UNIVERSIDADE PRESBITERIANA MACKENZIE
– Faculdade de Computação e Informática –

Criação de um protótipo para controle de estoque

Klenner Gomes do Prado, João Victor de França Freitas, Luan Henrique Torres

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

10370067@mackenzista.com.br, 1036538@mackenzista.com.br,
10369874@mackenzista.com.br

Abstract. *This project aims to develop a stock monitoring system using hardware and communication technologies to ensure efficient inventory management. The system consists of a sensor that detects the absence of specific items in stock. When an item is missing, the sensor triggers an LED, providing an immediate visual alert for the staff.*

To enhance communication and ensure that the information reaches the responsible personnel quickly, the system uses an ESP32 module to send messages via app. This functionality allows staff to receive notifications on their mobile devices, regardless of their location, facilitating a prompt and efficient response.

Resumo. *Este projeto visa desenvolver um sistema de monitoramento de estoque utilizando tecnologias de hardware e comunicação para garantir a eficiência na gestão de inventário. O sistema é composto por um sensor que detecta a ausência de itens específicos no estoque. Quando um item está em falta, o sensor aciona um LED, que serve como um alerta visual imediato para os funcionários.*

Para ampliar a comunicação e garantir que a informação chegue rapidamente aos responsáveis, o sistema utiliza um módulo ESP32 para enviar mensagens via aplicativo. Esta funcionalidade permite que os funcionários recebam notificações em seus dispositivos móveis, independentemente de onde estejam, facilitando uma resposta rápida e eficiente.

1. Introdução

No contexto da Indústria 4.0 e da Internet das Coisas Industrial (IoT), a gestão eficiente de estoque é crucial para garantir a continuidade das operações e a satisfação dos clientes. Este artigo apresenta o desenvolvimento de um sistema de monitoramento de estoque inovador, que utiliza tecnologias de hardware e comunicação para detectar a ausência de itens e notificar os responsáveis de maneira rápida e eficaz.

O sistema proposto integra um sensor de detecção, um controlador, as conexões com jumpers e um módulo para envio de notificações via aplicativo. Quando um item está em falta, o sensor calcula a distância até o produto e devido a variável proposta para distância mínima até o produto, é possível reconhecer se o produto existe ou não, de forma que o controlador junto a conexão MQTT envia uma mensagem para o dispositivo móvel do individual, garantindo que a informação chegue rapidamente aos responsáveis pela reposição.

Este projeto não só melhora a eficiência operacional, mas também minimiza o risco de falta de estoque, assegurando que os itens estejam sempre disponíveis quando necessário. A implementação de tal sistema representa um passo significativo em direção à automação e otimização dos processos de gestão de estoque, alinhando-se com os princípios da Indústria 4.0 (ASSIS, 2020).

2. Materiais e métodos

Para o funcionamento do projeto, algumas atividades e materiais precisam ser organizados:

- Construir um protótipo simulando um estoque.
- Codificar junto a IDE do Arduino para controlar todo o sistema de monitoramento do estoque.
- Utilizar o controlador ESP32 de forma a garantir que seja possível receber os alertas e gerenciar a parte lógica do projeto.
- Utilizar sensor de movimento HCSR04 para detectar a distância até o produto e as movimentações de retirada de itens.
- Led Branco para identificar visualmente a mudança.
- Protoboard: Utilizada para montagem e conexão dos componentes eletrônicos.
- Cabo USB para utilizar na alimentação do projeto.
- Cabos Jumpers para garantir a conectividade entre os módulos.

Os materiais e métodos escolhidos:

- Controlador: O modelo ESP32 é altamente recomendado, pois já possui conectividade Wi-Fi integrada.
- Sensor: O sensor de distância será o HCSR04 para detectar padrões e movimentações de retirada de itens.
- Broker MQTT: Um servidor MQTT para comunicação, que pode ser configurado localmente ou usar um serviço online. Nesse caso será utilizado o Broker EMQX.
- Biblioteca PubSubClient: Para a comunicação via MQTT no controlador, Biblioteca Wifi para conexão com a rede, Biblioteca do ESP32.
- Led Branco para identificar a mudança e gerar um alerta visual quando é necessário repor o estoque.

Para o projeto, será utilizado o app MQTT Dashboard, que é uma ferramenta para testar e monitorar dispositivos IoT que utilizam do protocolo. Dessa forma, ele possibilita a

visualização e publicação dos dados enviados e interage com o modelo do controlador escolhido (ESP32) diretamente do celular.

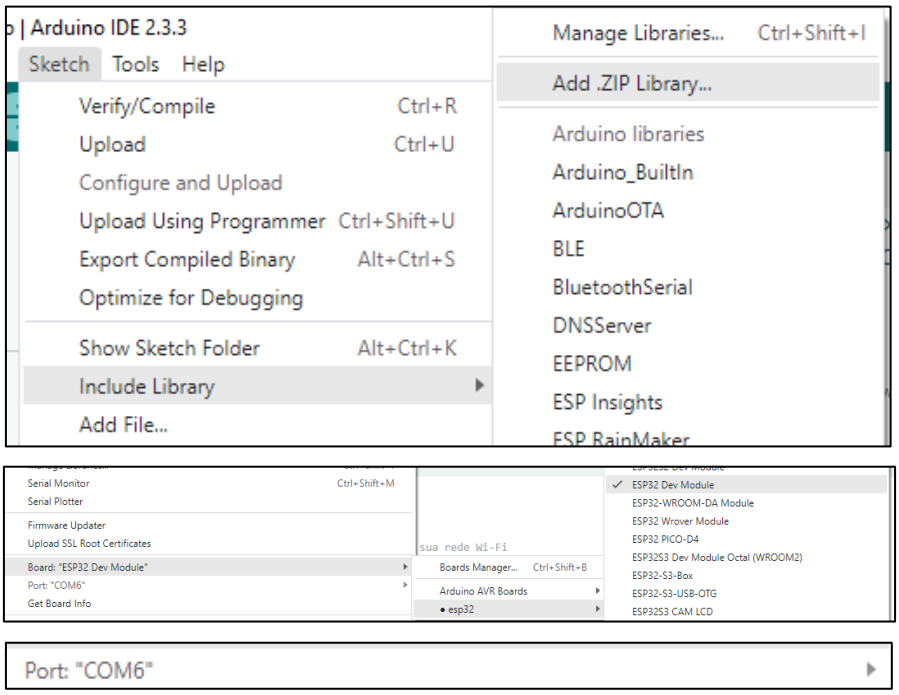
Para a programação, usaremos a IDE 2.3.3 do controlador juntamente com a linguagem de programação C++. Dessa forma, acessando o site oficial do mesmo, fazendo a

instalação e configurando a ESP32 na IDE 2.3.3, podemos iniciar o processo de programação do prototipo.

É possível fazer o download da IDE diretamente no site (https://www.arduino.cc/en/software) e a partir deste, devemos fazer algumas configurações:

O primeiro passo é quando já montado o prototipo, conectar o USB do controlador ao computador, se necessário, deverá ser baixado uma atualização de driver para compatibilidade com a porta USB.

Dentro da IDE, será necessário definir qual a board está usando e qual a porta para conexão. Além, também será necessário baixar as bibliotecas do ESP32 e do MQTT.



Depois de configurado tais cenários, será preciso criar a conexão com o protocolo MQTT via Broker, e também configurar o Widget (Mensagem de Texto) que será recebido.

Para tal, é necessário baixar e instalar o app do MQTT Dashboard Client (Android)

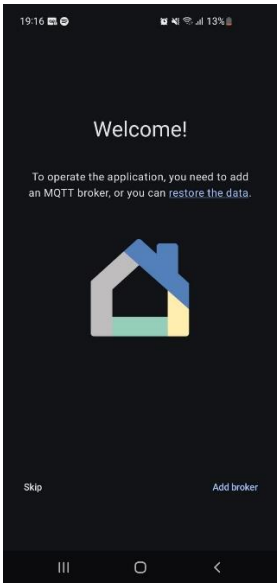


Figura 1 – Tela de início do App. Fonte: Imagem própria.

Logo, será necessário inserir os dados do broker, como:

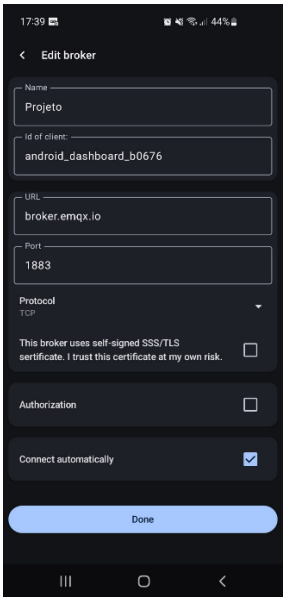
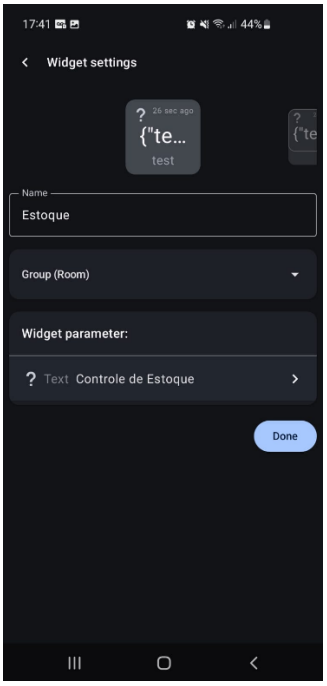


Figura 2 – Tela de configuração. Fonte: Imagem própria.

- Nome do servidor.
- Id do client (já configurado para o dispositivo).
- URL (que será o endereço ip ou URL do broker MQTT).
- Porta 1883 (já configurada para o dispositivo).

Após a configuração, será necessário adicionar os tópicos para monitoramento e envio dos dados coletados, como por exemplo a notificação de movimentação de estoque.
É possivel também adicionar widgets no app para melhor visualização, sendo possivel os seguintes formatos:



- Text View: Para visualizar mensagens publicadas em um tópico específico (por exemplo, atualizações sobre o status do estoque).

Feito todos esses processos, será necessário fazer a integração com o controlador por via de código para que ele receba corretamente as mensagens quando o sensor HCSR04 detectar a alteração.

O código utilizado:

```ino
#include <WiFi.h>
#include <PubSubClient.h>

// Configurações de Wi-Fi
const char* ssid = "####";     // Substitua pelo nome da sua rede Wi-Fi
const char* password = "####";  // Substitua pela senha da sua rede Wi-Fi

// Configurações MQTT
const char* mqtt_server = "broker.emqx.io";  // URL ou IP do broker MQTT
const int mqtt_port = 1883;                  // Porta MQTT padrão
const char* mqtt_username = "luan";
const char* mqtt_password = "public";
const char* mqtt_topic = "emqx/esp32";  // Tópico onde as mensagens serão publicadas

WiFiClient espClient;
PubSubClient client(espClient);

// Configurações do sensor

const int white = 13;
const int PINO_TRIG = 14;
const int PINO_ECHO = 25;

void setup() {
  Serial.begin(9600);
  pinMode (white, OUTPUT);

  // Configuração Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando ao Wi-Fi...");
  }
  Serial.println("Conectado ao Wi-Fi!");

  // Configuração MQTT
  client.setServer(mqtt_server, mqtt_port);

  String client_id = "esp32-client-";
  client_id += String(WiFi.macAddress());
  while (!client.connected()) {
    Serial.println("Conectando ao broker MQTT...");
    if (client.connect("ESP32Client")) {
      Serial.println("Conectado ao broker MQTT!");
    } else {
```

```ino
    } else {
      Serial.print("Falha na conexão. Código de erro: ");
      Serial.println(client.state());
      delay(2000);
    }
  }

  pinMode(PINO_TRIG, OUTPUT);
  pinMode(PINO_ECHO, INPUT);
}

void loop() {
  // Garante a conexão com o broker MQTT
  if (!client.connected()) {
    while (!client.connected()) {
      Serial.println("Reconectando ao broker MQTT...");
      if (client.connect("ESP32Client")) {
        Serial.println("Reconectado ao broker MQTT!");
      } else {
        Serial.print("Falha na reconexão. Código de erro: ");
        Serial.println(client.state());
        delay(2000);
      }
    }
  }

  // Envia pulso ao sensor
  digitalWrite(PINO_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PINO_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PINO_TRIG, LOW);

  long duracao = pulseIn(PINO_ECHO, HIGH);
  float distancia = (duracao * 0.0343) / 2;


  if(distancia >= 7.5) {
    digitalWrite(white,HIGH);    // Liga o LED se a distância for maior que 7.5 centímetros
  }
  else {
    digitalWrite(white,LOW);     // Desliga o LED
  }
```

```
88
89    String mensagem;
90    if (distancia >= 7.5) {
91      mensagem = "Produto não encontrado. Repor estoque";
92    } else {
93      mensagem = "Produto disponível.";
94    }
95
96    // Publica no tópico MQTT
97    if (client.publish(mqtt_topic, mensagem.c_str())) {
98      Serial.println("Mensagem publicada com sucesso: " + mensagem);
99    } else {
100     Serial.println("Falha ao publicar mensagem.");
101   }
102
103   delay(5000);  // Aguarda 5 segundos antes da próxima leitura
104 }
```

Primeiramente devem ser incluidos as bibliotecas de Wifi e PubSubClient, e configurado a rede wifi e a senha. Deve ser feito a configuração com o servidor MQTT público utilizado (broker.emqx.io) e também o nome do tópico (configurado como widget no aplicativo), onde as mensagens serão publicadas.

A configuração dos pinos dos sensores dependem da conexão feita no controlador/sensor, podendo ser variável. Neste caso foi utilizado os pinos 14 e 25.

Há então a tentativa de conexão via rede Wifi e também do servidor MQTT, que se bem sucedido alerta a mensagem de "Conectado ao Wi-Fi!" e “Conectado ao broker MQTT!”. Caso dê qualquer tipo de falha, será indicado o tipo de erro também e será feito a tentativa de reconexão.

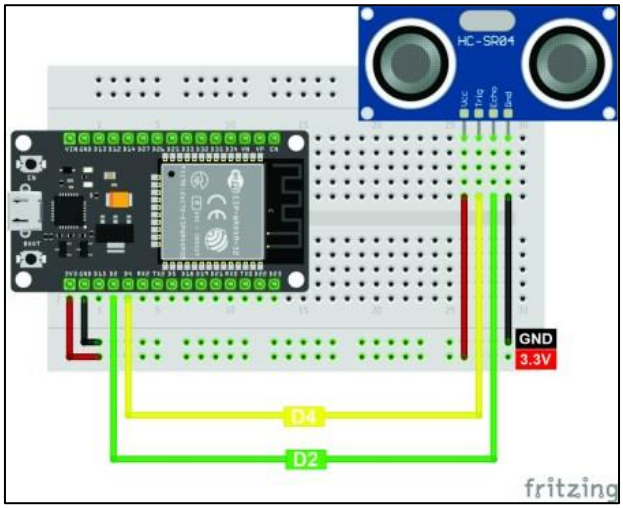
Já com as redes configuradas, então o sensor envia o pulso de emissão e echo e calcula a distância mensurada através do tempo que a informação do pulso é emitida e recebida. Com tal informação, podemos atribuir a variável de distância em cm.

Personalizando o código para o projeto, foi utilizado uma distância de 7.5 centímetros para detectar se o produto se encontra no estoque ou não. Caso a medição dê maior ou igual a 7.5 centímetros, o retorno será em uma string de "Produto não encontrado. Repor estoque", caso seja menor que 7.5, retornará "Produto disponível". Além, será acesso o led na cor branca para gerar o alerta visual, com as mesmas condições da mensagem.

Após isso, a mensagem deve ser enviada no tópico MQTT do Broker criado (via widget de texto). O resultado é a mensagem via aplicativo para informar o indivíduo.

3. Modelo de Montagem

O sensor utilizado foi o HCSR04 que é conectado ao microcontrolador ESP32, que irá processar os dados do estoque. Os cabos de alimentação do sensor são conectados às entradas de 5V e GND do ESP32, enquanto o sinal de saída do sensor é conectado ao pino digital 14 e 25 (Trigger e Echo). Este sistema é alimentado por uma fonte de 5V, adequada para a operação dos componentes eletrônicos envolvidos. Também há a possibilidade de alimentar todo o circuito através de um cabo USB direto de um computador, já que a ESP32 possui entrada para tal. Também foi inserido o led e as conexões com o jumper.



Equipamentos:

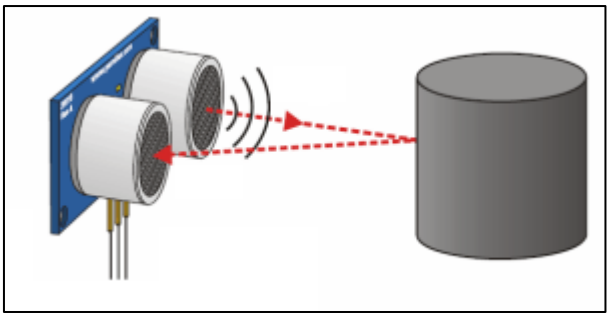
- Computador
- Controlador

- USB
- Sensor
- Jumpers
- Board
- Led branco

Funcionamento:

1. Detecção de Falta de Item:

- **Sensor:** Quando um item é retirado do estoque, o sensor detecta que a distância até o objeto se tornou maior e envia um sinal digital para o ESP32.
- **Microcontrolador ESP32:** Ao receber o sinal do sensor, o ESP32 inicia um processo de verificação.
- **Verificação:** O ESP32 compara o estado atual do sensor com um estado de referência pré-definido. Se houver uma mudança significativa na distância (indicando a retirada de um item), o sistema dispara um alerta.



2. Geração de Alertas:

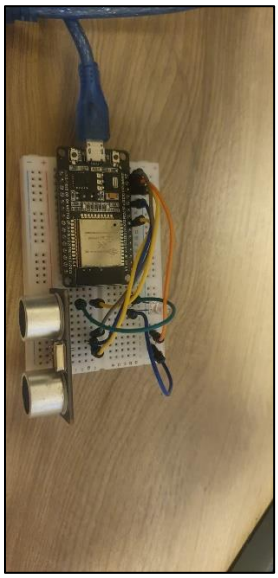
- **Notificação via aplicativo:** O ESP32, utilizando a biblioteca apropriada, envia uma mensagem para o aplicativo MQTT Client previamente configurado, alertando os responsáveis pela reposição sobre a falta do item. A mensagem pode incluir informações adicionais, como a data e hora da detecção.
- **Notificação visual:** O ESP32 irá também gerar o alerta para o LED configurado, que irá acender na cor branca quando o produto não está no estoque.

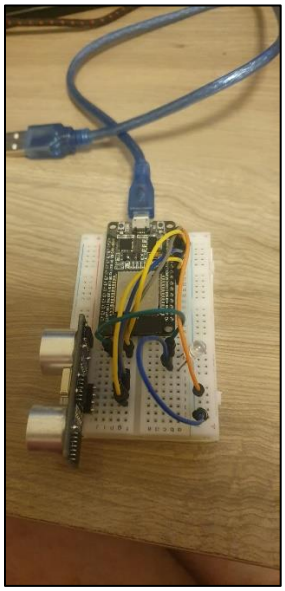
3. Fluxo de Dados:

- **Sensor→ ESP32:** O sensor envia um sinal digital (0 ou 1) para o ESP32, indicando a distância encontrada até o objeto.
- **ESP32→ LED:** Irá enviar o sinal para que o LED seja acesso ou apague dependendo da distância encontrada.
- **ESP32 → Servidor MQTT:** O ESP32 conecta-se ao servidor do Broker estabelecido e envia a mensagem para o grupo configurado.

4. Resultados

Os resultados obtidos foram bem sucedidos com o objetivo proposto no projeto. A seguir as imagens:

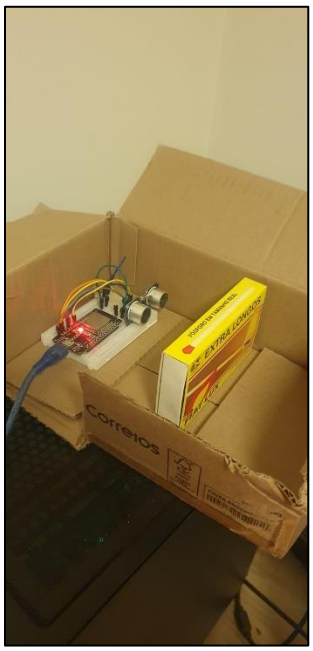




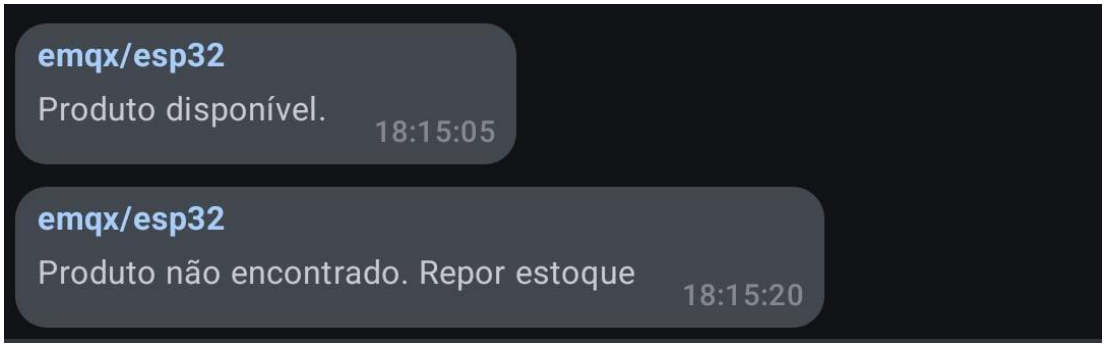
Sem o produto no estoque (não tem a distância minima que o produto estaria).


```
18:07:02.078 -> Conectando ao Wi-Fi...
18:07:02.110 -> Conectado ao Wi-Fi!
18:07:02.141 -> Conectando ao broker MQTT...
18:07:03.498 -> Conectado ao broker MQTT!
18:07:03.530 -> Mensagem publicada com sucesso: Produto não encontrado. Repor estoque
```

Já com o produto no estoque (Exemplo de uma Caixa de fósforos).



```
18:09:29.149 -> Mensagem publicada com sucesso: Produto disponível.
18:09:34.100 -> Reconectando ao broker MQTT...
18:09:34.634 -> Reconectado ao broker MQTT!
18:09:34.667 -> Mensagem publicada com sucesso: Produto disponível.
```



Tempo médio.

Medição do tempo entre o envio da informação pelo sensor até o aplicativo (recebimento dos dados na platafoma MQTT).

Sensor	Aplicativo	Tempo Médio (ms)
18:18:25	**18:18:56**	**31**
18:19:10	**18:19:47**	**37**
18:19:35	**18:20:12**	**23**

Dessa forma foi possivel observer que existe um atraso de aproximadamente 30,3 milisegundos entre a informação disponivel no sensor e enviada ao aplicativo. De modo geral é apresentado um bom resultado na diferença entre tempo, certamente irrisória para o funcionamento e objetivo proposto.

Link para Youtube: https://www.youtube.com/watch?v=Pqo7qzvN42M

Link para Github: https://github.com/Dantekenway/ObjInt_Mackenzie

5. Conclusões

1) Os objetivos propostos foram alcançados?

Os objetivos do projeto foram alcançados com sucesso. O sistema de monitoramento de estoque foi desenvolvido e integrado, permitindo a detecção rápida da ausência de itens e a notificação eficiente dos responsáveis.

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

- Problema: Integração dos diferentes componentes (sensor, ESP32) e comunicação via MQTT. Solução: Uso de bibliotecas específicas para cada componente e protocolo, além de testes e ajustes contínuos durante o desenvolvimento.
- Problema: Configuração do servidor MQTT e envio de notificações via aplicativo. Solução: Configuração cuidadosa do broker MQTT e uso de bibliotecas compatíveis com o ESP32 para o envio de mensagens.

3) Quais são as vantagens e desvantagens do projeto?

Vantagens:

- Eficiência Operacional: Redução do tempo de resposta para reposição de itens em falta.
- Automação: Alinhamento com os princípios da Indústria 4.0, promovendo a automação dos processos de gestão de estoque.
- Notificações Instantâneas: Uso de notificações via aplicativo garante que os responsáveis sejam informados imediatamente.

Desvantagens:

- Complexidade: A implementação pode ser complexa para equipes com pouca experiência em IoT e programação e para estoques com muitos produtos.

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

- Escalabilidade: Adaptar o sistema para gerenciar estoques de maior volume ou múltiplos locais.
- Segurança: Implementar medidas de segurança adicionais para proteger as comunicações e os dados sensíveis.
- Interface do Usuário: Melhorar a interface do usuário no aplicativo de monitoramento para facilitar a visualização e interação.

6. Referências

ENGENHEIRO NO CONTROLE. Como usar ESP32 - Introdução ao módulo e a programação. YouTube, 25 mar. 2020.
Disponível em: https://www.youtube.com/watch?v=J_bUAs-VXA8

BRUNO GARCIA - ARDUINO E CIA. ESP32 para Iniciantes - Como programar passo a passo. YouTube, 2 ago. 2022.
Disponível em: https://www.youtube.com/watch?v=hroFLYLQ8DQ

PROFESSOR ROBSON. ESP32 - Entenda a diferença entre ESP32, ESP8266 e Arduino. YouTube, 22 mar. 2023.
Disponível em: https://www.youtube.com/watch?v=xZ8kKT-DLxk.

CURTO CIRCUITO. Conhecendo o ESP32.
Disponível em: https://curtocircuito.com.br/blog/Categoria%20IoT/conhecendo-esp32?srsltid=AfmBOorJC5l5lMJne_o5JD663CcTMactlGDKkDVfyepI8khLGlEmD9ey.

ASSIS, Pietro Prato de. Desenvolvimento de um sistema automatizado para medição.
Disponível em: https://ifpr.edu.br/londrina/wp-content/uploads/sites/18/2022/07/Pietro-Prato-de-Assis-Desenvolvimento-de-um-sistema-automatizado-para-medicao....pdf