Desenvolvimento de um Localizador de Objetos – Integrando Tecnologias BLE e MQTT para Detectar e Estimar Distâncias em Ambientes Reais

Camille da Veiga[1], Yasmin Fadel[1]

[1]Universidade do Estado de Santa Catarina (UDESC) - CCT
Joinville – SC – Brasil

camille.dv@edu.udesc.com.br, yasmin.fadel15@edu.udesc.com.br

Abstract. *The rise of Bluetooth Low Energy (BLE) has provided opportunities for several technological advances, one of the possible applications being indoor location. This article presents the development of a BLE (Bluetooth Low Energy) device detection device using the ESP32 microcontroller within a radius of 10 meters. The project covers hardware configuration, scanning of BLE devices and distance estimation based on Received Signal Strength Indication (RSSI) values using the Message Queuing Telemetry Transport (MQTT) protocol. The system was designed to detect BLE devices, estimate their distances and send the data via topics (with the MQTT protocol) to the MQTT Explorer interface. Throughout this article, the system, the techniques used and the experiments carried out are described, together with the results obtained.*

Resumo. *A ascensão do Bluetooth Low Energy (BLE), proporcionou oportunidades para diversos avanços tecnológicos, sendo uma das possíveis aplicações a localização interna. Este artigo apresenta o desenvolvimento de um dispositivo de detecção de dispositivos BLE (Bluetooth Low Energy) utilizando o microcontrolador ESP32 num raio de 10 metros. O projeto abrange a configuração do hardware, o escaneamento de dispositivos BLE e a estimativa da distância baseada em valores de Received Signal Strength Indication (RSSI) com a utilização do protocolo Message Queuing Telemetry Transport (MQTT). O sistema foi projetado para detectar dispositivos BLE, estimar suas distâncias e enviar os dados através de tópicos (com o protocolo MQTT) para a interface do MQTT Explorer. Ao longo deste artigo, é descrito o sistema, as técnicas utilizadas e os experimentos realizados, juntamente com os resultados obtidos.*

1. Introdução

O avanço das tecnologias de comunicação sem fio tem impulsionado o desenvolvimento de dispositivos inteligentes capazes de interagir de maneira dinâmica com o ambiente ao seu redor. O *Bluetooth Low Energy* (BLE) destaca-se como uma dessas tecnologias, sendo amplamente empregada em aplicações que demandam baixo consumo de energia e comunicação eficiente. O microcontrolador ESP32, com seu módulo *Bluetooth*

integrado, oferece uma plataforma versátil para a criação de soluções de detecção e monitoramento BLE (Alves, 2019).

Este estudo tem como objetivo desenvolver um dispositivo utilizando o ESP32 para detectar dispositivos BLE, calcular suas distâncias e transmitir essas informações via MQTT. A aplicação desse dispositivo é vasta, abrangendo desde monitoramento ambiental até automação residencial e sistemas de segurança.

O desenvolvimento foi estruturado em três etapas principais. Primeiramente, a configuração do *hardware*, garantindo a funcionalidade do ESP32 e de seu módulo *Bluetooth*. Em seguida, foi realizada a configuração do *software* necessário para escanear dispositivos BLE e estimar suas distâncias com base nos valores de RSSI, utilizando uma média para melhor precisão. Por fim, integrou-se a comunicação MQTT, permitindo que o dispositivo envie dados sobre os dispositivos BLE detectados para um *broker* MQTT, facilitando a análise e o monitoramento remoto.

A implementação bem-sucedida deste projeto demonstra a viabilidade do uso do ESP32 em aplicações de detecção BLE, fornecendo uma solução eficiente e de baixo custo. Este artigo detalha cada etapa do desenvolvimento, discutindo os desafios enfrentados, as soluções adotadas e os resultados obtidos, contribuindo significativamente para o avanço das tecnologias de comunicação e detecção sem fio.

2. Trabalhos Relacionados

Este trabalho baseia-se em diversos estudos relacionados ao uso de tecnologias de comunicação sem fio para localização de objetos e estimativa de distâncias. A seguir, é apresentada uma análise dos trabalhos relevantes que contribuíram para a fundamentação deste estudo.

O artigo escrito por Lira et al. (2019) oferece uma visão do desenvolvimento de um sistema que utiliza *Bluetooth Low Energy* (BLE) para localizar objetos em curtas distâncias, com monitoramento em tempo real através do protocolo IoT MQTT. A pesquisa mostrou que o BLE é eficaz para rastreamento em ambientes internos, com precisão na detecção de proximidade. Este sistema é útil para aplicações domésticas e comerciais que necessitam de rastreamento preciso.

O estudo de Alves et al. (2019) demonstra a detecção de proximidade de dispositivos usando WiFi ESP8266, com base na intensidade do sinal recebido (RSSI). Apesar da pesquisa focar no *WiFi* e utilizar outro modelo de ESP, os métodos podem ser aplicados no uso do BLE, contribuindo para o desenvolvimento teórico de sistemas de localização com essa tecnologia.

Os trabalhos mencionados acima contribuem para a fundamentação do presente estudo, que visa estimar distâncias em ambientes reais. Ao unir essas abordagens, espera-se desenvolver um sistema de rastreamento que seja aplicável a uma ampla gama de cenários.

3. Estudo das Tecnologias e *Hardware*

Para a implementação do projeto, é importante compreender as ferramentas e o *hardware* que serão utilizados ao longo do desenvolvimento do projeto. Cada componente foi escolhido pela sua capacidade de atender às necessidades específicas de comunicação e processamento do sistema. Nesta seção, será apresentado os principais elementos

envolvidos, a começar pelo microcontrolador ESP32, passando pela plataforma de desenvolvimento Arduino IDE e a ferramenta de visualização e gerenciamento de mensagens MQTT *Explorer*, até o aplicativo NRF *Connect* para a interação com os dispositivos BLE para testes e execução da aplicação. Esse estudo proporciona uma base sólida para a construção e otimização do localizador, a fim de garantir que cada parte do sistema funcione de maneira integrada e eficiente.

3.1 ESP32

O ESP32 é um microcontrolador integrado com conectividade *Wi-Fi* e *Bluetooth*, projetado pela *Espressif Systems*. Este é comumente utilizado em projetos de IoT (*Internet* das Coisas) devido à sua versatilidade, potência e baixo custo. Equipado com um processador *dual-core* de 32 bits, o ESP32 oferece um ótimo desempenho para uma diversidade de aplicações (Blog da Curto, 2018). Seu módulo BLE (*Bluetooth Low Energy*) permite comunicação eficiente com dispositivos *Bluetooth*, essencial para o desenvolvimento de localizadores de objetos. Além disso, o ESP32 possui diversos periféricos integrados, como GPIOs, ADCs, DACs, e interfaces de comunicação como SPI, I2C e UART, embora para este projeto seja utilizado apenas suas funcionalidades de *Wi-Fi* e BLE (UsinaInfo, 2024).

3.2 Protocolo MQTT

O protocolo MQTT (*Message Queuing Telemetry Transport*), desenvolvido inicialmente pela IBM, tornou-se um padrão aberto para comunicação entre dispositivos, conforme estabelecido pelo OASIS em 2014. A arquitetura do MQTT é baseada no modelo de publicação e assinatura (*publish-subscribe*). Nesse modelo, os dispositivos enviam (*publish*) informações a um servidor intermediário, conhecido como *broker*. O *broker*, por sua vez, retransmite essas informações para os clientes interessados (*subscribers*). O model*o publish-subscribe* do MQTT é apresentado na Figura 1.

A Figura 1 ilustra um cenário típico de comunicação em uma rede de mensagens MQTT. Esta mostra um *broker* MQTT central conectado a vários clientes, incluindo um sensor, um dispositivo móvel, um *desktop* e um *laptop*. A imagem destaca o fluxo de mensagens bidirecional entre o *broker*, que atua como intermediário, e os clientes. As setas azuis representam as mensagens *publish* (publicadas) pelos clientes, enquanto as setas verdes representam as mensagens *subscribe* (assinadas) pelos clientes. As linhas tracejadas indicam que um cliente pode se inscrever em um tópico e receber mensagens de outros clientes que publicam nesse tópico. Esse cenário demonstra a capacidade do MQTT de conectar dispositivos heterogêneos.

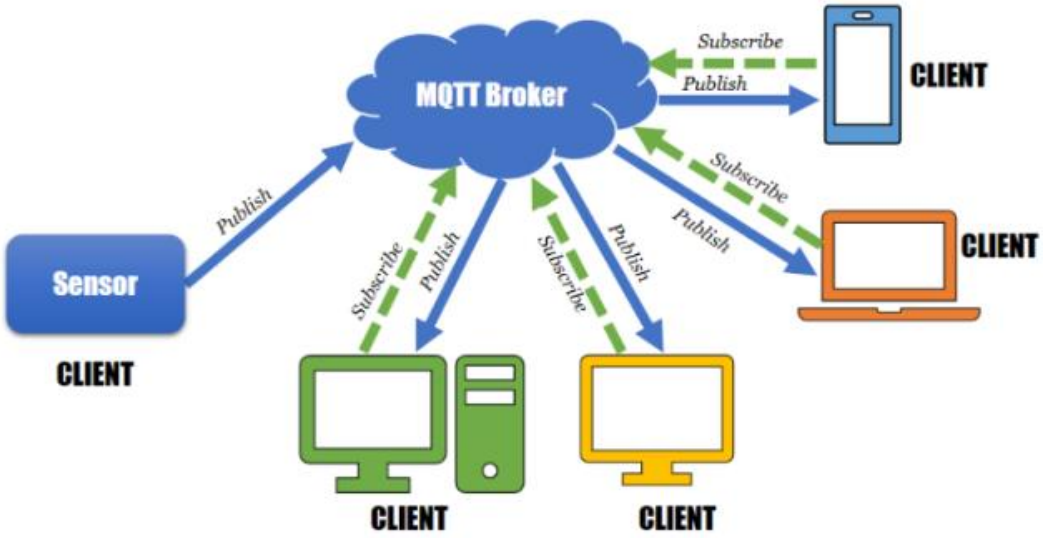

Figura 1. O Modelo *Publish/Subscribe* do MQTT. Fonte: (EG Innovations, 2019).

O MQTT opera na camada de aplicação da arquitetura TCP/IP, conforme ilustrado na Figura 2. Este define o modelo de operação entre os dispositivos, especificando os papéis de cada um, o formato das mensagens e a sequência de comunicação. A infraestrutura de rede é suportada pelas camadas subjacentes, incluindo o protocolo TCP, que assegura a entrega confiável dos dados, e o protocolo IP, que permite a transmissão dos dados pela Internet (Amazon, 2024).

A Figura 2 apresenta um diagrama que demonstra as camadas de um modelo de rede de comunicação, com seus protocolos correspondentes. A camada de aplicação representa a interface entre o usuário e a rede, com protocolos como MQTT para a comunicação entre dispositivos. Abaixo, a camada de transporte fornece serviços de entrega de dados confiáveis, com o TCP. A camada de rede gerencia o roteamento de dados, utilizando protocolos como IPv4 e IPv6. Por fim, a camada de rede física define as interfaces físicas para a comunicação, abrangendo tecnologias como *Ethernet*, *Wi-Fi*, etc. O diagrama ilustra a estrutura hierárquica da comunicação de rede, com cada camada construindo sobre a anterior para garantir uma transferência de dados eficiente e confiável.

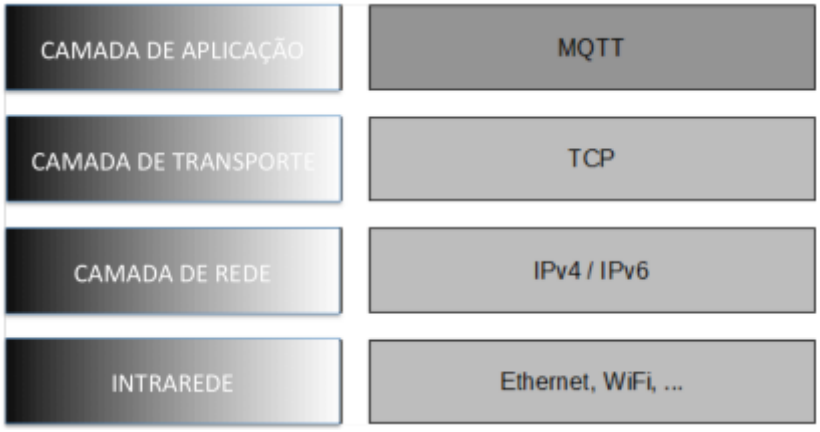

Figura 2. Distribuição dos Protocolos nas Camadas. Fonte: (Conceição; Costa, 2019).

Uma característica importante do MQTT é a possibilidade de trabalhar com o protocolo UDP, além do TCP. Embora o UDP não garanta a entrega dos dados, este requer menos controle sobre a transmissão, tornando-o adequado para dispositivos com recursos limitados ou para aplicações na qual a latência causada pela retransmissão não é aceitável (Conceição; Costa, 2019).

O protocolo IP permite que os dados transmitidos pelo MQTT sejam enviados pela *Internet*, possibilitando que dispositivos e *brokers* comuniquem-se de qualquer lugar com acesso à rede. O *broker*, em particular, pode ser oferecido como um serviço hospedado em um servidor remoto na Internet. Para o artigo atual, é utilizado o *broker* Mosquitto.

3.3 *Broker* Mosquitto

Mosquitto é um *broker* MQTT desenvolvido pela *Eclipse Foundation* e é um *software* de código aberto, que oferece uma implementação leve e eficiente do protocolo MQTT. Este, define claramente os papéis de cada dispositivo na rede, o formato das mensagens e a sequência de comunicação. Ele opera na camada de aplicação da arquitetura TCP/IP e se apoia também no protocolo TCP, que permite a transmissão dos dados pela *Internet*. Pode, também, operar sobre o protocolo UDP para aplicações que exigem menor controle de transmissão e menor latência (EngProcess, 2018).

Conforme EngProcess (2018), o Mosquitto costuma ser utilizado em automação residencial e atuadores e controladores para criar ambientes inteligentes e integrados. Também é ideal para redes de sensores distribuídos, permitindo que dispositivos espalhados por grandes áreas comuniquem dados de forma eficiente e confiável.

Em resumo, o Mosquitto é um *broker* MQTT eficiente e seguro, adotado em diversos projetos de IoT. Sua flexibilidade, facilidade de uso e capacidade de escalar para grandes implementações fazem dele uma escolha preferida tanto por entusiastas quanto por profissionais. Com suporte contínuo da comunidade e da *Eclipse Foundation*, Mosquitto continua a evoluir, oferecendo uma plataforma confiável para a comunicação de dispositivos no ecossistema IoT.

3.4 Arduino IDE

O Arduino IDE (*Integrated Development Environment*) é uma plataforma de desenvolvimento de código aberto utilizada para a programação de microcontroladores, incluindo o ESP32. Projetada para ser acessível, a Arduino IDE oferece uma interface intuitiva que simplifica o processo de escrita, compilação e *upload* de código para os dispositivos. A IDE suporta uma variedade de bibliotecas e exemplos que facilitam a integração de sensores, atuadores e módulos de comunicação, como *Wi-Fi* e *Bluetooth*. No contexto do atual projeto, o Arduino IDE permite a conexão e compilação do código no ESP32, utilizando suas funcionalidades de *Wi-Fi* e BLE para garantir a comunicação e o rastreamento dos objetos (ArduinoBLE, 2024).

3.5 MQTT *Explorer*

O MQTT *Explorer* é uma ferramenta gráfica para a visualização e o gerenciamento de mensagens MQTT (*Message Queuing Telemetry Transport*), um protocolo de comunicação, geralmente utilizado em aplicações de IoT (Internet das Coisas). Esta ferramenta permite conectar-se a um *broker* MQTT, explorar tópicos, publicar e subscrever mensagens de maneira intuitiva. No contexto do atual projeto, o MQTT *Explorer* é fundamental para monitorar e gerenciar a comunicação entre o ESP32 e o servidor, permitindo a visualização em tempo real dos dados transmitidos e recebidos (Singh, 2015).

3.6 NRF *Connect*

O NRF *Connect* é uma aplicação móvel desenvolvida pela *Nordic Semiconductor* para dispositivos *Android* e iOS, projetada para interagir com dispositivos *Bluetooth Low Energy* (BLE). Esta ferramenta permite a exploração, análise e depuração de conexões BLE, tornando-se essencial para o desenvolvimento e teste de projetos que utilizam essa tecnologia. Com o NRF *Connect*, é possível escanear dispositivos BLE próximos, conectar-se a eles, explorar seus serviços e características, ler e escrever dados, além de monitorar a comunicação em tempo real. No contexto do atual projeto, o NRF *Connect* é utilizado para emitir um sinal BLE do dispositivo no qual está instalado. Além disso, sua funcionalidade de escaneamento dos dispositivos, mostrando o MAC *Adrress* de cada um também é importante. (nRF Connect for Desktop, 2024).

4. Desenvolvimento do Projeto

O desenvolvimento do projeto foi orientado pelo objetivo de reconhecer dispositivos *Bluetooth Low Energy* (BLE) em um raio de 10 metros utilizando a placa ESP32, além de estimar a distância desses dispositivos em relação à placa. Este capítulo detalha as etapas seguidas, as ferramentas utilizadas e as configurações implementadas durante o processo de desenvolvimento, assim como os aspectos técnicos específicos relacionados à comunicação e cálculo de distâncias.

4.1 Ambiente de Desenvolvimento

O projeto foi desenvolvido em um ambiente Linux. A escolha desse ambiente foi motivada por sua robustez e suporte eficiente às ferramentas necessárias. A linguagem de programação adotada foi C++, enquanto as operações de desenvolvimento e *upload* do código na placa ESP32 foram realizadas através da Arduino IDE, selecionada por sua interface amigável e ampla compatibilidade com a ESP32.

A instalação da Arduino IDE foi realizada através do site oficial, seguida pela configuração adicional para suportar a ESP32. Esse processo envolveu acessar o menu de preferências, adicionar o *link*[1] específico do gerenciador de placas da *Espressif* e, posteriormente, instalar o pacote ESP32 através do gerenciador de placas da Arduino IDE.

---

[1] Link para biblioteca do ESP32: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

4.2 Configuração do *Broker* Mosquitto

Para a comunicação via protocolo MQTT, foi utilizado o *Broker* Mosquitto. Este *software* foi instalado no Ubuntu utilizando o comando 'sudo dnf install mosquitto'. A configuração do Mosquitto envolveu a edição do arquivo de configuração 'mosquitto.conf' para permitir conexões anônimas e a utilização da porta 1883. Com isso, o sistema foi preparado para operar sem a necessidade de autenticação por usuário ou senha, facilitando a comunicação entre os dispositivos.

Comandos adicionais foram utilizados para gerenciar o serviço Mosquitto, como verificar a abertura da porta 1883, habilitar o serviço para iniciar automaticamente, e iniciar ou parar o serviço conforme necessário. Além disso, a verificação dos logs do Mosquitto foi feita para monitorar e solucionar possíveis problemas. A seguir, é apresentada uma tabela com comandos que podem ser úteis na utilização do *broker*.

Tabela 1. Comandos para utilização do Mosquitto.

Descrição do Comando	Comando
Habilitar o Mosquitto	`sudo systemctl enable mosquitto`
Iniciar o Mosquitto	`sudo systemctl start mosquitto`
Parar o Mosquitto	`sudo systemctl stop mosquitto`
Verificar se a porta 1883 está aberta	`sudo ss -tuln | grep 1883`
Verificar *logs* do Mosquitto	`sudo journalctl -u mosquitto`

Fonte: Autoras.

Dessa forma, a partir da configuração correta dos arquivos e do *broker*, conforme os comandos da Tabela 1, é possível continuar a conexão com outros *softwares* para um acesso mais dinâmico dos dados. A flexibilidade proporcionada pelo Mosquitto permite a integração com diversas plataformas e sistemas, ampliando as possibilidades de utilização e análise dos dados coletados pelos dispositivos BLE.

4.3 Comunicação e Protocolo

A ESP32 foi configurada para operar utilizando o protocolo IPv4, o qual foi atribuído de acordo com a rede *WiFi* à qual a placa está conectada. A comunicação entre a ESP32 e o *Broker* Mosquitto utiliza a porta 1883, o que é padrão para o protocolo MQTT. A camada de transporte utilizada foi o TCP (*Transmission Control Protocol*), escolhido devido à sua confiabilidade em garantir a entrega dos dados de forma ordenada e sem perdas, essencial para a correta transmissão das informações de detecção de dispositivos BLE.

4.3.1 Descobrindo o IP da rede

Para que a conexão seja estabelecida, é preciso descobrir o endereço IP da rede na qual está conectado. O IP (Internet Protocol) permitirá a sua identificação e comunicação com outros dispositivos.

Na figura 3, foi utilizado o comando `ip a` no terminal para listar os endereços IP configurados nas interfaces de rede do dispositivo. Este comando exibe informações detalhadas sobre cada interface de rede, como o MAC *Address* e o IPv4.

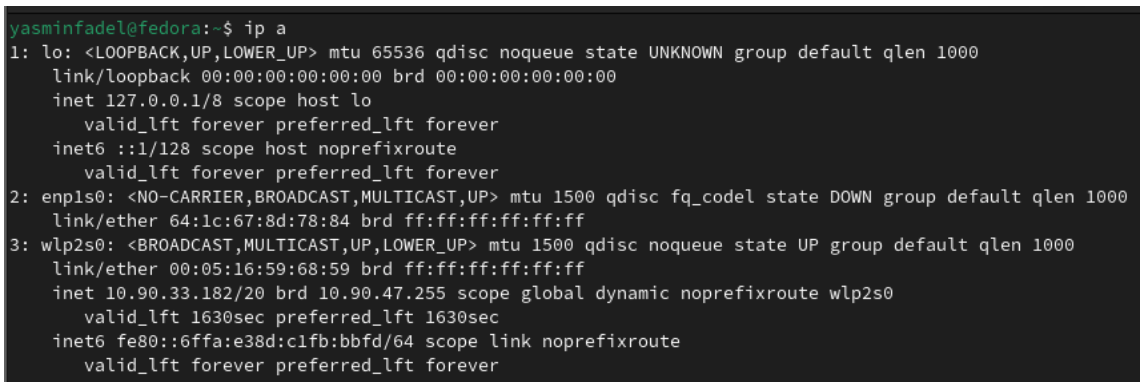

Figura 3. Comando para descobrir o IP no terminal. Fonte: Autoras.

A partir da Figura 3, verifica-se que a interface `wlp2s0` é a que está ativa e conectada à rede sem fio, possuindo o endereço IP `10.90.33.182`. Este endereço é utilizado para estabelecer conexões e comunicações com outros dispositivos na mesma rede local.

Dessa maneira, será possível configurar a conexão entre o *broker* Mosquitto, o MQTT *Explorer* e a Arduino IDE para verificar o fluxo de dados do projeto.

4.4 Utilização do MQTT *Explorer*

Para testar e monitorar as mensagens MQTT, foi empregado o MQTT *Explorer*. Esta ferramenta cliente foi baixada do site oficial e configurada para se conectar ao *broker* Mosquitto. A configuração envolveu a inserção do nome, *host*, protocolo e porta, com a conexão estabelecida sem a necessidade de autenticação devido à configuração do Mosquitto para permitir conexões anônimas. A partir disso, é possível inscrever-se no tópico relevante para monitorar os dados enviados pela ESP32.

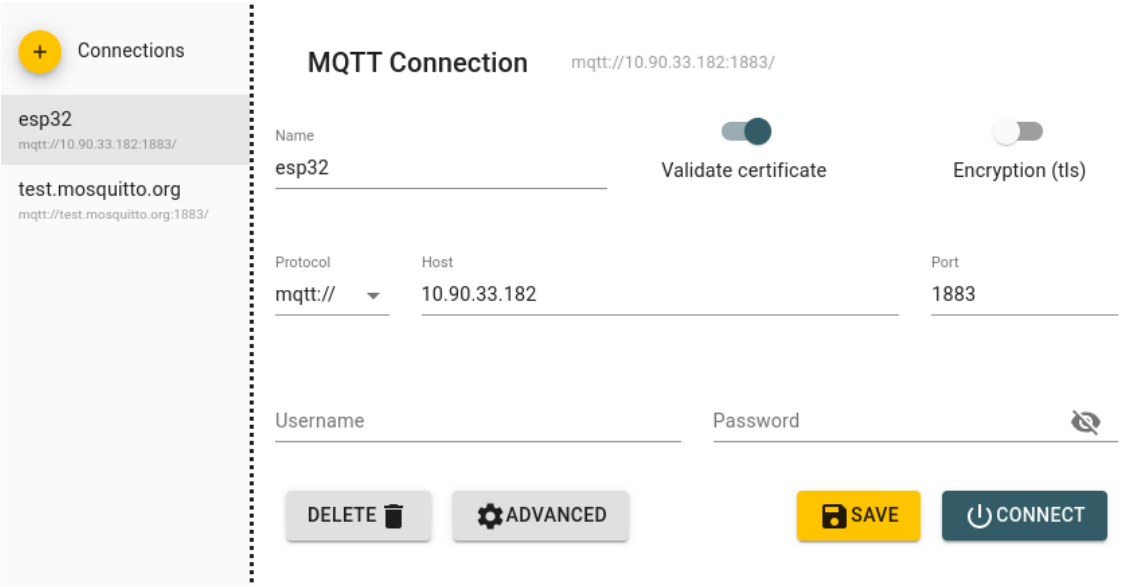

Figura 4. Interface de conexão do MQTT *Explorer*. Fonte: Autoras.

A Figura 4 apresenta a interface gráfica do MQTT *Explorer*, que permite conectar um dispositivo a um servidor MQTT. A interface se divide em duas áreas principais: à esquerda, uma lista de conexões MQTT existentes; à direita, uma caixa de diálogo para configurar uma nova conexão. A caixa de diálogo possui campos para informar o nome da conexão, o protocolo MQTT (MQTT ou MQTTS), o endereço do *host*, a porta do servidor, o nome de usuário, a senha, e um botão para validar o certificado do servidor. Na lista de conexões à esquerda, estão listadas duas conexões com seus respectivos nomes e endereços. O usuário pode editar ou remover essas conexões através de botões específicos. Ao clicar em "*CONNECT*", o usuário poderá conectar o dispositivo ao servidor MQTT.

A partir da conexão estabelecida com o *broker* Mosquitto, será possível visualizar os dados de forma mais intuitiva e eficiente, permitindo ao usuário monitorar e controlar os dispositivos conectados à rede. Além disso, a interface do MQTT *Explorer* fornece recursos avançados para gerenciar as mensagens publicadas e assinadas, permitindo ao usuário filtrar, ordenar e visualizar os dados de acordo com suas necessidades específicas.

4.5 Cálculo de Distância

A distância entre o ESP32 e os dispositivos BLE é calculada utilizando uma fórmula específica baseada na intensidade do sinal recebido (RSSI). A seguir a fórmula aplicada:

$$d = 10^{\frac{(rssi_{ref} - RSSI)}{10 * N}}$$, onde:

- d **(Distância):** representa a distância estimada entre o ESP32 e o dispositivo BLE. É o valor que se deseja calcular.

- rssi_ref **(RSSI de Referência):** este é o valor de referência da intensidade do sinal recebido a uma distância conhecida (1 metro) do transmissor BLE. Serve como um ponto de partida para a comparação com o RSSI atual.
- RSSI **(Intensidade do Sinal Recebido):** é a medida da potência do sinal recebido pelo ESP32 do dispositivo BLE. Esse valor é suavizado para reduzir as flutuações e interferências momentâneas.
- N **(Fator de Propagação):** este é o fator que leva em consideração as condições do ambiente, como obstáculos e interferências, que podem afetar a propagação do sinal. Ele ajusta a fórmula para diferentes ambientes, sendo crucial para uma estimativa precisa da distância.

Essa fórmula, derivada do artigo de referência de Venkatesh (2021), é utilizada para estimar a distância com base na diferença entre a intensidade do sinal de referência (rssi_ref) e a intensidade do sinal (RSSI). Esses valores são ajustados por um fator de propagação N, que leva em consideração as condições do ambiente, como obstáculos e interferências, que podem afetar a propagação do sinal. O RSSI varia inversamente com a distância, ou seja, quanto mais fraco o sinal, maior a distância entre o ESP32 e o dispositivo BLE.

Ao aplicar essa fórmula, é possível converter as leituras de RSSI em uma estimativa de distância, permitindo uma medição razoavelmente precisa. O uso dessa abordagem facilita a obtenção de dados confiáveis para aplicações que dependem de monitoramento de proximidade ou localização, utilizando os dados de RSSI coletados pelo ESP32.

5. Resultados

A estratégia de desenvolvimento foi dividida em três etapas principais: configuração do *hardware*, desenvolvimento da aplicação no ESP32 e desenvolvimento da aplicação MQTT. Esta abordagem modular garantiu que cada componente fosse implementado e testado antes de avançar para a próxima fase.

A Figura 3 a seguir, descreve o fluxo de comunicação que começa com a geração de dados por dispositivos BLE, seguida pela transmissão via RF, processamento no ESP32, encaminhamento pelo roteador *gateway*, gerenciamento pelo *broker* MQTT, e finalização com a visualização dos dados no MQTT *Explorer*. Este processo demonstra como as diversas tecnologias foram integradas para a comunicação e monitoramento em tempo real dos dados.

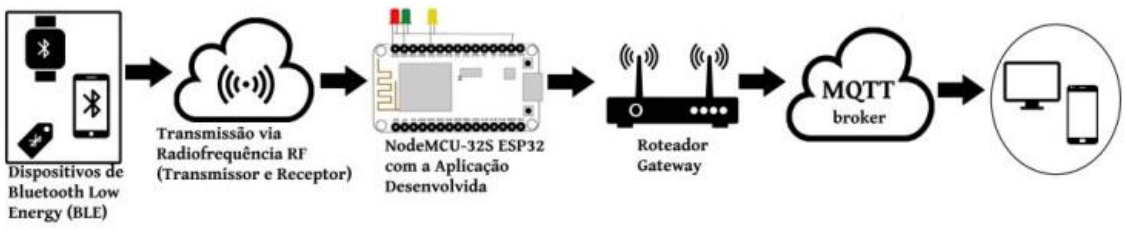

Figura 5. Fluxo de Comunicação. Fonte (Lira et. al, 2019).

Durante a configuração do *hardware*, foi focado em assegurar que o ESP32 e o módulo *Bluetooth* estivessem funcionando corretamente. Isso envolveu testes básicos de comunicação e escaneamento BLE. Após validar o *hardware*, foi desenvolvida a aplicação no ESP32 para escanear dispositivos BLE e calcular suas distâncias estimadas. Foram realizados testes iterativos para garantir a precisão das estimativas e ajustado o valor de RSSI conforme necessário para definir um raio de alcance adequado.

Na etapa final, foi integrada a funcionalidade MQTT, permitindo que o ESP32 enviasse dados sobre os dispositivos BLE detectados para um *broker* MQTT. Foi implementada a lógica de reconexão para assegurar a eficácia da comunicação.

As funcionalidades desenvolvidas incluem a detecção de dispositivos BLE, o cálculo de distância estimada com base no RSSI e comunicação MQTT para transmissão dos dados. As informações recebidas via MQTT são exibidas na interface do MQTT *Explorer*.

Este trabalho demonstra a viabilidade de utilizar o ESP32 para criar um dispositivo eficaz de detecção de BLE, destacando a integração bem-sucedida de *hardware* e *software* para alcançar os objetivos propostos. Abaixo, segue um exemplo de como o resultado da detecção dos dispositivos é mostrada no MQTT *Explorer*.

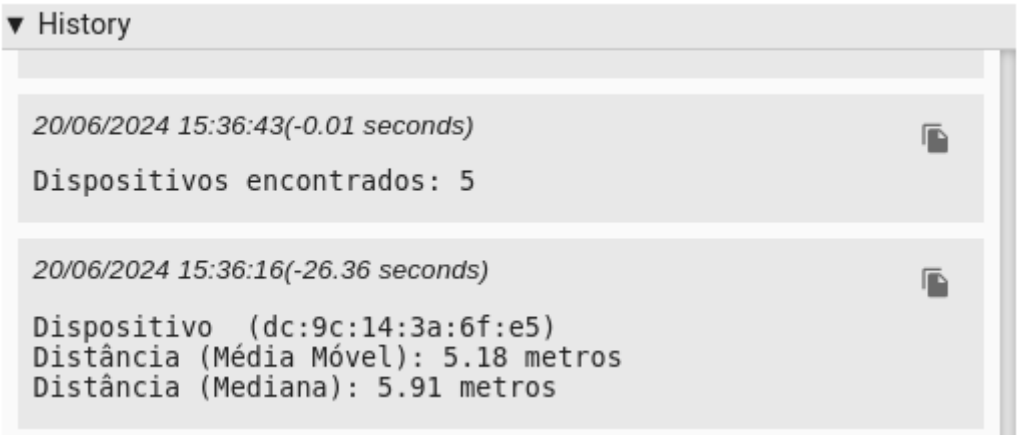

Figura 6. Interface do MQTT Explorer com resultado das distâncias. Fonte: Autoras.

A Figura 6 mostra o histórico de eventos das detecções de dispositivos *Bluetooth Low Energy* (BLE) em um tópico do MQTT *Explorer*. Neste evento, foram detectados 5 dispositivos BLE, seguidos do MAC *Adrress* do dispositivo detectado, sua distância de acordo com a média móvel e com a mediana. A exibição de diferentes tipos de distâncias, como média móvel e mediana, oferece uma visão mais completa da localização do dispositivo BLE, levando em consideração a flutuação da qualidade do sinal.

6. Considerações Finais e Trabalhos Futuros

Esta aplicação de detecção pode ser utilizada futuramente para aplicações mais sofisticadas em áreas como automação, segurança, entre outras. Um dos exemplos de futuras melhorias é a integração deste sistema de detecção com redes IoT para monitoramento de ativos em tempo real. Com essa melhoria, seria possível rastrear e gerenciar a localização de equipamentos e ferramentas em grandes instalações industriais

ou hospitais, garantindo que recursos críticos estejam sempre disponíveis quando necessários e reduzindo significativamente o tempo perdido na busca por itens deslocados. Essa funcionalidade poderia ser ampliada para incluir alertas automáticos quando os objetos saem de zonas pré-determinadas, aumentando ainda mais a eficiência operacional e a segurança.

7. Referências

Alves et al. (s.d). Detecção de Proximidade de Dispositivos WiFi com ESP8266 utilizando Intensidade de Sinal Recebido. Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE). Acesso em: 17 jun. de 2024.

Amazon. (2024). O que é MQTT? Disponível em: https://aws.amazon.com/pt/what-is/mqtt/. Acesso em: 18 jun. 2024.

ArduinoBLE - Arduino Reference. Disponível em: https://www.arduino.cc/reference/en/libraries/arduinoble/. Acesso em: 13 jun. 2024.

Badihi et. al. (2019). Intelligent Construction Site: On Low Cost Automated Indoor Localization Using Bluetooth Low Energy Beacons. IEE Conference. Acesso em: 17 jun. de 2024.

Blog da Curto, B. (2018). Conhecendo o esp32. Disponível em: https://www.curtocircuito.com.br/blog/conhecendo-esp32/. Acesso em: 13 jun. 2024.

Boulic, R. and Renault, O. (1991). “3D Hierarchies for Animation”, In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England. Acesso em: 02 jun. 2024.

Conceição, Wellington; Costa, Romualdo. (2019). Análise do Protocolo MQTT para Comunicação IoT através de um Cenário de Comunicação. UniAcademia. Disponível em: https://seer.uniacademia.edu.br/index.php/cesi/article/viewFile/1688/1231. Acesso em: 18 jun. 2024.

ESP32 Arduino: Obtendo o IP do cliente remoto. [S. l.], 2024. Disponível em: https://suadica.com/dica.php?d=391&t=esp32-arduino-obtendo-o-ip-do-cliente-remoto. Acesso em: 20 jun. 2024.

ESP32 WIFI: COMUNICAÇÃO COM A INTERNET. [S. l.], 20 jun. 2024. Disponível em: https://www.usinainfo.com.br/blog/esp32-wifi-comunicacao-com-a-internet/. Acesso em: 20 jun. 2024.

ESP32 WiFi e Bluetooth de Diversos Modelos. (s.d.). UsinaInfo. Disponível em: https://www.usinainfo.com.br/esp32-611#:~:text=As%20especifica%C3%A7%C3%B5es%20do%20ESP32%20s%C3%A3o. Acesso em: 18 jun. 2024.

K. Mekki, E. Bajic and F. Meyer. (2019). "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, pp. 787-792, doi: 10.1109/WF-IoT.2019.8767287. Acesso em: 11 jun. 2024.

Lira et al. (s.d). Localizador de objetos em curtas distâncias baseado em Bluetooth BLE com monitoramento IoT via MQTT. Instituto Federal de Educação Ciência e Tecnologia do Ceará (IFCE). Acesso em: 17 jun. de 2024.

Mosquitto - Entenda as suas funcionalidades e vantagens. (2018). Disponível em: https://engprocess.com.br/mosquitto/. Acesso em: 18 jun. 2024.

nRF Connect for Desktop. Disponível em: https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-Desktop. Acesso em: 13 jun. 2024.

OASIS – OASIS Standard. (2014). MQTT Version 3.1.1. Acesso em: 18 jun. 2024.

Riesebos et. al. (2022). Smartphone-Based Real-Time Indoor Positioning Using BLE Beacons. IEE. Acesso em: 17 jun. de 2024.

Singh, M., Rajan, M., Shivraj, V., and Balamuralidhar, P. (2015). Secure MQTT for Internet of Things (IoT). In 2015 Fifth International Conference on Communication Systems and Network Technologies, pages 746–751. IEEE. Disponível em: https://ieeexplore.ieee.org/abstract/document/7280018/. Acesso em: 13 jun. 2024.

Souza et al. (2020). Uma avaliação do Broker Mosquitto no contexto de Internet das Coisas. UTFPR. Disponível em: https://eventos.utfpr.edu.br/sicite/sicite2020/paper/download/6659/2548. Acesso em: 18 jun. 2024.

Subhan et al. (2019). Experimental analysis of received signals strength in Bluetooth Low Energy (BLE) and its effect on distance and position estimation. IEE. Acesso em: 17 jun. de 2024.

Venkatesh et. al. (s.d.) Indoor Localization in BLE using Mean and Median Filtered RSSI Values. IEE. ICOEI. Acesso em: 17 jun. 2024.

What is Mosquitto MQTT? Disponível em: https://www.eginnovations.com/documentation/Mosquitto-MQTT/What-is-Mosquitto-MQTT.htm. Acesso em: 18 jun. 2024.