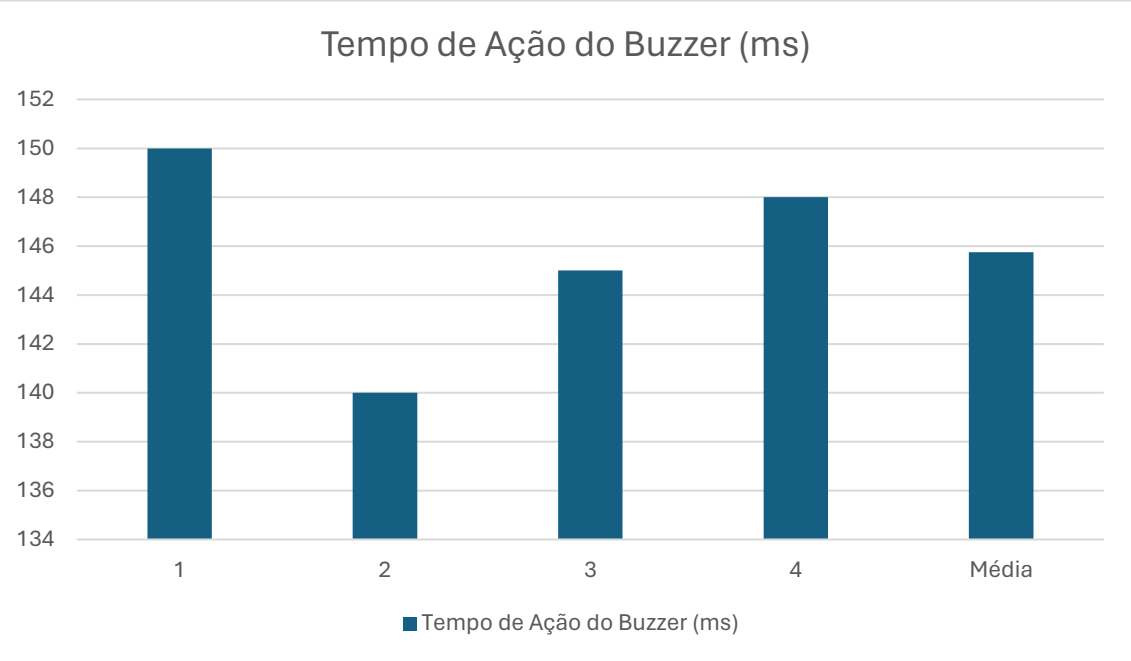
https://youtu.be/RPIP-yA6nx8

TEMPO MÉDIO REALIZADO ENTRE O ENVIO DE COMANDOS E AÇÃO DO ATUADOR

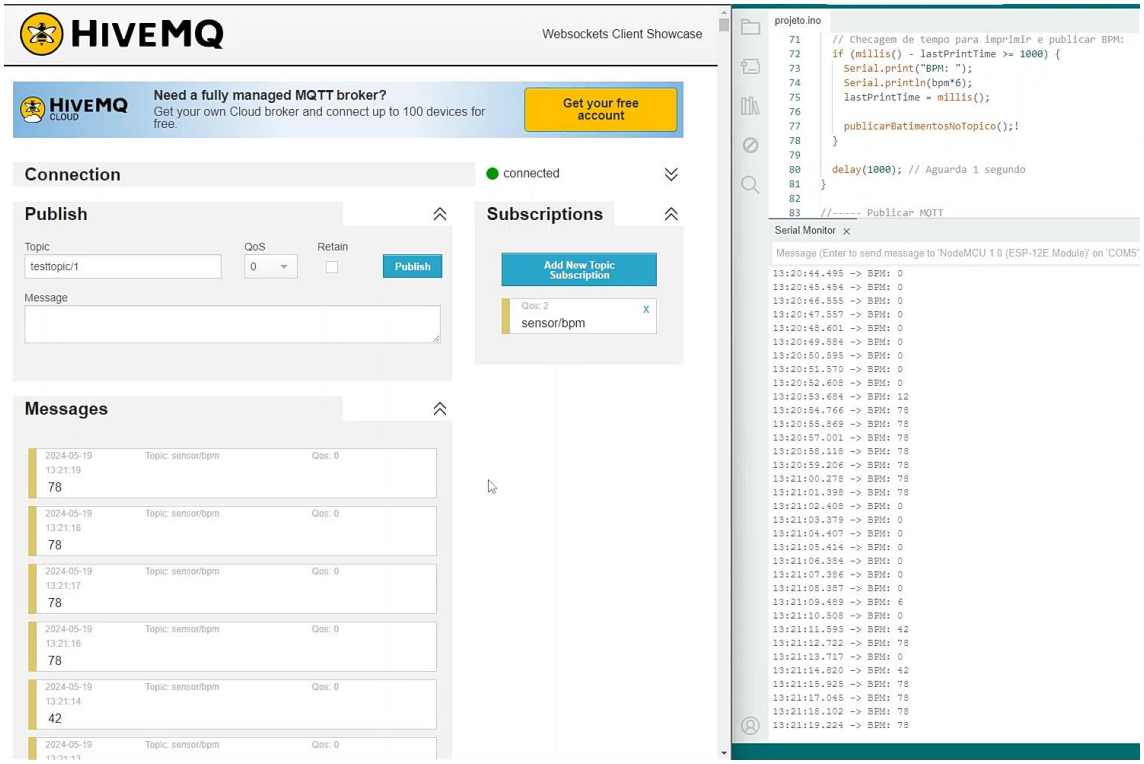
Medição	Tempo de Resposta do Sensor (ms)
1	**120**
2	**110**
3	**115**
4	**118**
Média	115,75



Medição	Tempo de Ação do Buzzer (ms)
1	**150**
2	**140**
3	**145**
4	**148**
Média	145,75



CAPTURA DE TELA DEMONSTRANDO O FUNCIONAMENTO DO PROTÓTIPO E COMUNICAÇÃO COM O BROKER MQTT



Conclusões

i) Os objetivos propostos foram alcançados?

Sim, o objetivo inicial foi alcançado com sucesso de acordo com a proposta feita início do projeto.

ii) Quais são os principais problemas enfrentados e como foram resolvidos?

A própria conexão com o MQTT foi um tanto quanto difícil de ser feita porém com muita pesquisa em artigos foi possível corrigir essa avaria.

ii) Quais são as vantagens e desvantagens do projeto?

As vantagens são que foi possível descobrir uma forma menos custosa para sabermos sobre a frequência cardíaca de pessoa para pessoa. Em um monitor cardíaco ou frequencímetro, por exemplo, com certeza seria algo com valor muito acima do valor utilizado no projeto. A desvantagem é que é leva uma certa quantidade de tempo para ser elaborado o sistema completo para que possa ser utilizado com todos os acessórios.

iii) O que deveria/poderia ser feito para melhorar o projeto?

Comprar componentes de tecnologia mais avançada ajudaria muito no projeto, apesar de torna-lo mais caro seria muito mais preciso para a leitura dos batimentos cardíacos.

5. Referências

KENSHIMA, Gedeane. Aprenda a usar o Sensor de frequência cardíaca. Maker Hero, 2017. Disponível em: https://www.makerhero.com/blog/aprenda-usar-o-sensor-de-frequencia-cardiaca/. Acesso em: 07/03/2024

PANOEIRO, Jonathan. Frequência cardíaca normal: batimento cardíaco por idade. Tua Saude, 2023. Disponível em: https://www.tuasaude.com/frequencia-cardiaca/. Acesso em: 07/03/2024

VIANA, Carol. Como utilizar o sensor de batimento cardíaco / monitor de pulso com Arduino. Blog da Robótica, 2023. Disponível em: https://www.blogdarobotica.com/2023/08/21/como-utilizar-o-sensor-de-batimento-cardiaco-monitor-de-pulso-com-arduino/. Acesso em: 21/03/2024.

BERTOLETI, Pedro. Controle e Monitoramento IoT com NodeMCU e MQTT. Maker Hero, 2016. Disponível em: https://www.makerhero.com/blog/controle-monitoramento-iot-nodemcu-e-mqtt/. Acesso em 15/05/2024.

Getting Started with the HiveMQ MQTT Platform. HIVEMQ, 2023. Disponível em: https://docs.hivemq.com/hivemq/latest/user-guide/getting-started.html. Acesso em: 15/05/2024.

NodeMcu e Esp8266: medindo e publicando dados com MQTT. Alura. Disponível em: https://www.alura.com.br/conteudo/iot-com-nodemcu. Acesso em 15/05/2024.