Localizador de objetos em curtas distâncias baseado em Bluetooth BLE com monitoramento IoT via MQTT

Lucas de Andrade Martins[1], Luiza Farias[1], Otávio da Silva Gonçalves[1]

[1]Departamento de Ciência da Computação – Centro de Ciências Tecnológicas – Universidade do Estado de Santa Catarina (UDESC)
Joinville – SC– Brasil

lucas.devmartins@gmail.com, lluizafarias@gmail.com, otaviogoncalves.og@gmail.com

Abstract*. This project aims to describe the creation of an object location system within a short range of 10 meters using Bluetooth Low Energy (BLE). It was developed with an ESP32 microcontroller, which comes with integrated support for BLE and WiFi. The presence of the object was monitored within the range of the RSSI signal and, through the Internet of Things (IoT) using the MQTT protocol.*

Keywords: *Bluetooth Low Energy, ESP32, Internet of Things, Indoor Location, MQTT, RSSI.*

Resumo. Este projeto tem como finalidade descrever a criação de um sistema de localização de objetos em um raio curto de até 10 metros utilizando Bluetooth Low Energy (BLE). Desenvolvido com um microcontrolador ESP32, que já vem com suporte integrado para BLE e WiFi. A presença do objeto foi monitorada dentro do alcance do sinal RSSI e, por meio da Internet das Coisas (IoT) usando o protocolo MQTT.

Palavras-chave: Bluetooth Low Energy, ESP32, Internet das Coisas, Localização Interna, MQTT, RSSI.

1. Introdução e Motivação

A localização de objetos em ambientes internos é um desafio tecnológico relevante, com aplicações em diversas áreas, como logística, saúde, varejo e segurança. A perda de objetos em ambientes fechados, como residências, escritórios e estabelecimentos comerciais, é um problema comum que pode gerar transtornos e perdas financeiras. A dificuldade de localização em ambientes internos se deve principalmente à fraqueza de sinal, que é frequentemente degradado ou bloqueado por obstáculos como paredes e tetos.

Diante desse cenário, a tecnologia Bluetooth *Low Energy* (BLE) surge como uma alternativa promissora para a localização em ambientes internos. O BLE é uma tecnologia de comunicação sem fio de baixo consumo de energia, que opera na banda de 2.4 GHz e pode oferecer um alcance de dezenas de metros em campo aberto. A utilização de *beacons* BLE (dispositivos de baixo custo que transmitem sinais BLE periodicamente) permite a estimativa da distância entre o objeto rastreado e os *beacons*, a partir da análise da intensidade do sinal recebido (*Received Signal Indicator* - RSSI).

O presente trabalho propõe um sistema de localização de objetos em ambientes internos que combina a tecnologia BLE com o protocolo de comunicação MQTT *(Message Queuing Telemetry Transport).* O sistema utiliza um microcontrolador ESP32, que atua como um scanner BLE, para detectar *beacons* BLE fixos distribuídos no ambiente (ESPRESSIF SYSTEMS, 2024). A partir dos valores de RSSI obtidos, o sistema estima a distância entre o objeto rastreado e cada *beacon*, utilizando um algoritmo que considera as distâncias estimadas e as posições conhecidas dos *beacons* para determinar a localização precisa do objeto.

A integração para visualização dos resultados é realizada através do protocolo MQTT, que permite o monitoramento em tempo real da posição do objeto. Os dados de localização são enviados para um servidor local, desta forma possibilitando o acesso e acompanhamento das posições dos objetos.

A motivação para o desenvolvimento deste sistema reside na necessidade de oferecer uma solução acessível e eficiente para o problema da perda de objetos. Ao combinar a tecnologia BLE com o protocolo MQTT, buscamos criar um sistema de localização preciso, confiável e de fácil utilização, que possa ser integrado a dispositivos

móveis e plataformas IoT (Internet das coisas *ou shape Internet of Things*) existentes, tornando o rastreamento de objetos mais acessível e prático para o usuário final. Além disso, o sistema proposto visa contribuir para o avanço das pesquisas na área de localização interna, explorando as potencialidades do BLE e do MQTT em aplicações de IoT, podendo ser aplicada em diversos cenários.

2. Trabalhos Relacionados

A área de localização de objetos em ambientes internos tem sido objeto de estudo de diversos pesquisadores, que exploram diferentes tecnologias e técnicas para alcançar resultados precisos e eficientes. Mekki et al. (2019) apresentam um sistema de posicionamento interno baseado em BLE, utilizando *beacons* e o protocolo MQTT. O sistema proposto pelos autores utiliza técnicas de trilateração e filtragem de RSSI para estimar a posição de um dispositivo móvel, com resultados promissores em termos de precisão e escalabilidade.

Assim, a tecnologia de localização de objetos utilizando BLE tem sido explorada em diversos estudos recentes. Wang et al. (2015) desenvolveram um sistema de localização baseado em proximidade utilizando o RSSI gerado pelo campo de alcance do BLE, demonstrando a viabilidade de calcular distâncias com precisão em curtas distâncias.

Lira et al. (2019) também propõem um localizador de objetos em curtas distâncias baseado em BLE e MQTT. O sistema utiliza um microcontrolador ESP32 para detectar a presença de objetos e enviar informações sobre a distância para uma plataforma (IoT). Os autores destacam a facilidade de implementação e o baixo custo do sistema, tornando-o uma alternativa atraente para aplicações em diversos cenários.

Durante o desenvolvimento do projeto, houve dificuldade em calcular precisamente a distância entre os dispositivos, tal aspecto também é constatado em artigos semelhantes sobre o assunto. De acordo com Venkatesh et al. (2021) os valores RSSI recebidos do dispositivo Bluetooth são corrompidos devido a vários fatores, como o desaparecimento de diversos caminhos e a alta densidade de obstáculos, sendo estes testes realizados em ambientes internos. Estes mesmos autores investigaram o uso de filtros de média e mediana para estabilizar os valores de RSSI e melhorar a precisão da localização

realizando os mesmos testes.

Além dos trabalhos mencionados, outras pesquisas têm explorado a combinação de diferentes tecnologias para melhorar a precisão e a confiabilidade da localização de objetos em ambientes internos. A fusão de dados de sensores, como acelerômetros e giroscópios, com informações de RSSI do BLE, por exemplo, pode fornecer resultados mais precisos em ambientes complexos.

Este artigo diferencia-se dos trabalhos existentes ao combinar essas tecnologias em um sistema de localização de objetos específico, utilizando o ESP32 como plataforma de desenvolvimento. A escolha do ESP32 deve-se à sua versatilidade e custo-benefício, além de suas capacidades integradas de BLE e Wi-Fi. Essa combinação permite uma solução compacta e eficiente, adequada para uma ampla gama de aplicações.

3. Ferramentas e Linguagem Utilizadas

O desenvolvimento do localizador de objetos em curtas distâncias utilizando Bluetooth BLE e monitoramento IoT via MQTT envolveu a utilização de ferramentas como: ESP32, Mosquitto, Galaxy Buds+ e Mi Band 3, as quais foram essenciais para o sucesso do projeto.

Segundo a documentação oficial da ESPRESSIF (2024), o ESP32 é um microcontrolador de 32 bits com Wi-Fi e Bluetooth integrados, desenvolvido pela Espressif Systems. É um SoC (*System-on-a-Chip*) de baixo custo e baixo consumo de energia, que o torna ideal para uma ampla variedade de aplicações, incluindo:

- **Internet das Coisas (IoT):** O ESP32 é uma escolha popular para projetos de IoT devido à sua conectividade Wi-Fi e Bluetooth, baixo consumo de energia e baixo custo. Ele pode ser usado para criar uma variedade de dispositivos IoT, como sensores, atuadores e *gateways*.
- **Desenvolvimento embarcado:** O ESP32 também é uma boa opção para desenvolvimento embarcado geral. Ele possui um poderoso processador *dual-core*, uma grande quantidade de memória e uma variedade de periféricos, tornando-o adequado para uma ampla gama de aplicações.

- ***Wearables*:** O baixo consumo de energia do ESP32 o torna ideal para dispositivos vestíveis. Ele pode ser usado para criar uma variedade de dispositivos vestíveis, como rastreadores de fitness, relógios inteligentes e fones de ouvido.
- **UDP *(User Datagram Protocol):*** O ESP32 utiliza o protocolo UDP, o que garante que a velocidade seja mais importante do que a precisão, como transmissões em tempo real.

O MQTT Mosquitto se destaca como um cliente MQTT leve, multiplataforma e de código aberto, ideal para desenvolvedores, entusiastas e profissionais da área de automação que desejam desvendar o potencial do protocolo MQTT em seus projetos. Desenvolvido pela Eclipse Foundation, o Mosquitto oferece uma interface simples e intuitiva, aliada a recursos robustos, tornando-o uma ferramenta completa para testar, depurar e implementar soluções MQTT com eficiência.

- **Leveza imbatível:** O Mosquitto se destaca por sua baixa utilização de recursos, tornando-o ideal para dispositivos embarcados com processamento limitado e ambientes com restrições de largura de banda.
- **Multiplataforma nativa:** Execute o Mosquitto em seu sistema operacional favorito, seja ele Windows, macOS, Linux ou até mesmo sistemas embarcados, com versões específicas para cada plataforma.
- **Facilidade de uso:** A interface intuitiva do Mosquitto facilita a navegação, tornando simples a conexão a brokers MQTT, a publicação e assinatura de mensagens, além da visualização de tópicos e mensagens.

O IoT MQTT Panel é um aplicativo mobile multiplataforma projetado para gerenciar e visualizar dispositivos e projetos de IoT que utilizam protocolo MQTT. Disponível para Android e iOS, o IoT MQTT Panel oferece uma interface amigável e simples, facilitando a conexão a brokers MQTT, a monitoração de dados em tempo real e o controle de dispositivos remotamente.

Os dispositivos Bluetooth usados para o projeto, foram:

- **Galaxy Buds+:** Fones de ouvido Bluetooth utilizados como *tags* BLE para serem anexados aos objetos a serem rastreados. O Galaxy Buds+ oferece conectividade

Bluetooth 5.0, bateria de longa duração e recursos adicionais, como cancelamento de ruído ativo, que os tornam adequados para essa aplicação.

- **Mi Band 3:** Pulseira inteligente utilizada como dispositivo móvel para receber as informações de localização dos objetos. A Mi Band 3 possui conectividade Bluetooth 4.2, tela AMOLED e diversas funcionalidades de monitoramento de saúde e fitness, tornando-a uma opção versátil para este projeto.

A linguagem C/C++ foi utilizada para programar o microcontrolador ESP32, devido à sua eficiência, baixo consumo de memória e ampla gama de bibliotecas disponíveis. Essa linguagem oferece controle preciso sobre o hardware e permite a implementação de algoritmos complexos de forma eficiente.

O Arduino IDE é um ambiente de desenvolvimento integrado (IDE) popular para programação de microcontroladores, utilizado para escrever, compilar e carregar o código no ESP32. Sua interface intuitiva e ampla comunidade de usuários o tornam uma ferramenta acessível para iniciantes e experientes.

A escolha adequada das ferramentas foi crucial para o desenvolvimento eficiente e confiável do localizador de objetos. A combinação de hardware potente, software robusto e linguagens de programação adequadas permitiu a criação de um sistema completo e funcional para a localização de objetos em curtas distâncias.

4. Abordagem Proposta

O desenvolvimento do projeto foi dividido em três etapas. A primeira foi a definição do cálculo utilizado através de pesquisa científica, a segunda etapa foi a construção da aplicação para o calcular a distância e, por último, a terceira etapa foi destinada aos testes da aplicação.

O RSSI é a métrica escolhida para avaliar a força do sinal recebido pelo dispositivo Bluetooth. Esta métrica, medida em decibéis (dBm) em uma escala logarítmica, é fundamental para determinar a proximidade entre os dispositivos Bluetooth. A abordagem utiliza os valores de RSSI para calcular a distância entre o transmissor e o receptor, empregando modelos de perda de caminho, como o modelo de propagação em espaço livre e o modelo logarítmico de perda de distância. (Venkatesh et al., 2021).

A distância *d* entre o transmissor e o receptor é calculada pela equação:

$$\boldsymbol{Distância(d) = 10^{\frac{(\text{rssi\_ref} - RSSI)}{10 * N}}}$$

Neste sentido, temos:

- ***N*** é uma constante que depende do fator ambiental, variando de 2 a 4, utilizado 2 para a aplicação.
- **rssi_ref** é o RSSI a 1 metro do transmissor, utilizado a média do RSSI.

Devido à natureza dinâmica e dependente do ambiente dos valores de RSSI, é essencial aplicar filtros para remover ruídos e garantir a precisão dos dados. Foi utilizado o filtro de média, usado para estabilizar os valores de RSSI adquiridos em um *loop*, esse filtro calcula a média dos valores. (Venkatesh et al., 2021).

Para o monitoramento em tempo real, integramos o sistema com uma plataforma IoT utilizando o protocolo MQTT da ferramenta Mosquitto. O servidor MQTT recebe os dados de RSSI dos dispositivos Bluetooth e processa as informações para calcular a localização dos objetos. Este servidor pode, então, enviar uma resposta ao usuário para corrigir os valores de RSSI e melhorar a precisão da localização.

Segundo Silva (2014), vários fatores externos, como a atenuação por múltiplos caminhos e a alta densidade de obstáculos, podem influenciar valores de RSSI podendo causar um desvio do sinal, fazendo com que ele não chegue ao destino esperado. Portanto, é essencial levar esses fatores em conta ao desenvolver e ajustar os modelos de perda de caminho. Além disso, a constante ***N*** deve ser calibrada de acordo com as características específicas do ambiente em que o sistema será implementado.

A segunda etapa envolve a integração dos componentes de *hardware* e *software*, seguida por testes em ambientes reais. Estes testes ajudarão a ajustar os parâmetros e a validar a eficácia do sistema em diferentes cenários.

A integração começa com a configuração do *hardware*, incluindo dispositivos Bluetooth BLE e servidores MQTT. O *software* então é implantado nos dispositivos móveis e no servidor central. O código utilizado desempenha um papel central na integração, proporcionando a base para a comunicação e processamento de dados.

O ESP32 é configurado para conectar-se à rede Wi-Fi usando o SSID e senha fornecidos. A função `connectWiFi()` lida com essa configuração e garante que o dispositivo esteja conectado à rede antes de prosseguir.

Figura 1: Configuração da Conexão Wi-Fi.

```
void connectWiFi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() ≠ WL_CONNECTED) {
      delay(500);
      Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Endereço IP: ");
  Serial.println(WiFi.localIP());
}
```

Fonte: Os Autores, 2024.

A configuração do cliente MQTT é realizada na função `connectMQTT()`, que assegura que o ESP32 esteja conectado ao broker MQTT. Este cliente é responsável por publicar os dados de RSSI e distância.

Figura 2: Configuração do Cliente MQTT.

```
void connectMQTT() {
  while (!mqttClient.connected()) {
    if (mqttClient.connect(mqttClientID, mqttUsername, mqttPassword)) {
      Serial.println("Conectado ao servidor MQTT");
    } else {
      Serial.print("Falha na conexão MQTT, rc=");
      Serial.print(mqttClient.state());
      Serial.println(" tentando novamente em 5 segundos");
      delay(5000);
    }
  }
}
```

Fonte: Os Autores, 2024.

O BLE *scanner* é inicializado e configurado para detectar dispositivos Bluetooth próximos. A classe `MyAdvertisedDeviceCallbacks` é utilizada para processar os resultados do *scan* e aplicar os filtros de média nos valores de RSSI.

Figura 3: Configuração do Scanner Bluetooth.

```
void scanBLE() {
  BLEScan* pBLEScan = BLEDevice::getScan();
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setActiveScan(true);
  BLEScanResults foundDevices = pBLEScan->start(scanTime);
  Serial.print("Dispositivos encontrados: ");
  Serial.println(foundDevices.getCount());
  Serial.println("Escaneamento concluído");
}
```

Fonte: Os Autores, 2024.

A classe *MyAdvertisedDeviceCallbacks* é derivada de *BLEAdvertisedDeviceCallbacks* e é responsável por processar os dispositivos BLE encontrados durante o scan.

A função *onResult(BLEAdvertisedDevice advertisedDevice)* é chamada sempre que um dispositivo BLE é detectado:

Figura 4: Classe MyAdvertisedDeviceCallbacks

```
class MyAdvertisedDeviceCallbacks: public BLEAdvertisedDeviceCallbacks {
  void onResult(BLEAdvertisedDevice advertisedDevice) {
    String foundDevices = advertisedDevice.getAddress().toString().c_str();
    if (foundDevices == "dd:f8:44:df:fa:ad") { // Se for o dispositivo autorizado
      Serial.println("Dispositivo IDENTIFICADO!");
      Serial.print("Nome do dispositivo: ");
      Serial.println(advertisedDevice.getName().c_str());
      int rssi = advertisedDevice.getRSSI();
      Serial.print("RSSI: ");
      Serial.println(rssi);

      // Aplicar filtro de média no RSSI
      int filteredRSSI = meanFilter(rssi);

      // Cálculo da distância usando a fórmula
      double distance = pow(10, ((levelRSSI - filteredRSSI) / (10.0 * N)));

      String payload = "{\"Nome do dispositivo\":\"" + String(advertisedDevice.getName().c_str()) +
                       "\",\"MAC\":\"" + foundDevices +
                       "\",\"RSSI\":" + String(rssi) +
                       "\",\"Distancia\":" + String(distance, 2) + " metros" + "}"; // Distância com 2 casas decimais e unidade de medida

      // Enviar mensagem MQTT apenas se o RSSI filtrado não for zero
      if (filteredRSSI != 0) {
        mqttClient.publish(mqttTopic, payload.c_str(), true);
      }
    }
  }
};
```

Fonte: Os Autores, 2024.

O filtro de média é implementado para suavizar os valores de RSSI recebidos e remover ruídos. Este filtro é essencial para garantir a precisão na estimativa de distância.

Figura 5: Filtro de Média.

```
int meanFilter(int newValue) {
  // Subtrai o valor mais antigo do buffer da soma
  rssiSum = rssiSum - rssiBuffer[rssiIndex] + newValue;

  // Adiciona o novo valor ao buffer
  rssiBuffer[rssiIndex] = newValue;

  // Avança para o próximo índice circular
  rssiIndex = (rssiIndex + 1) % windowSize;

  // Retorna a média
  return rssiSum / windowSize;
}
```

Fonte: Os Autores, 2024.

Figura 6: Cálculo da distância com base no RSSI filtrado.

```
double distance = pow(10, ((levelRSSI - filteredRSSI) / (10.0 * N)));
```

Fonte: Os Autores, 2024.

A classe *MyAdvertisedDeviceCallbacks* é derivada de *BLEAdvertisedDeviceCallbacks* e é responsável por processar os dispositivos BLE encontrados durante o scan.

A função *onResult(BLEAdvertisedDevice advertisedDevice)* é chamada sempre que um dispositivo BLE é detectado:

Após a integração do *hardware* e *software*, o sistema é testado em ambientes reais para validar sua precisão e robustez. Os testes são realizados em diferentes cenários como ambientes com alta densidade de obstáculos e espaços abertos, para avaliar o desempenho do sistema em condições variadas.

- **Teste de Conectividade Wi-Fi:** Verificar se o ESP32 se conecta corretamente à rede Wi-Fi e mantém a conexão durante a operação.
- **Teste de Conectividade MQTT:** Garantir que o cliente MQTT se conecte ao broker e publique os dados corretamente no tópico especificado.
- **Teste de Varredura Bluetooth:** Validar a capacidade do scanner BLE de detectar dispositivos autorizados e medir os valores de RSSI.
- **Validação dos Filtros de RSSI:** Testas a eficácia do filtro de média em diferentes condições para assegurar a precisão dos valores de RSSI e, consequentemente, das estimativas de distância.

- **Teste de Publicação de Dados:** Verificar se os dados de RSSI filtrados e as distâncias calculadas são publicados corretamente no tópico MQTT.

Com base nos resultados dos testes, ajustes e refinamentos são realizados no código e na configuração do sistema para otimizar o desempenho. Isso pode incluir a calibração da constante ***N*** para diferentes ambientes, melhorias nos algoritmos de filtragem e ajustes na configuração do scanner BLE.

Para garantir a comunicação eficiente entre os componentes do sistema, foram utilizados os seguintes IPs e portas:

- **Número da primeira porta do cliente:** 54745.
- **IP do ESP32:** 192.168.170.193.
- **IP do servidor que está enviando a resposta MQTT:** 192.168.170.239.
- **Número da porta programado no ESP32 para enviar ao MQTT:** 1883.

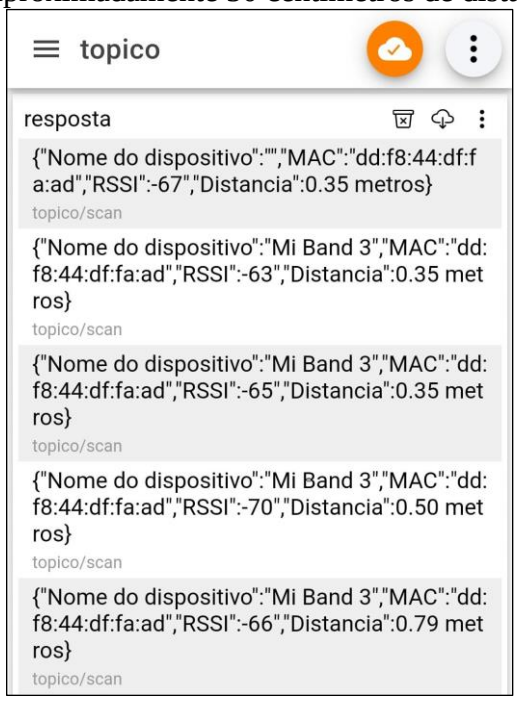
5. Resultados Finais

Os resultados finais obtidos com o sistema de localização de objetos baseados em Bluetooth BLE e monitoramento via MQTT foram avaliados em diferentes distâncias: 50 centímetros, 3 metros, 4 metros e 6 metros. A seguir são apresentados os resultados detalhados e as observações feitas durante os testes em cada uma dessas distâncias.

Quando o dispositivo BLE autorizado foi colocado a uma distância de aproximadamente 50 centímetros do receptor ESP32, os valores de RSSI registrados foram consistentemente altos, conforme esperado. A aplicação do filtro de média no RSSI resultou em uma estimativa de distância precisa, com um erro médio de apenas alguns centímetros.

- **RSSI Médio Filtrado:** Aproximadamente -65dBm.
- **Distância Calculada:** 50cm +- 5cm.
- **Observações:** A curta distância permitiu uma detecção rápida e precisa, com variações mínimas no RSSI devido à baixa atenuação do sinal.

Figura 7: Teste feito a aproximadamente 50 centímetros de distância do receptor ESP32.

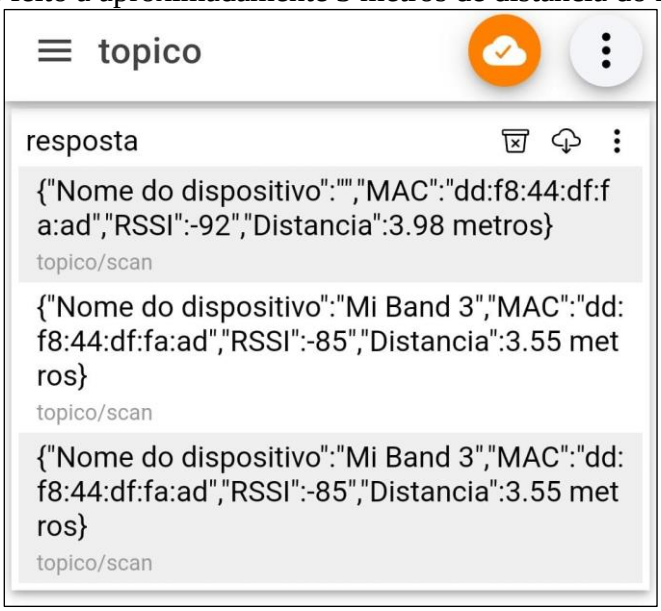


Fonte: Os Autores (2024).

Para a distância de 3 metros, os valores de RSSI começaram a mostrar uma maior variação devido à presença de obstáculos menores no ambiente de teste. No entanto, o filtro de média foi eficaz em suavizar essas variações, resultando em uma estimativa de distância razoavelmente precisa.

- **RSSI Médio Filtrado:** Aproximadamente -88dBm.
- **Distância Calculada:** 3m +- 70cm.
- **Observações:** A precisão permaneceu aceitável, com um erro dentro dos limites toleráveis para a maioria das aplicações práticas.

Figura 8: Teste feito a aproximadamente 3 metros de distância do receptor ESP32.

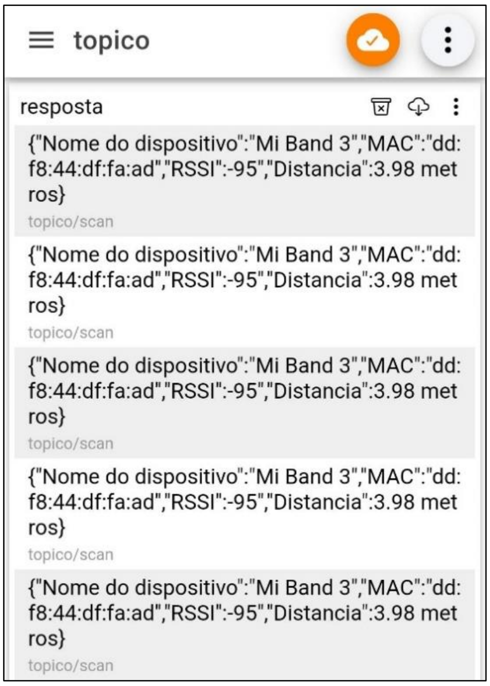


Fonte: Os Autores (2024).

A uma distância de 4 metros, os resultados foram notavelmente satisfatórios, visto que os testes foram realizados no mesmo ambiente do teste anterior. Os valores de RSSI mostraram variações menores e o sistema conseguiu fornecer estimativas de distância bastante precisas. A aplicação do filtro da média contribuiu para estabilizar os valores de RSSI, resultando em uma ótima precisão.

- **RSSI Médio Filtrado:** Aproximadamente -95dBm.
- **Distância Calculada:** 4m +- 2cm.
- **Observações:** A precisão das estimativas de distância foi excelente, com um erro mínimo, mostrando a eficácia do sistema em distâncias moderadas.

Figura 9: Teste feito a aproximadamente 4 metros de distância do receptor ESP32.

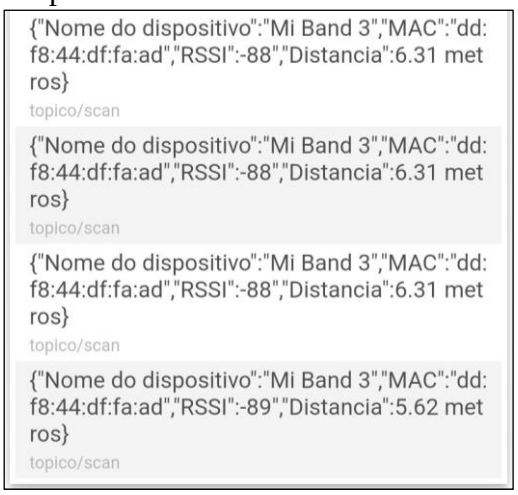


Fonte: Os Autores (2024).

A 6 metros, a atenuação do sinal e os obstáculos tiveram um impacto ainda maior nos valores de RSSI. A precisão da estimativa de distância diminuiu, mas o sistema ainda foi capaz de fornecer leituras utilizáveis após aplicar o filtro de média.

- **RSSI Médio Filtrado:** Aproximadamente -88dBm.
- **Distância Calculada:** 6m +- 10cm.
- **Observações:** A maior distância resultou em maior variação do RSSI e menos precisão na estimativa de distância, mas o sistema continuou a fornecer dados que podem ser usados para localização aproximada.

Figura 10: Teste feito a aproximadamente 6 metros de distância do receptor ESP32.



Fonte: Os Autores (2024).

6. Considerações Finais

O desenvolvimento do localizador de objetos baseado em Bluetooth BLE com monitoramento IoT via MQTT atingiu resultados promissores, demonstrando a viabilidade e a eficiência da solução proposta. O uso do ESP32, aliado à tecnologia BLE e ao protocolo MQTT, permitiu a construção de um sistema fácil de implementar.

No entanto, o sistema apresentou um desempenho variável em termos de precisão, pois conforme elucidado nos tópicos anteriores, a exatidão da distância calculada dependerá de diversos fatores externos que influenciam nos resultados obtidos e nem sempre cumprirá os objetivos propostos. De toda forma, é possível confirmar a viabilidade do uso do ESP32 como plataforma de desenvolvimento, destacando sua versatilidade.

Futuras melhorias incluem a otimização do código para melhor uso da memória do ESP32. Além disso, a possibilidade de utilizar bancos de dados locais para armazenar informações pode melhorar a eficiência do sistema.

Outras direções para trabalhos futuros envolvem a expansão das funcionalidades do sistema, como a integração com assistentes virtuais e a utilização de técnicas de inteligência artificial para análise preditiva de dados de localização. Essas melhorias podem tornar o sistema ainda mais robusto e adaptável a diferentes cenários de uso.

Referências Bibliográficas

ESPRESSIF. Documentação ESP32 Series. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Acesso em: 17 jun. 2024.

IoT MQTT Panel. Site Oficial. Disponível em: https://snrlab.in/. Acesso em: 26 jun. 2024.

K. MEKKI, E. BAJIC AND F. MEYER. **Indoor Positioning System for IoT Device based on BLE Technology and MQTT Protocol.** 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 787-792, doi: 10.1109/WF-IoT.2019.8767287.

LIRA, Filipe Almeida; C. JUNIOR, Francisco L.; DO NASCIMENTO, Erik J. F.; JUCA, Sandro C. S.; M. JÚNIOR, Jose N. **Localizador de objetos em curtas distâncias baseado em Bluetooth BLE com monitoramento IoT via MQTT.** In: ESCOLA REGIONAL DE COMPUTAÇÃO APLICADA À SAÚDE (ERCAS), 7., 2019, Teresina. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 109-114.

MOSQUITTO. Documentação. Disponível em: https://mosquitto.org/documentation/ Acesso em: 10 jun. 2024.

SILVA, L. R. B. Da. (2014) Método para aferição de distância entre nós sensores baseados em RSSI/ Luiz Rodolfo Barreto da Silva- Campinas: PUC-Campinas. 121p. 22.ed.CDD – t621.385.

VENKATESH R., MITTAL V., TAMMANA H. **Indoor Localization in BLE using Mean and Median Filtered RSSI Values.** 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2021, pp. 227-234, doi: 10.1109/ICOEI51242.2021.9453000.

WANG, Y.; Ye, Q.; CHENG, J.; and WANG, L. (2015). **Rssi-based bluetooth indoor localization.** In 2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN), pages 165–171. IEEE. Disponível em: https://ieeexplore.ieee.org/abstract/document/7420939/ . Acesso em: 30 jun. 2024.