UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

LUIZ FELIPE RODRIGUES GONÇALVES

ESTUDO E AVALIAÇÃO DO BLUETOOTH LOW ENERGY E SCADABR EM AMBIENTE INDOOR E INDUSTRIAL

JOÃO PESSOA - PB
Dezembro de 2021

LUIZ FELIPE RODRIGUES GONÇALVES

ESTUDO E AVALIAÇÃO DO BLUETOOTH LOW ENERGY E SCADABR EM AMBIENTE INDOOR E INDUSTRIAL

Trabalho de Conclusão de Curso submetido à Universidade Federal da Paraíba, como requisito necessário para obtenção do grau de Bacharel em Engenharia Elétrica

Orientador: Prof. Dr. Juan Moises Mauricio Villanueva

João Pessoa - PB

Dezembro de 2021

Catalogação na publicação
Seção de Catalogação e Classificação

G635e Goncalves, Luiz Felipe Rodrigues.
Estudo e avaliação do Bluetooth Low Energy e ScadaBR em
ambiente indoor e industrial / Luiz Felipe Rodrigues
Goncalves. - João Pessoa, 2021.
61 f. : il.

Orientação: Juan Moises Mauricio Villanueva.
TCC (Graduação) - UFPB/CEAR.

1. Sistema de Posicionamento. 2. Bluetooth Low Energy.
3. ScadaBR. I. Villanueva, Juan Moises Mauricio. II.
Título.

UFPB/CT/BSCT CDU 621.3(043.2)

Elaborado por ONEIDA DIAS DE PONTES - CRB-CRB-15 - 198

LUIZ FELIPE RODRIGUES GONÇALVES

ESTUDO E AVALIAÇÃO DO BLUETOOTH LOW ENERGY E SCADABR EM AMBIENTE INDOOR E INDUSTRIAL

Trabalho de Conclusão de Curso submetido à Coordenação de Engenharia Elétrica do Centro de Energias Alternativas e Renováveis da Universidade Federal da Paraíba como parte dos requisitos necessários para a obtenção do título de Engenheiro Eletricista.

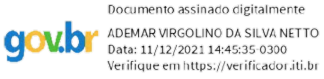
Data de Aprovação: 09 / 12 / 2021

Nota: 9,0

Aprovado por:

Prof. Dr. Juan Moises Mauricio Villanueva (Orientador)
Universidade Federal da Paraíba

Prof. Dr. Waslon Terllizzie Araújo Lopes (Avaliador)
Universidade Federal da Paraíba



Prof. Dr. Ademar Virgolino da Silva Netto (Avaliador)
Universidade Federal da Paraíba

João Pessoa - PB
DEZEMBRO/2021

AGRADECIMENTOS

A Deus, por ter me criado e conservado apesar de minha ingratidão e por ter sido a força que me sustentou, na eucaristia, para continuar seguindo em frente.

A Virgem Maria Santíssima, a quem dedico e entrego este trabalho, por ser dona e mestra desse miserável e pecador escravo.

A São José, meu pai solícito e fiel intercessor, por ter me auxiliado imensamente ao longo de toda minha vida.

A minha mãe Maria da Graça por se sacrificar diariamente para me dar uma boa vida e lembrar-me sempre que eu devia me dedicar aos estudos.

Ao meu pai Ricardo (in memorian), por todos os ensinamentos e por ter ajudado na formação do meu caráter.

Ao professor Juan Villanueva, pela orientação e paciência ao longo deste trabalho e por ter me dado novo ânimo para continuar no curso.

Aos demais professores que contribuíram na minha formação como engenheiro e profissional. Em especial à professora Camila Gehrke pelos ensinamentos no Fablab.

Ao meu amigo Thommas Flores, por todo auxílio no desenvolvimento deste trabalho e a Polyana Collaço pelos longos anos de amizade e paciência.

Aos amigos que fiz no decorrer do curso, espero encontra-los no futuro para partilharmos nossas experiências.

A todos os familiares e amigos que colaboraram indiretamente para a realização deste trabalho.

RESUMO

Sistemas de posicionamento tem sido amplamente investigados quanto da qualidade de sua implementação em ambientes indoor residenciais e industriais. A distância de um objeto pode ser medida utilizando o indicador de intensidade de sinal recebido (RSSI) da tecnologia *Bluetooth Low Energy*(BLE), amplamente utilizada em aplicações que envolvem cálculo de distâncias de até 10 metros. Este trabalho propõe um estudo acerca da eficiência do *Bluetooth Low Energy* em aplicações de ambiente fechado controlado e em um laboratório com diversas máquinas e possível interferência eletromagnética usando localizadores. Para realizar a aquisição dos valores da potência e analisar os dados foi utilizado a placa de desenvolvimento ESP32, traçando as curvas de RSSI *versus* distância e plotando as curvas características com diferentes parâmetros. Foi utilizado também o software supervisório ScadaBR para monitorar os dados vindos da placa através do protocolo Modbus TCP/IP. Por fim, foi feita uma análise comparativa do comportamento de dois localizadores quanto da interferência causada por um conjunto motobomba de uma rede de abastecimento de água do Laboratório de Eficiência Energética e Hidráulica em Saneamento (LENHS). Os resultados obtidos demonstraram um erro entre 50 centímetros e 1 metro na medição da distância, além de uma variação -6 dBm no valor médio do RSSI.

Palavras-chave: Sistema de Posicionamento; *Bluetooth Low Energy*; ScadaBR.

ABSTRACT

Positioning systems are widely investigated due to its quality of implementation inside residential and industrial areas. The distance from an object can be measured by a Received Signal Strength Indicator (RSSI) of the Bluetooth Low Energy (BLE) technology, extensively applied in calculation of distances up to 10m. This project aims to study the efficiency of BLE in closed controlled environment applications and in a lab with multiple machines with the possibility of electromagnetic interference using locators. The development board ESP32 was used in order to acquire the power values and analyze the data, tracing the RSSI versus distance curves and plotting the characteristic curve with different parameters. The ScadaBR supervisor software was also used to monitor the coming data from the board through the Modbus TCP/IP protocol. Finally, a comparative analysis of the behavior between two locators and the interference caused by a motor pump set in a water supply network of the Laboratory of Energy Efficiency and Hydraulics in Sanitation (LNHS) was conducted. The results acquired showed a measurement error between 50cm and a meter, in addition -6dBm variation in the average RSSI value.

Keywords: Positioning System; Bluetooth Low Energy; ScadaBR.

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de ambiente com pontos de localização 14
Figura 2 – Trilateração 15
Figura 3 – Modelo OSI 16
Figura 4 – Comparação entre modelo OSI e TCP/IP 17
Figura 5 – Estrutura do Modbus TCP/IP 17
Figura 6 – Divisão do Espectro de Frequência 20
Figura 7 – Exemplo de decaimento logarítmico de RSSI 22
Figura 8 – Localizador *Bluetooth* LE 25
Figura 9 – Placa de desenvolvimento ESP32 26
Figura 10 – Configuração do DataSource 28
Figura 11 – Criação da DataPoint de monitoramento de um Localizador 29
Figura 12 – Estrutura de envio 29
Figura 13 – Endereço IP criado pela ESP 32
Figura 14 – Modelo do experimento 1 34
Figura 15 – Curvas obtidas da tabela 7 35
Figura 16 – Curva Fit polinomial de 6ª ordem 35
Figura 17 – Curva Fit Polinomial de 5ª ordem 36
Figura 18 – Curva Fit polinomial de 4ª ordem 37
Figura 19 – Curva Fit polinomial de 3ª ordem 37
Figura 20 – Curva Exponencial Estimada em Ambiente Controlado 39
Figura 21 – Comparativo entre curvas medidas e teórica estimada 40
Figura 22 – Modelo do experimento 2 41
Figura 23 – Ambiente dos experimentos no LENHS 42
Figura 24 – Dados do RSSI monitorados no ScadaBR 42
Figura 25 – Modelo do terceiro experimento 44
Figura 26 – Montagem utilizada para o terceiro experimento 45
Figura 27 – Monitoramento dos localizadores no ScadaBR 45
Figura 28 – Configuração do Arduino para Pleno Funcionamento do Código 52

LISTA DE CÓDIGOS

Código 1 Configuração do Scan BLE 30
Código 2 Obtenção do RSSI 31
Código 3 Acesso a rede Wi-Fi local 31
Código 4 Definição do Cabeçalho do Modbus 32
Código 5 Função de Leitura dos Registradores 32

LISTA DE TABELAS

Tabela 1 – MBAP do protocolo Modbus TCP/IP 18
Tabela 2 – Comparação entre *Bluetooth* padrão, BLE e *Wi-Fi* 19
Tabela 3 – Valor de β para diferentes ambientes 21
Tabela 4 – Especificações do localizador 24
Tabela 5 – Diferença entre RSSI com diferente posição da Antena 25
Tabela 6 – Especificações da ESP32 26
Tabela 7 – RSSI medido em ambiente controlado 36
Tabela 8 – Erro Médio e Desvio Padrão para diferentes valores de β 39
Tabela 9 – Erro e Desvio Padrão para melhor aproximação teórica 39
Tabela 10 – RSSI medido no LENHS 43
Tabela 11 – Especificações do motor 44
Tabela 12 – Valor Médio e desvio padrão encontrados 46

LISTA DE ABREVIATURAS E SIGLAS

AFH	Salto em Frequência Adaptativo
BLE	Bluetooth Low Energy
CLP	Controlador Lógico Programável
GFSK	Modulação por Chaveamento de Frequência
IDE	Ambiente de Desenvolvimento Integrado
IIoT	Internet das Coisas Industrial
IP	Protocolo de Internet
IPS	Sistema de Posicionamento Indoor
IoT	Internet das Coisas
LENHS	Laboratório de Eficiência Energética e Hidráulica em Saneamento
MBAP	Modbus Application Protocol
OSI	Sistema Aberto de Interconexão
RMSE	Raiz do Erro Médio Quadrático
RSSF	Redes de Sensores Sem Fio
RSSI	Indicador de Intensidade de Sinal Recebido
SCADA	Controle Supervisório e Aquisição de Dados
TCP	Protocolo de Transmissão de Controle
Wi-Fi	Wireless Fidelity

SUMÁRIO

1 INTRODUÇÃO 11
1.1 TRABALHOS RELACIONADOS **12**
1.2 OBJETIVOS **12**
1.3 ORGANIZAÇÃO DO DOCUMENTO **13**

2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 SISTEMAS DE LOCALIZAÇÃO INDOOR **14**
2.2 PROTOCOLOS INDUSTRIAIS **15**
2.2.1 Protocolo Modbus TCP/IP 16
2.3 BLUETOOTH LOW ENERGY (RSSI) **18**
2.3.1 Potência RSSI 20
2.4 SOFTWARE SUPERVISÓRIO **22**

3 MATERIAIS E MÉTODOS 24
3.1 LOCALIZADOR BLE **24**
3.2 MÓDULO ESP32 . . . . . **24**
3.3 SUPERVISÓRIO SCADA-BR **27**
3.3.1 Data Source 27
3.3.2 Data Point 28
3.4 ALGORITMO DE COMUNICAÇÃO **29**
3.4.1 Comunicação bluetooth 30
3.4.2 Comunicação Modbus TCP/IP via WI-FI 31

4 RESULTADOS 34
4.1 EXPERIMENTO 1 . . . . . **34**
4.2 EXPERIMENTO 2 . . . . . **40**
4.3 EXPERIMENTO 3 . . . . . **43**

5 CONCLUSÃO 47

REFERÊNCIAS 48

APÊNDICE A – CODIGO IMPLEMENTADO EM ARDUINO PARA COMUNICACAO BLE E WI-FI **52**

1 INTRODUÇÃO

Com a chegada da quarta revolução industrial novos mecanismos de integração de processos foram desenvolvidos para abrigar as novas tecnologias como as redes sem fio (*wireless*) e a interconexão de máquinas.

Na era da Internet das Coisas (IoT) realizar a comunicação entre diferentes dispositivos, monitorá-los e guardar grandes volumes de dados tem sido um desafio de toda a engenharia. Nesse sentido, cada vez mais se tem pensado em soluções que ofereçam um maior grau de eficiência e baixo custo de implementação.

Uma dessas soluções busca realizar a aquisição de dados de posicionamento em diferentes ambientes a partir de redes de sensores sem fio (RSSF). Tanto para localização *indoor* como para industrial existe a necessidade de se ter uma elevada precisão na posição medida, na ordem de centímetros, e menor taxa de variação de sinal.

Segundo Naik et al. (2019) os métodos ou tecnologias usadas no posicionamento *indoor* têm muitas desvantagens, como baixa precisão, alta complexidade, falta de confiabilidade e alto custo de hardware. Além disso, Mekki, Bajic e Meyer (2019) discute que não há uma única tecnologia ou combinação que seja capaz de recriar em ambientes *indoor* a experiência de um sistema de localização global (GPS) para ambientes outdoor. Como alternativa de baixo custo e maior eficiência energética há o *Bluetooth Low Energy* que vêm sendo empregado em larga escala em aplicações de IoT e localização.

O BLE utiliza principalmente o indicador de intensidade de sinal recebido (em inglês *Received Signal Strenght Indicator*) de diversos localizadores para calcular a distância real do objeto desejado, sendo utilizados diferentes métodos de implementação. (MEKKI; BAJIC; MEYER, 2019)

Um outro problema para sistemas de localização é a forma como os dados são monitorados, guardados e, posteriormente, analisados. Uma vez que a posição de uma pessoa possa ser verificada continuamente um grande montante de valores de posição são gerados e não é possível verificar a relação entre eles de forma prática sem o uso de gráficos e tabelas.

Nessa perspectiva surge o ScadaBR, um software supervisório livre e de código-fonte aberto que permite realizar a aquisição de dados em tempo real e exibi-los em representações gráficas para melhor tomada de decisão. O ScadaBR permite a comunicação com diversos protocolos, dentre eles o Modbus TCP/IP que será utilizado em comunhão com o módulo de desenvolvimento ESP32 para registrar os valores de RSSI de diferentes localizadores.

Neste trabalho será realizado o estudo do *Bluetooth* LE para aplicações de sistemas de localização *indoor* e industrial com monitoramento em tempo real através do software

supervisório ScadaBR.

1.1 TRABALHOS RELACIONADOS

Após uma revisão sistemática da literatura nos últimos anos, foram encontrados diversos trabalhos que influenciaram a pesquisa em questão e que, portanto, contribuíram para a análise dos resultados obtidos, como caracterização das curvas e metodologias aplicáveis para redução de ruído e erros na medição .

D'Aloia et al. (2020) desenvolveu um sistema de posicionamento *indoor* (SPI) através do método de *fingerprinting* utilizando diversos módulos ESP32 e BLE. No trabalho são comparadas duas técnicas de aprendizado de máquina (*machine learning*) para melhor predição da posição real.

Em Mekki, Bajic e Meyer (2019) a metodologia de trilateração é utilizada em conjunto com BLE e protocolo MQTT. Os resultados dos experimentos realizados em ambientes internos chegaram a uma precisão de 1,5 metros. O trabalho também traz um comparativo de valores de coeficientes de decaimento de sinal de diferentes ambientes como em espaço aberto, *indoor* e industrial.

Li et al. (2018) utiliza uma série de condicionamentos matemáticos e computacionais, como o filtro de Kalman, para tratar os dados de RSSI. Também é exposto no trabalho a alta taxa de variação da potência do sinal, chegando a um elevado valor de erro.

Em Kaczmarek, Ruminski e Bujnowski (2016) foi desenvolvido um sistema de localização *indoor* por triangulação. Também é discutido sobre a interferência entre os sinais Wi-Fi e BLE visto que usam a mesma frequência de operação de 2.4 GHz.

Um modelo de decaimento de RSSI *versus* distância é mostrado por Röbesaat et al. (2017). Dois filtros são utilizados para suavizar as curvas. Também é feita a detecção do caminho percorrido por um objeto através de um sistema de trilateração.

Mouhammad et al. (2019) desenvolve um SPI com a tecnologia BLE e diferentes filtros em conjuntos, bem como um sistema de pesos para técnicas de trilateração. Seus resultados chegaram a um erro médio de 50 centímetros em 95% do tempo.

Em Chou et al. (2018) é realizada a implementação do sistema de localização em uma fábrica de lentes ópticas. Os testes realizados em ambiente aberto e industrial mostraram que o elevado número de obstruções e máquinas podem levar a perda de sinal, bem como a confiabilidade dos dados em decorrência de reflexão do sinal.

1.2 OBJETIVOS

O objetivo deste trabalho é realizar a análise do *Bluetooth Low Energy* em ambientes abertos e fechados, com características industrial, isto é, na presença de equipamentos possam interferir no sinal de comunicação

Os objetivos específicos são:

- Avaliar o esquema de comunicação entre o sistema de localizador, microcontrolador e supervisório, utilizando o protocolo MODBUS TCP/IP;
- Calibrar o localizador através da análise do decaimento do RSSI em função da distância em ambiente *indoor* e industrial com interferência eletromagnética proveniente de máquinas;
- Avaliar o impacto da interferência eletromagnética no sinal transmitido pelo localizador.

1.3 ORGANIZAÇÃO DO DOCUMENTO

Este trabalho é composto por cinco capítulos. No primeiro capítulo é apresentada a revisão bibliográfica de trabalhos similares, os objetivos gerais e específicos.

O segundo capítulo apresenta a fundamentação teórica dos principais temas abordados no trabalho como sistemas de localização *indoor*, *Bluetooth low energy*, potência RSSI, protocolo Modbus TCP/IP e softwares supervisórios.

O terceiro capítulo apresenta os materiais e a metodologia empregada na realização dos experimentos.

O quarto capítulo traz a análise dos resultados dos experimentos, através da caracterização das curvas de distância em função da potência e o monitoramento.

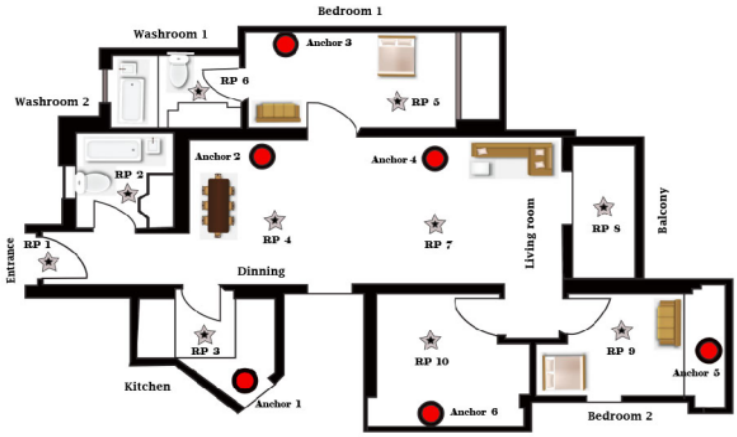
O quinto capítulo, por fim, apresenta a conclusão acerca dos resultados do capítulo quarto, comparando os valores obtidos com os apresentados na literatura, limitações de implementação e trabalhos futuros que podem ser realizados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SISTEMAS DE LOCALIZAÇÃO INDOOR

Sistemas de localização e rastreamento são uma peça fundamental no cotidiano das grandes cidades. Devido ao crescente uso de aplicativos mobile destinados a locomoção a necessidade de saber onde pessoas e objetos estão se torna imprescindível. Muitas tecnologias foram avaliadas para este propósito como *ZigBee*, *Wi-Fi* e *Bluetooth*. (KACZMAREK; RUMINSKI; BUJNOWSKI, 2016) O número de localizadores utilizados também influencia em quão preciso o resultado será. Na Figura 1 é mostrado um exemplo de ambiente em que foram posicionados dispositivos para mapear o RSSI.

Figura 1 – Exemplo de ambiente com pontos de localização



Fonte: Qureshi, Umair e Hancke (2019)

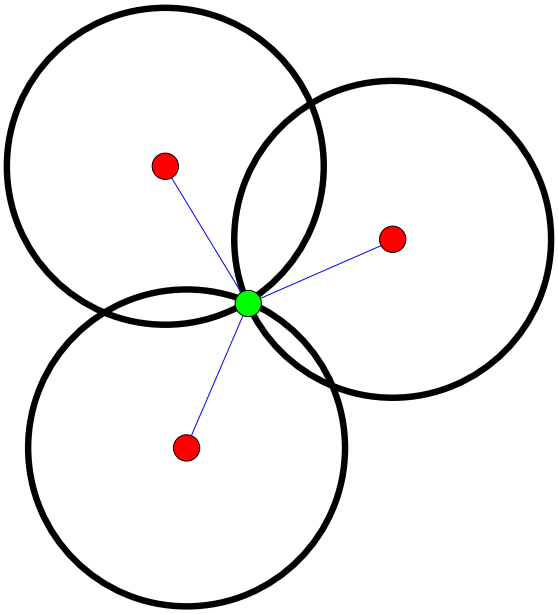
Também tem se demonstrado, como em Qureshi, Umair e Hancke (2019) , que localização *indoor* é especialmente útil no cuidado de idosos e outras aplicações na área da saúde (*Health Care*).

Além disso, a guarda dos dados de localização podem ser utilizados para criar algoritmos de inteligência artificial a fim de traçar perfis de movimentação, por exemplo, de clientes em um shopping, sendo capaz de informar qual o melhor caminho para a compra dos produtos.

Quando se trata de localização através de BLE, porém, não se chega a uma precisão alta devido ao valor RSSI variar constantemente. (SALIMIBENI et al., 2021). Segundo D'Aloia et al. (2020), a presença de obstáculos como paredes produz reflexões nas ondas eletromagnéticas que podem prejudicar a acurácia da medição. Por essa razão é comum utilizar métodos como cálculo da média ou filtro de Kalman para se chegar a um melhor resultado. (PATERNA et al., 2017)

Duas técnicas comumente utilizadas são a triangulação e a trilateração. A primeira parte de valores conhecidos de ângulos e é empregado, por exemplo, em sistemas de GPS, já a segunda parte de distâncias conhecidas, como pode ser visto na Figura 2.

Figura 2 – Trilateração



Fonte: Autor

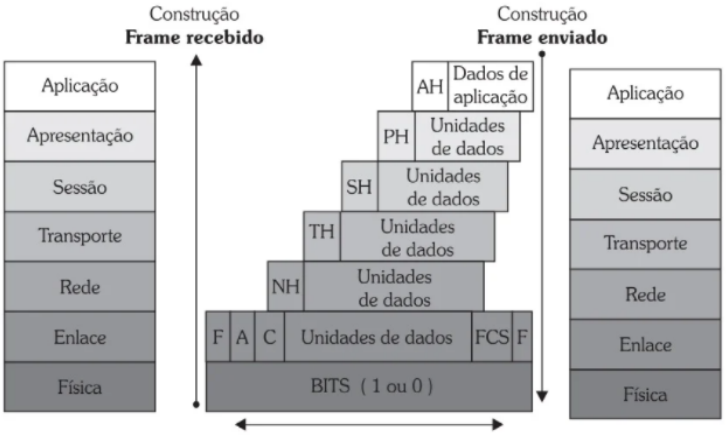
2.2 PROTOCOLOS INDUSTRIAIS

Para realizar corretamente todo o processo industrial é necessário que os componentes como computadores, computadores lógico programáveis (CLP), esteiras, braços robóticos, *etc.* sejam capazes de trocar dados entre si. Para suprir essa necessidade foram desenvolvidos os protocolos industriais. Um protocolo pode ser definido como um conjunto de regras e procedimentos para comunicação entre dispositivos. (MORAES, 2010)

Um protocolo industrial atua na interoperabilidade entre os diferentes sensores e atuadores com controladores lógicos e computadores. Dentre os principais protocolos industriais pode-se citar o Profibus, AS-I e Modbus.

No início da década de 80 houve um aumento no número padrões e protocolos, onde cada grande empresa tinha o seu próprio. Em razão disso foi criado o modelo OSI (*Open System Interconection*) para padronizar todo o sistema de redes e tornar possível a integração entre diferentes sistemas. É apresentado na Figura 3 a hierarquia do modelo, onde cada camada oferece e faz uso de serviços das camadas superior e inferior, respectivamente. A transmissão é feita a partir do encapsulamento dos dados da camada imediatamente superior com adição das informações da sua própria. No recebimento ocorre o processo inverso. (LUGLI; SANTOS, 2009)

Figura 3 – Modelo OSI



Fonte: Moraes (2010)

2.2.1 Protocolo Modbus TCP/IP

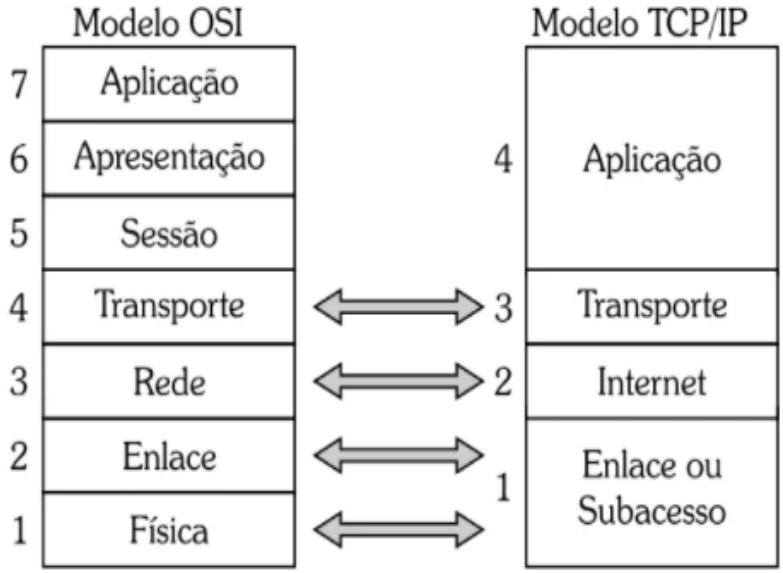
O protocolo Modbus é um protocolo industrial de comunicação desenvolvido no ano de 1979 pela Modicon. Faz parte da camada de aplicação para comunicação entre dispositivos dentro do processo de automação. A ligação estabelecida entre os componentes é do tipo mestre e escravo, no qual o mestre ordena uma função ao escravo como ler ou escrever um valor em um registrador.

O protocolo de controle de transmissão (TCP) destina-se a ser usado entre computadores em rede como um suporte contra erros. (POSTEL et al., 1981). Ele permite comunicação *Half* e *Full Duplex*, dependendo do processo solicitado. IP é o protocolo para forma de endereçamento. A junção dos dois protocolos forma o TCP/IP que designa uma arquitetura própria. Diferente do modelo OSI a estrutura é formada por apenas 4 camadas, como mostra a Figura 4.

No modelo TCP/IP a camada de aplicação realiza as funções das camadas de sessão, apresentação e aplicação do modelo OSI. Na camada de aplicação é realizada a interface entre o protocolo de comunicação e o aplicativo utilizado na rede. Na camada de apresentação é feita a conversão de formatos para que possam ser entendidos pelos subsistemas da rede e na camada de sessão é estabelecida a sincronia de comunicação, evitando que dados sejam perdidos por falhas de conexão. A camada de transporte realiza a troca de pacotes entre os sistemas que estão se comunicando, garantindo confiabilidade à transmissão dos dados. A camada de internet é responsável pelo roteamento. A camada de enlace faz a montagem dos *frames*. A camada física indica qual o padrão usado para troca de dados (Ethernet, RS232, *Bluetooth*).

Quando as informações do Modbus são enviadas usando esses protocolos, os dados são passados para o TCP, onde as informações adicionais são anexadas e fornecidas ao IP. O IP então coloca os dados em um pacote e os transmite. (SURYAWANSHI et al., 2019)

Figura 4 – Comparação entre modelo OSI e TCP/IP



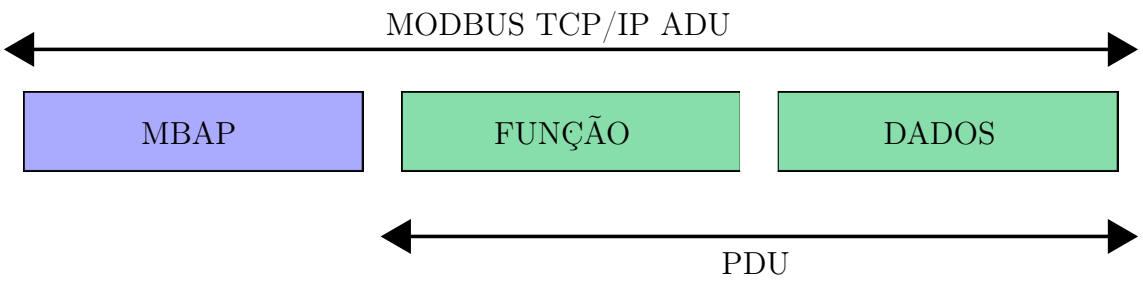
Fonte: Moraes (2010)

O protocolo Modbus TCP/IP é, simplificadamente, o Modbus RTU (*Remote Terminal Unit*) transmitido com um *wrapper* TCP/IP e enviado pela rede ao invés de um barramento serial. (GOMES, 2015)

A estrutura difere da convencional possuindo o cabeçalho de aplicação MBAP (*Modbus Application Protocol*), como visto na Figura 5, a PDU (*Protocol Data Unit*) do Modbus perde o CRC e com o acréscimo do *header* se torna o ADU(*Application Data Unit*). A relação mestre-escravo agora estabelecida através de cliente-servidor.

O endereço do escravo é substituído pelo identificador de unidade que é usado para se comunicar por meio de dispositivos como roteadores e *gateways* que usam um único endereço IP. Na tabela 1 são listados os elementos do cabeçalho de aplicação, o número de *bytes* e sua função no protocolo.

Figura 5 – Estrutura do Modbus TCP/IP



Fonte: Adaptado do Guia de Implementação Modbus (2006)

Tabela 1 – MBAP do protocolo Modbus TCP/IP

Elemento	Tamanho	Descrição
Identificador de Transação	2 *Bytes*	É usado para emparelhamento de transação entre as partes, o servidor Modbus copia na resposta o identificador de transação da solicitação
Identificador de Protocolo	2 *Bytes*	É usado para multiplexação dentro do sistema. O protocolo Modbus é identificado pelo valor 0
Tamanho da Mensagem	2 *Bytes*	É uma contagem de bytes dos campos seguintes, incluindo o Identificador de Unidade e o campo de dados
Identificador de Unidade	1 *Byte*	É usado para fins de roteamento dentro do sistema. Serve para identificar escravos conectados remotamente em linha serial ou outros barramentos

Fonte: Adaptado do Guia de Implementação Modbus (2006)

2.3 BLUETOOTH LOW ENERGY (RSSI)

As tecnologias de comunicação sem fio surgiram no final do século 19 com invenções como o telégrafo sem fio de Marconi. Desde então a possibilidade de se comunicar sem utilizar um meio físico visível de transmissão se tornou um campo de estudos com vasta aplicabilidade.

Segundo Siqueira (2006): "o *bluetooth* é um padrão de comunicação sem fio de curto alcance, baixo custo e baixo consumo de energia que utiliza tecnologia de rádio."

O *Bluetooth* também é dividido em 4 camadas, sendo elas:

1. Camada Física ou de Rádio Frequência (RF): Relacionada com os transceptores e filtros gaussianos.
2. Camada de Controle de Enlace: Estabelece como os dispositivos são associados.
3. Camada de Transporte Lógico: Gerencia os enlaces e dispositivos, implementa o Protocolo de Gerenciamento de Link (LMP).
4. Camada L2CAP: Atua como interface entre protocolos de camadas superiores e inferiores.

Em 2007 foi lançada uma nova tecnologia na qual o gasto de energia era da ordem de 10 a 20 vezes menor que a do bluetooth tradicional possuindo uma estrutura idêntica. Essa especificação seria chamada de *Bluetooth* 4.0 ou *Bluetooth Low Energy*. (ROCHOL, 2018)

Um comparativo entre as duas versões do *Bluetooth* e o *Wi-Fi* é mostrado na Tabela 2. Embora atue numa taxa de transmissão e alcance menor, o BLE possui um custo de implementação de hardware muito menor, o que permite que dispositivos de pequeno porte, como o localizador antifurto utilizado neste trabalho, sejam feitos o que se torna um atrativo para aplicações em Internet das coisas (IoT), como sensores em dispositivos de bem-estar e médicos, smartwatches e controle de iluminação residencial. (KACZMAREK; RUMINSKI; BUJNOWSKI, 2016). Além disso, os sinais BLE não são tão influenciados pelo ambiente devido a sua menor potência de transmissão e saltos de frequência. (RÖBESAAT et al., 2017)

Isso permite que sejam implementados sistemas em ambientes industriais, no que é chamado de Internet das Coisas Industrial (IIoT), onde a reflexão de sinal e a interferência eletromagnética tendem a ser maior. (ELIZALDE et al., 2021)

Tabela 2 – Comparação entre *Bluetooth* padrão, BLE e *Wi-Fi*

	Bluetooth	BLE	*Wi-Fi*
Taxa de transmissão	1 Mbps	720 kbps	50-200 Mbps
Frequência	2,4 GHz	2,4 GHz	2,4 e 5 GHz
Potência de Transmissão	10 dBm	1 dBm	20 dBm
Consumo de Energia	57 mA	15 mA	50-100 mA
Alcance	10 m	10 m	100 m
Custo de implementação	moderado	baixo	alto

Fonte: Adaptado de Röbesaat et al. (2017)

De acordo com a especificação *bluetooh* v.04[1], a faixa de frequência do BLE vai de 2,4 a 2,4835 GHz em 40 canais de 2 MHz, enquanto que o *Wi-Fi* utiliza 14 canais de 22 MHz. A Equação 2.1 indica as frequências centrais.

$$f_c = 2402 + k \cdot 2 \quad MHz \tag{2.1}$$

onde k = 0,. . . ,39.

A modulação é feita pela técnica *Gaussian Frequency Shift Keying* (GFSK) utilizando um filtro gaussiano, capaz de reduzir o ruído e perda de energia através de uma variação mais suave da frequência, isto é, a modulação GFSK atua em fase contínua. (LATHI; DING, 2012)

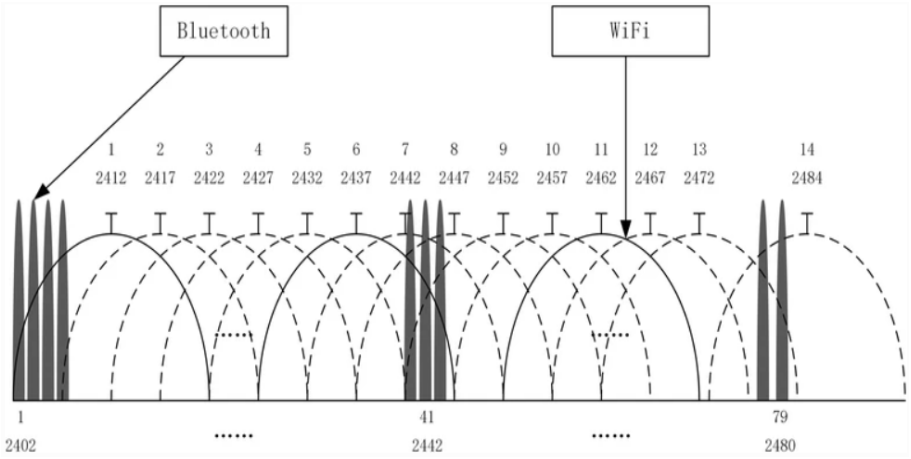
Outras tecnologias que utilizam o padrão IEEE 802.11 e 802.15 operam na mesma faixa de frequência, o que pode ocasionar perda dos dados. Para diminuir a interferência de outros sinais na mesma frequência é realizado o salto de frequência no qual o sinal muda constantemente de canal, a uma frequência de 1,6 kHz, para evitar colisões com outros

[1] https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737

pacotes. É ilustrado na Figura 6 como é feita a divisão entre o bluetooth convencional e o Wi-Fi na mesma frequência.

A tecnologia BLE utiliza o *Adaptive Frequency Hopping* (AFH), no qual a variação é feita de maneira inteligente, onde canais que possuem alto grau de ruído ou que já se encontram em uso por outro dispositivo tendem a ser evitados. A AFH permite que o caminho mais confiável seja escolhido ainda que as interferências mudem constantemente.

Figura 6 – Divisão do Espectro de Frequência



Fonte: Pei et al. (2017)

2.3.1 Potência RSSI

Na transmissão dos dados via BLE será utilizado o indicador de intensidade de sinal recebido (RSSI) para fazer a estimativa da distância que o localizador se encontra. Quando se trata da análise da distância Paterna et al. (2017) argumenta que existem duas principais formas de ser realizada: através de um mapeamento da perda de sinal de rádio propagado pela distância, e por meios de técnicas de "*fingerprinting*".

A primeira consiste em considerar que o decaimento do sinal ocorre de acordo com um modelo preestabelecido e utilizar sua equação característica para chegar ao resultado da distância. O grande problema nesse método é que escolher o melhor modelo nem sempre é uma tarefa fácil por causa da mudança nos parâmetros internos dos localizadores, além de que devido a múltiplos fatores como variação de temperatura, humidade, localização e obstáculos o mesmo modelo pode não servir para localizadores dispostos em diferentes espaços de um mesmo prédio.

O segundo consiste em realizar múltiplas medições em diferentes posições a fim de criar um mapa de variação RSSI para cada dispositivo. Este método possui uma exatidão muito maior que a primeira, por ser possível estimar os parâmetros de diminuição da potência, mas requer uma caracterização exaustiva do ambiente. Além disso, em caso de

variações como descritas anteriormente é necessário realizar uma nova bateria de medições para configurar o mapa corretamente.

Desse modo, uma abordagem mais próxima do segundo método foi realizada neste trabalho, onde se buscou criar uma curva característica capaz de determinar com certa precisão a distância a partir do valor do RSSI. De um modo geral a potência do sinal é dada por:

$$P(d) = P(d_0) - 10 \cdot \beta \cdot log_{10}(\frac{d}{d_0}) \tag{2.2}$$

Onde d é a distância que se deseja conhecer, d_0 é a distância de referência e P é a função de potência. A letra β é a constante de propagação do sinal que varia de acordo com o ambiente, normalmente para áreas abertas é aproximadamente 2. (KACZMAREK; RUMINSKI; BUJNOWSKI, 2016)

Alguns valores normalmente encontrados em diferentes ambientes podem ser vistos na Tabela 3.

Tabela 3 – Valor de β para diferentes ambientes

Ambiente	β
Campo Aberto	2
indoor Residencial	1,6 - 1,8
indoor Residencial com Barreiras	4 - 6
Industrial com Barreias	2 - 3

Fonte: Mekki, Bajic e Meyer (2019)

Fazendo $d_0 = 1$ temos A como valor de referência do sinal para um metro, e sendo $P(d) = RSSI$,

$$RSSI = A - 10 \cdot \beta \cdot log_{10}(d) \tag{2.3}$$

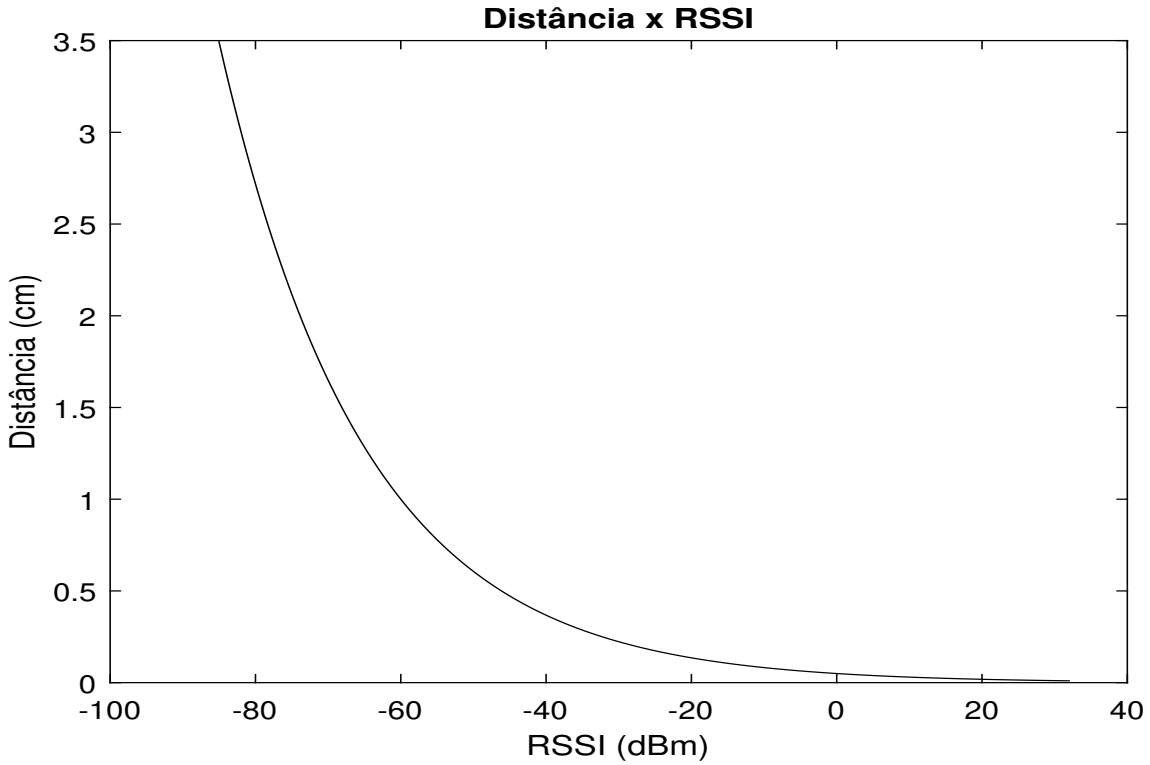
Rearranjando matematicamente a Equação 2.3 chegamos na Equação 2.4 que nos dá a distância em função do RSSI.

$$d = 10^{\frac{A-RSSI}{10 \cdot \beta}} \tag{2.4}$$

Assim, para um valor de $A = -60$ e $\beta = 2$, por exemplo, teríamos a curva característica de decaimento de sinal ilustrada na Figura 7. Em todos os experimentos realizados neste trabalho o menor valor de RSSI encontrado foi de -25 dBm, o que indica que mesmo

as melhores aproximações para a curva da figura ainda possuirão um elevado erro para pequenas distâncias até 10 centímetros, quando comparado de forma proporcional com distâncias longas acima de 3 metros.

Figura 7 – Exemplo de decaimento logarítmico de RSSI



Fonte: Autor

2.4 SOFTWARE SUPERVISÓRIO

A automação industrial tem evoluído de uma parte isolada da fábrica a um ambiente totalmente integrado e compartilhado com os demais sistemas. A integração e o monitoramento de todos os sistemas industriais se tornaram fundamentais para qualquer organização. (AZEVEDO; AZEVEDO, 2011)

Um sistema SCADA (*Supervisory Control and Data Acquisition*) é composto por um conjunto de dispositivos de aquisição de dados, supervisão e controle. Dentro do SCADA o supervisório é um software gerenciador de processos da planta, normalmente composto por telas sinóticas, tags informativas, alarmes e possibilidade de gerar relatórios[2]. São capazes de monitorar os dados advindos dos sensores e atuadores de diferentes fabricantes e representa-los graficamente sendo uma ferramenta extremamente útil no desenvolvimento industrial. Também é característica dos supervisórios a capacidade de se comunicar atra-

[2] https://www.hitecnologia.com.br/blog/o-que-e-um-sistema-scada/

vés de diversos protocolos e padrões. Além disso permitem que seus dados sejam guardados em bancos de dados para análises posteriores permitindo melhorias estratégicas no sistema como um todo e tomada de decisão. (AGHENTA; IQBAL, 2019)

Sistemas supervisórios são utilizados em larga escala na indústria por sua capacidade de controle sobre o sistema, ganho de qualidade e velocidade de produção, diminuição de custos e aproveitamento de dados. Através dele um único operador pode definir os parâmetros de funcionamento de toda a rede, desde alterações na abertura de válvulas mecânicas a valores binários a serem enviados para dispositivos eletrônicos.

Entretanto, as vantagens oferecidas são normalmente de difícil implementação, visto que sistemas modernos de automação chegaram a um grau de complexidade que intuição e experiência sozinhos não são capazes de coordenar devido ao enorme número de variáveis existentes em cada processo que exigem alta capacidade de processamento, tornando os sistemas SCADA ferramentas de alto custo para as empresas. (SILVA; CUNHA, 2017)

3 MATERIAIS E MÉTODOS

A metodologia empregada nos experimentos buscou encontrar os melhores parâmetros de decaimento da curva, a fim de traçar um mapa de como demais dispositivos reagiriam nas mesmas condições. Os materiais e algumas partes do algoritmo utilizado são descritos a seguir

3.1 LOCALIZADOR BLE

Os dispositivos *Bluetooth Low Energy* utilizados neste trabalho são localizadores de pequeno porte e baixo custo. O localizador é especialmente recomendada para servir como dispositivo contra perdas de objetos como chaves, celulares e mochilas. Existem diversos aplicativos mobile destinados a realizar a comunicação com o localizador, permitindo que sinalizações sonoras sejam emitidas em ambas as partes através do comando de um ou outro. Os dados do localizador apresentado na Figura 8 são mostrados na Tabela 4

Como os ambientes em que os experimentos foram feitos possuíam muitas barreiras e interferência advinda de máquinas a distância máxima em que foi possível obter o sinal RSSI foi de aproximadamente 10 metros. Cada aparelho capaz de utilizar a tecnologia BLE é identificado pelo seu número *Universally Unique Identifier* (UUID) formado por um número de 12 dígitos hexadecimais.

Tabela 4 – Especificações do localizador

Elemento	Especificação
Bateria	3 V (Lítio)
Distância Máxima em Locais Abertos	$\sim$ 23m
UUID Localizador 1	FC:58:FA:B5:09:C0
UUID Localizador 2	FC:58:FA:B3:81:1C

Fonte: Manual do usuário

O localizador possui uma antena interna que transmite os dados para o módulo ESP32. Dado a forma como o sinal é propagado precisa-se levar em consideração que os testes buscaram sempre manter a antena direcionada para a placa, do contrário o valor de RSSI seria reduzido drasticamente como ilustra a tabela 5.

3.2 MÓDULO ESP32

Desenvolvido pela *Espressif Systems*, o ESP32 é um microcontrolador amplamente utilizado em aplicações de IoT devido aos seus módulos Wi-Fi e *Bluetooth*. A lingua-

Tabela 5 – Diferença entre RSSI com diferente posição da Antena

Distância	Face a face		Costas com costas	
m	RSSI	Desvio	RSSI	Desvio
0,2	-56	15,64	-79	14,51
0,4	-63	6,68	-85	18,44
0,6	-66	0,77	-87	34,54
0,8	-66	3,57	-82	19,90
1,0	-66	17,26	-85	-28,51
2,0	-72	19,34	-86	19,44
3	-73	3,09	-93	1,82

Fonte: Ramirez et al. (2021)

Figura 8 – Localizador *Bluetooth* LE

Fonte: Autor

gem de programação utilizada é C++ por padrão, possibilitando o uso do ambiente de desenvolvimento integrado (IDE) Arduino para sua implementação.

A placa foi escolhida entre as demais disponíveis no mercado principalmente por atender as demandas do projeto e por ser de baixo custo. As principais especificações da placa são mostradas na Tabela 6.

Da mesma forma que o localizador BLE a placa possui uma antena *Bluetooth* interna que recebe os dados. Desse modo, o ESP32 também conta com um endereço UUID próprio, apesar de não ser visto durante o algoritmo. Isso também indica que a posição na qual se encontra influencia drasticamente nos valores de RSSI como visto na tabela 5.

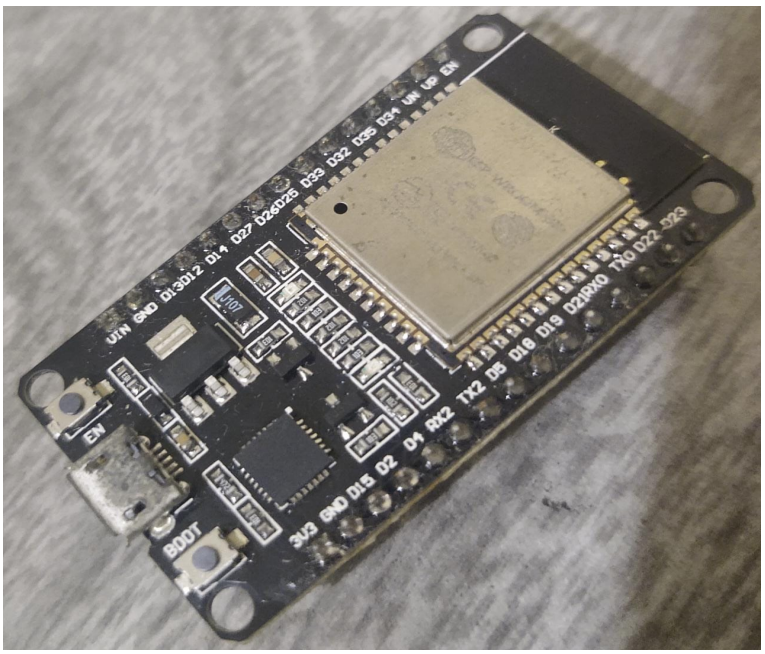
Ela também possui 2 conversores digitais analógicos, o que permite gerar ondas analógicas a partir de valores hexadecimais. Assim, é possível conectar a placa a um dispositivo sonoro e gerar sinais sonoros de intensidade e frequência variável. Além disso, pode ser

Tabela 6 – Especificações da ESP32

Elemento	Especificação
Processador	Dual Core 32-bit
Frequência de Operação	160 - 240 MHz
Memória SRAM	520 kB
Memória Flash	2 MB
Tensão de Alimentação	9 - 12 V / 3,3 V
GPIOs	36
ADC	10
DAC	2
Periféricos	PWM/UART/I2C/CAN2.0
Conector	USB Micro
Wi-Fi	sim
Bluetooth Low Energy	sim

Fonte: Adaptado de Maier, Sharp e Vagapov (2017) e Datasheet da placa

Figura 9 – Placa de desenvolvimento ESP32



Fonte: Autor

utilizada em uma gama de tarefas como codificação de voz e streaming de músicas. (GIULIANO et al., 2020)

Outras aplicações podem ser realizadas da união entre localizador e o módulo, por exemplo:

- Sensores de proximidade para acionar portões elétricos prediais e trancas magnéticas em laboratórios;
- Ligar dispositivos através de comandos no botão da tag;
- Enviar mensagens sonoras para deficientes visuais através de um buzzer conectado ao DAC da placa.

3.3 SUPERVISÓRIO SCADA-BR

Segundo o manual do ScadaBR (2010) o software é uma aplicação multiplataforma de código-fonte aberto e gratuita baseada em Java executado a partir de um servidor. O software oferece visualização de variáveis, gráficos, estatísticas, configuração dos protocolos, alarmes e telas sinóticas.

Dentre as diversas aplicações possíveis com o software destaca-se o uso em automação industrial e residencial(*smart buildings*), redes de saneamento e geração de energia. (FARIA et al., 2012)

Os principais elementos presentes no ScadaBR utilizados neste trabalho são os *data sources*, onde os dados são transmitidos, e *data points* que atuam como ponto de acesso e monitoramento dos dados de um determinado dispositivo a ele conectado, bem como definição de valores e visualização do banco de dados.

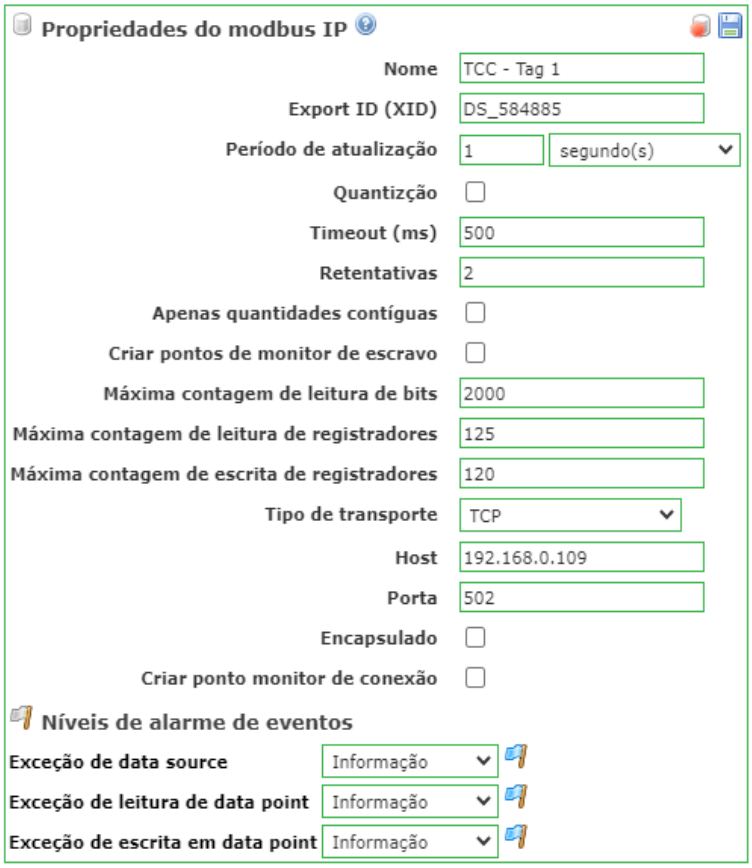
3.3.1 Data Source

A criação dos *data sources* é o primeiro passo na configuração da transmissão dos dados. O ScadaBR admite diversos protocolos para realizar a comunicação como BACnet I/P, HTTP, OPC DA, além do Modbus na versão serial e TCP/IP. É ilustrado na Figura 10 as propriedades do nó que será utilizado.

O período de atualização escolhido foi de um segundo, assim é possível coletar em média 10 valores dos sinais recebidos do BLE durante cada varredura dos nós para se obter a média. O período de *timeout* e o número de retentativas indicam qual o tempo máximo sem resposta do servidor e quantas vezes a conexão pode ser restabelecida até que se indique um erro.

Em Host é colocado o endereço IP do nó de comunicação criado pela ESP32 como será visto adiante. A porta deve ser a mesma escolhida na configuração do código na

Figura 10 – Configuração do DataSource



Fonte: Autor

placa. Também é permitido alterar o nível dos alarmes de eventos, variando desde status de informação a avisos urgentes de falhas.

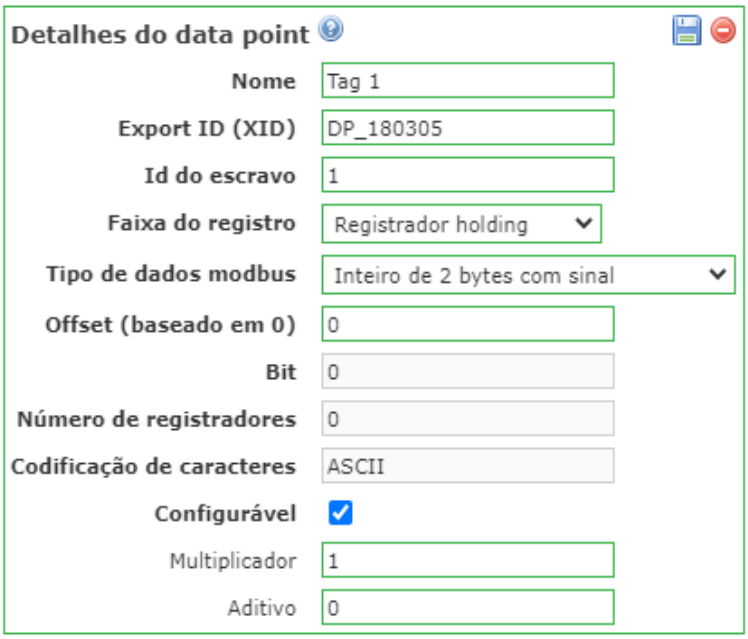
3.3.2 Data Point

Após o *data source* ser testada deve ser feita a criação dos data points que estarão ligados aos valores dos registradores da ESP32. Isto significa que quando a placa receber o valor da potência RSSI ou da distância, ele deverá ser fornecido ao ScadaBR através da sua *data point* correspondente. Um exemplo de *data point* pode ser visto na figura 11.

Em ID do escravo é colocado a identificação de qual dispositivo se deseja conectar naquele ponto, podendo utilizar até 240 escravos. Como nos experimentos realizados cada módulo se conectou com apenas um localizador o ID é 1. Em faixa de registro coloca-se o *range* sob o qual o valor será encontrado. Ele pode ser um status de bobina, status de entrada, registrador holding ou registrador de entrada. Em tipo de dados é escolhida a codificação dos valores recebidos. A faixa de registradores holding permite classificar como binários, inteiros de 2, 4 ou bits com ou sem sinal, floats de 4 ou 8 bits,

dentre outros. O ScadaBR permite que o valor seja multiplicado ou receba um acréscimo, ampliando a faixa de resposta.

Figura 11 – Criação da DataPoint de monitoramento de um Localizador

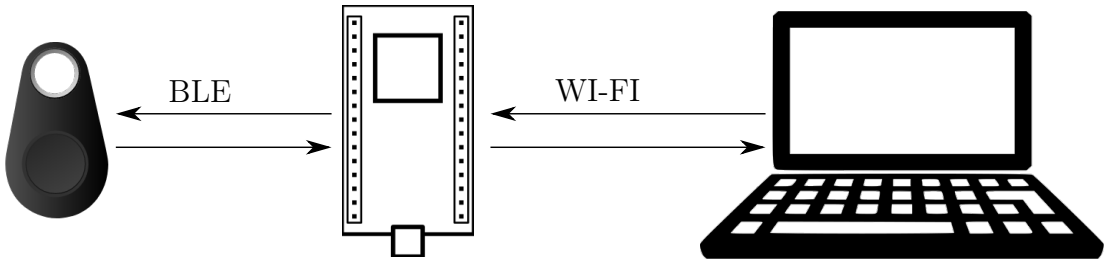


Fonte: Autor

3.4 ALGORITMO DE COMUNICAÇÃO

A transmissão dos dados é feita em duas etapas. Primeiro, o localizador envia o valor de RSSI através de BLE para o módulo ESP32. A placa então armazena os dados em um registrador e faz o envio para o ScadaBR através do *Wi-Fi*. A estrutura utilizada para comunicação é apresentada na Figura 12.

Figura 12 – Estrutura de envio



Fonte: Autor

Uma clara vantagem é que não existe necessidade de circuitos de condicionamento. Uma vez que o meio de transmissão é o ar os protocolos se encarregam de realizar os

procedimentos necessários para que não ocorram falhas. Entretanto, devido às limitações de hardware a taxa de atualização é de, pelo menos, 1 segundo.

3.4.1 Comunicação bluetooth

A comunicação *bluetooth* é realizada a partir de *scans* periódicos que obtêm o endereço e a potência do sinal. O código 1 apresenta a primeira etapa da configuração entre localizador e a placa. A biblioteca BLEDevice.h traz diversos métodos para uso de dispositivos adaptados ao bluetooth como as funções que serão utilizadas posteriormente. Em *ADRESS* é colocado o endereço UUID do localizador que lhe é único. Em *SCAN_INTERVAL* é definido o tempo que se deve esperar entre um *scan* e outro. O valor escolhido foi de 100 milissegundos, entretanto nos experimentos descritos no capítulo 4 e outros que foram realizados, calculam uma média, normalmente de 10 ou 20 valores, para ter uma resposta mais precisa, visto que o valor da potência varia frequentemente. Em *TARGET_RSSI* é definido o menor valor possível, em dBm, de sinal para que a rotina de *scan* possa ser feita. Por fim, em *MAX_MISSING_TIME* é colocado o tempo de *timeout*, i.é., o máximo período de espera que o scan aguarda até que não seja mais reconhecida uma ação, dado em milissegundos.

Código 1 – Configuração do Scan BLE

```
1 #include <BLEDevice.h>
2
3 #define ADDRESS "fc:58:fa:b5:09:c0"
4 #define SCAN_INTERVAL 100
5 #define TARGET_RSSI -90
6 #define MAX_MISSING_TIME 5000
```

O código 2 trata da rotina de obtenção do RSSI. O método *getAdress()* guarda o valor de cada dispositivo *bluetooth* e o compara com o endereço da tag colocada anteriormente, a função recebe o nome de qualquer aparelho conectado, como celulares e computadores. Caso a tag esteja na área de -90 dBm o código ativa a condição e manda o *scan* parar em *getScan()->stop()*. Então é guardado o valor de rssi com *getRSSI()*. O valor pode ser exibido na serial como mostra o código como também pode ser guardado em um registrador *holding* para ser enviado para o ScadaBR.

Código 2 – Obtenção do RSSI

```
1 if(advertisedDevice.getAddress().toString() == ADDRESS)
2         {
3             found = true;
4             advertisedDevice.getScan()->stop();
5             rssi = advertisedDevice.getRSSI();
6             Serial.print("RSSI:␣");
7             Serial.println(rssi);
8         }
```

3.4.2 Comunicação Modbus TCP/IP via WI-FI

A segunda parte do algoritmo consiste em realizar a conexão entre ESP32 e ScadaBR através do protocolo Modbus TCP/IP. Como visto na seção 2.2.1 há diferenças na estrutura dos protocolos, sendo preciso criar um canal de comunicação por onde os dados serão enviados via Wi-Fi. Por isso, como mostra o código 3, é preciso que a placa se conecte ao Wi-Fi local para criar um endereço de IP próprio que servirá como nó de comunicação com o supervisório, o que pode ser visto na Figura 13. O endereço IP criado não é escolhido pelo usuário.

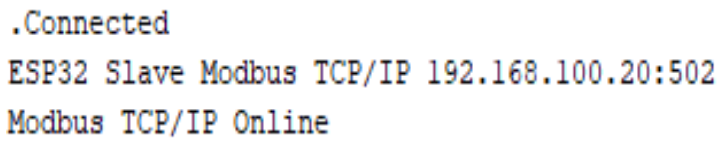
Código 3 – Acesso a rede Wi-Fi local

```
1 #include <WiFi.h >
2
3 //Definir nome e senha do WiFi que vai acessar
4 const char* ssid = "TP-LINK_9291E8";
5 const char* password = "";
6 int ModbusTCP_port = 502;
```

Em *ssid* é colocado o nome da rede e em *password* a senha. A *ModbusTCP_port* é a porta utilizada pelo servidor para fazer a comunicação, é interessante que seja escolhida uma porta que não seja normalmente utilizada para outros fins, como é o caso da porta 8081 que hospeda o canal nos experimentos realizados.

O endereço IP criado será então utilizado na configuração do *Data Source* do ScadaBR. No código 4 é configurado o *MBAP* como mostrado na Tabela 1 . *TID* é o identificador de transação, *PID* é o identificador de protocolo, *LEN* é o tamanho da mensagem, *FUNC* é o código da função, *REGISTER_START* é o endereço do primeiro registrador utilizado e *REGISTER_NUMBER* o número de registradores que serão utilizados.

Figura 13 – Endereço IP criado pela ESP



Fonte: Autor

Código 4 – Definição do Cabeçalho do Modbus

```
#define MB_TCP_TID 0
#define MB_TCP_PID 2
#define MB_TCP_LEN 4
#define MB_TCP_UID 6
#define MB_TCP_FUNC 7
#define MB_TCP_REGISTER_START 8
#define MB_TCP_REGISTER_NUMBER 10
```

Por fim, é apresentado no código 5 a implementação da função de leitura dos registradores. Em *WordDataLength* é colocado o número de registradores que serão lidos após o registrador que foi listado como primeiro. *ByteDataLength* indica o número de bytes utilizados e é sempre igual a duas vezes o número de registradores, visto que cada um usa 2 *Bytes*. Em seguida é gravada a mensagem e enviada para o servidor.

Código 5 – Função de Leitura dos Registradores

```
case MB_FC_READ_REGISTERS: // 03 Funcao de leitura
ByteDataLength = WordDataLength * 2;
ByteArray[5] = ByteDataLength + 3;
ByteArray[8] = ByteDataLength;
for(int i = 0; i < WordDataLength; i++)
{
ByteArray[ 9 + i * 2] = highByte(MBHoldingRegister[Start + i])
   ;
ByteArray[10 + i * 2] = lowByte(MBHoldingRegister[Start + i]);
}
MessageLength = ByteDataLength + 9;
client.write((const uint8_t *)ByteArray,MessageLength);
 byteFN = MB_FC_NONE;
 break;
```

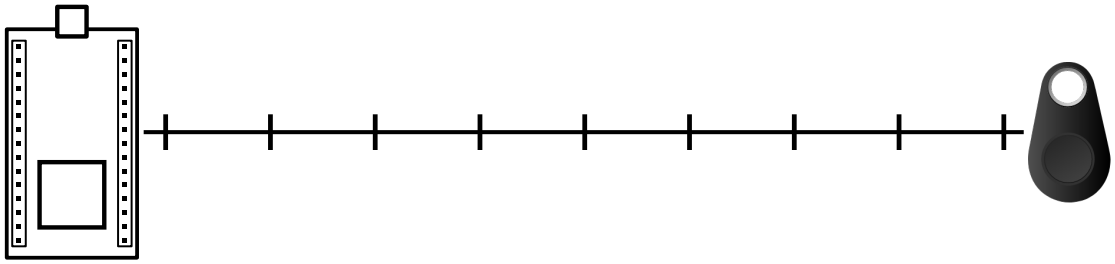
A partir dos códigos acima é possível estabelecer a conexão indireta entre o localizador BLE e o supervisório ScadaBR. Algumas alterações foram feitas momentaneamente para fazer cálculos específicos como valor médio de RSSI ou a distância em que o localizador se encontrava.

4 RESULTADOS

4.1 EXPERIMENTO 1

O primeiro experimento foi realizado em ambiente controlado com baixa interferência eletromagnética a fim de obter os parâmetros da curva de decaimento do sinal RSSI. O módulo ESP32 foi fixado em um ponto enquanto o localizador era movimentado em linha reta a distâncias específicas. Em cada ponto de medição são realizadas 100 medições do valor de RSSI do sinal *Bluetooth*, num intervalo entre si de 100 milissegundos, e obtido o valor médio. O localizador é então movido para a próxima distância e o processo de medição é refeito. A distância entre os componentes varia entre 0 e 3 metros. Como mostrado na Figura 7 a variação de RSSI tende a ser maior para pequenas distâncias, diminuindo gradualmente conforme o objeto se afasta, por essa razão os pontos de medição não estão uniformemente distribuídos, havendo uma diferença entre 25 centímetros nos primeiros valores e 50 centímetros a partir de 1,5 metros, o mesmo método foi adotado em Ramirez et al. (2021).

Figura 14 – Modelo do experimento 1



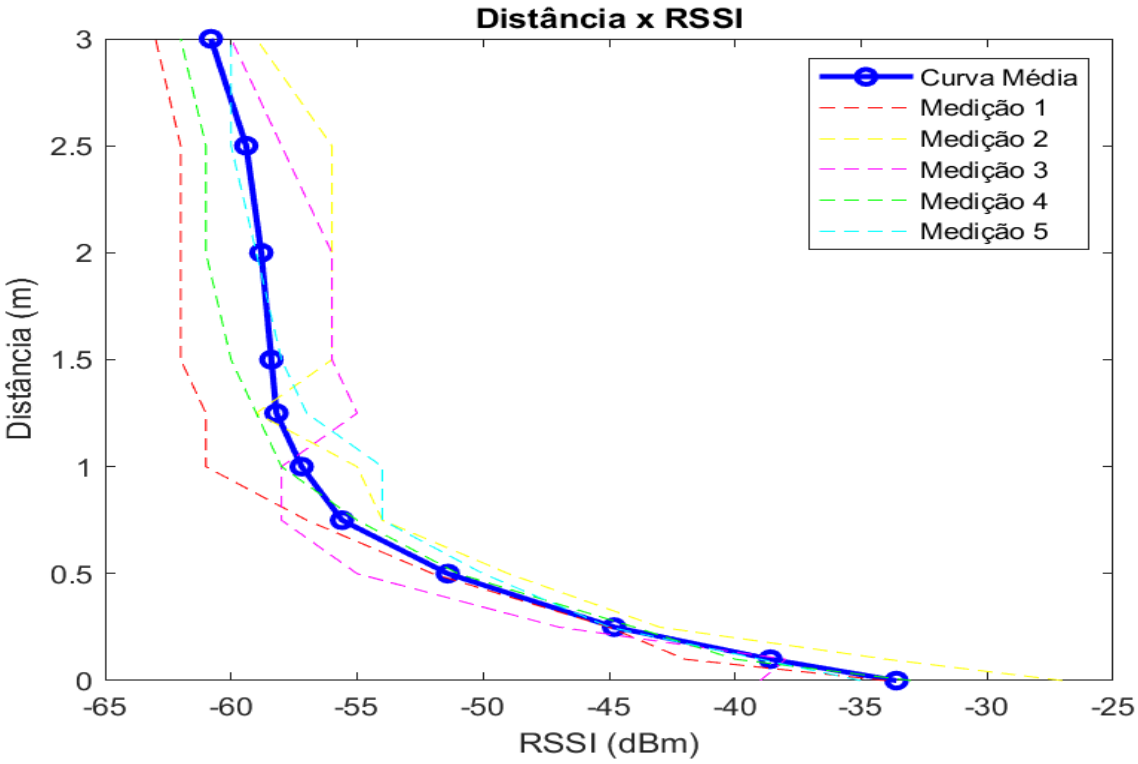
Fonte: Autor

Depois que todos os valores são coletados o processo é repetido. Após 5 medições completas é calculada a média e o desvio padrão do valor RSSI. Na Tabela 7 são apresentados os valores medidos, as grandezas das medições são dadas em dBm.

Com os valores da tabela foi feita a análise com o software MATLAB para comparar os dados encontrados nos experimentos com a referência. A Figura 15 mostra o gráfico das curvas do RSSI medido, onde a média está em azul forte.

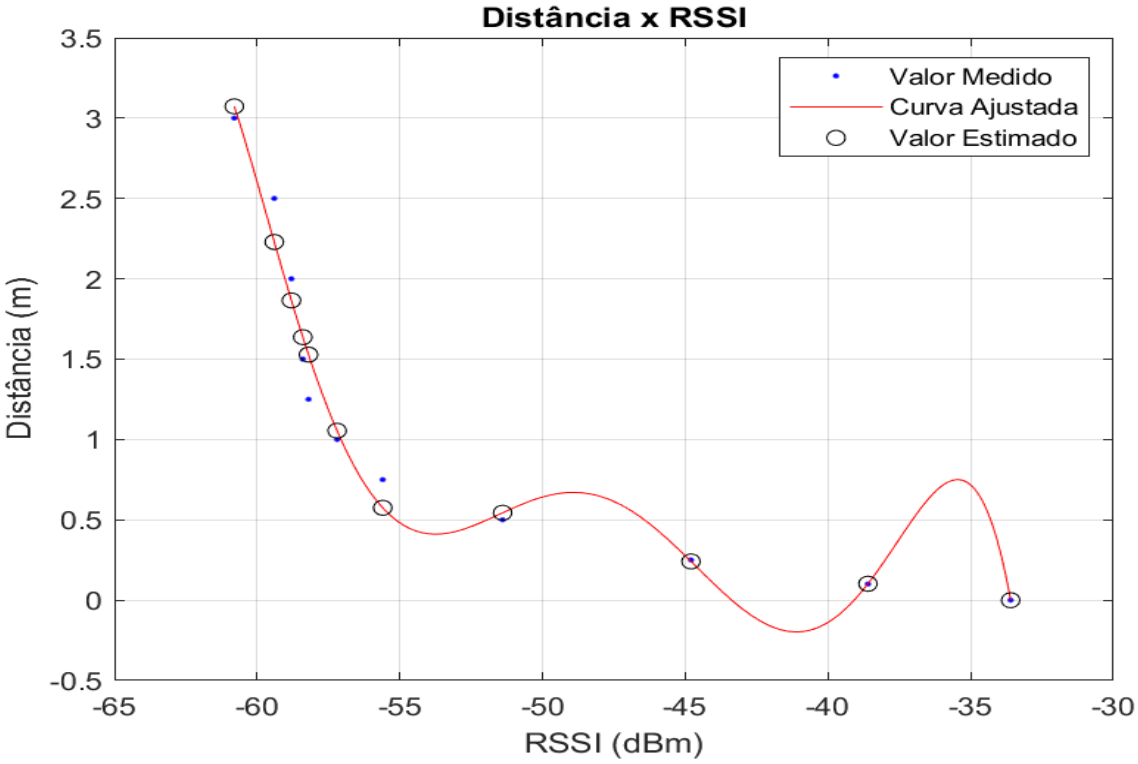
A partir dos valores médios obtidos foi feito um algoritmo para traçar uma curva de decaimento aproximada e obter seus parâmetros. Para verificar a melhor forma de aproximar a curva medida para os valores obtidos e fazer com que as distâncias de experimentos futuros pudessem ser medidas foi utilizada a função fit e foram feitas 5 curvas, sendo 4 polinomiais e 1 exponencial. As Figuras de 16 a 19 apresentam funções polinomiais aproximadas para o menor erro em relação aos valores medidos.

Figura 15 – Curvas obtidas da tabela 7



Fonte: Autor

Figura 16 – Curva Fit polinomial de 6ª ordem



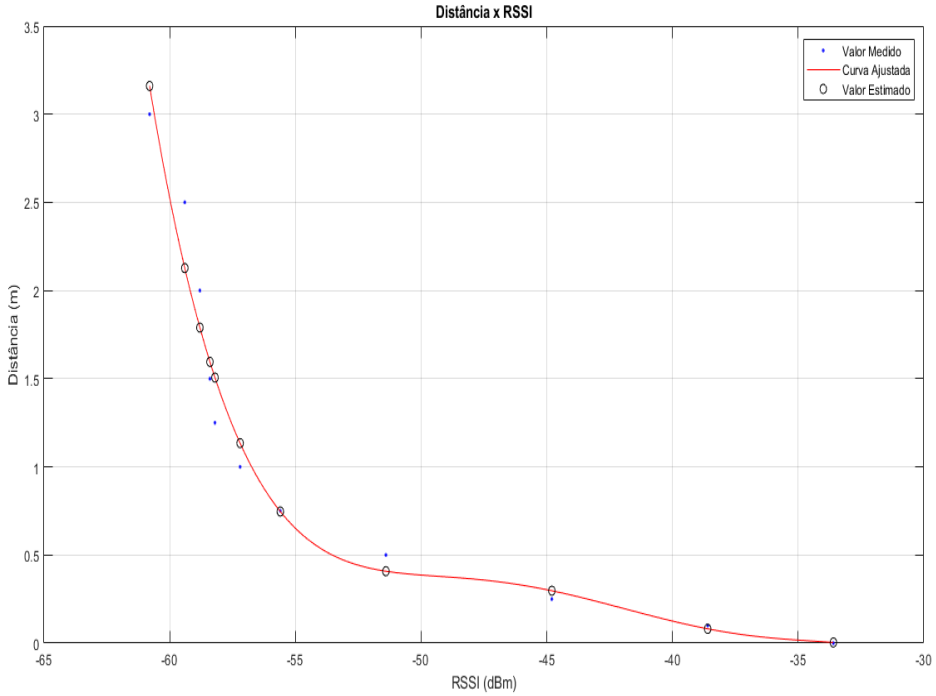
Fonte: Autor

Tabela 7 – RSSI medido em ambiente controlado

Distância (m)	1ª	2ª	3ª	4ª	5ª	Média	Desvio Padrão
0	-34	-27	-39	-33	-35	-33,6	4,336
0,10	-42	-34	-38	-40	-39	-38,6	2,966
0,25	-45	-43	-47	-44	-35	-44,8	1,483
0,50	-52	-49	-55	-51	-50	-51,4	2,302
0,75	-57	-54	-58	-55	-54	-55,6	1,817
1,00	-61	-55	-58	-58	-54	-57,2	2,775
1,25	-61	-59	-55	-59	-57	-58,2	2,280
1,50	-62	-56	-56	-60	-58	-58,4	2,608
2,00	-62	-56	-56	-61	-59	-58,8	2,775
2,50	-62	-56	-58	-61	-60	-59,4	2,408
3,00	-63	-59	-60	-62	-60	-60,8	1,643

Fonte: Autor

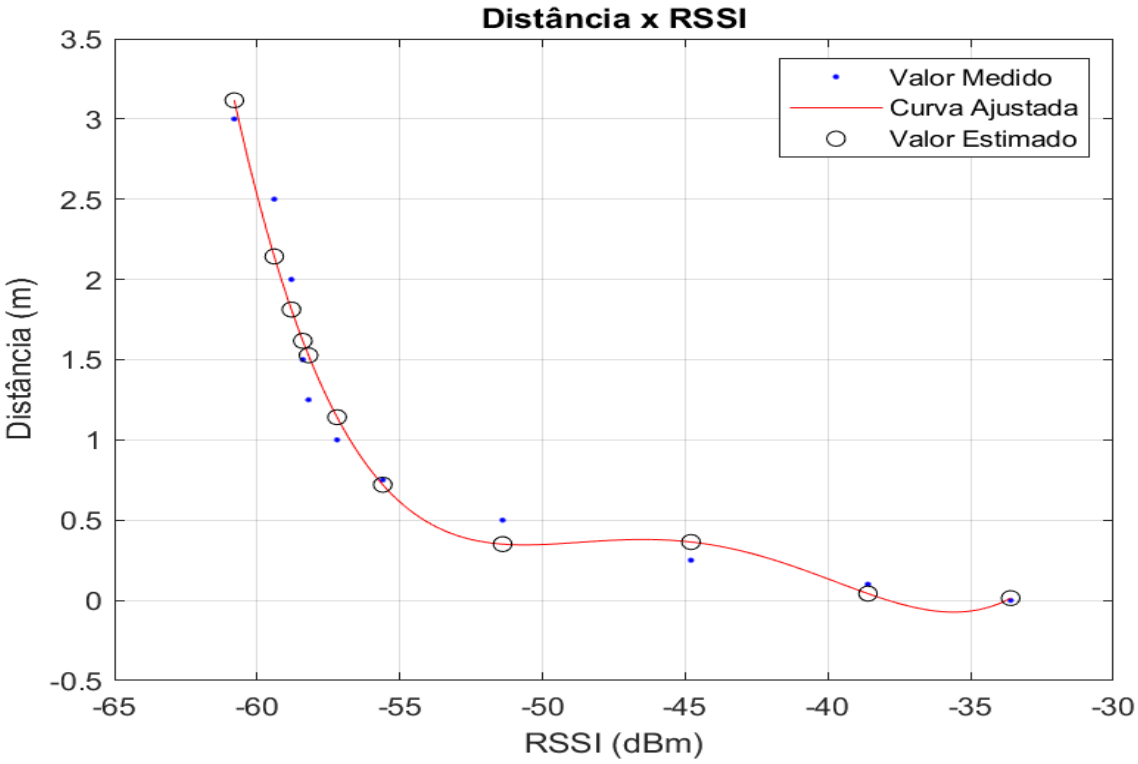
Figura 17 – Curva Fit Polinomial de 5ª ordem



Fonte: Autor

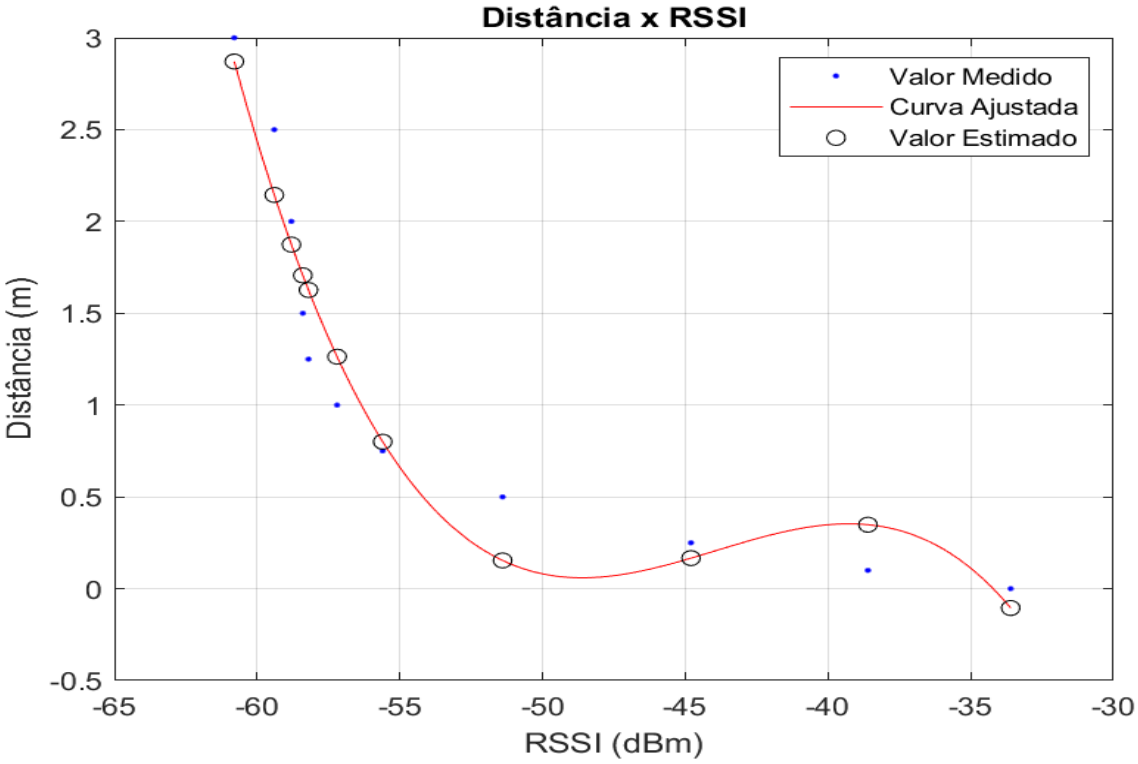
As curvas acima então precisam ser comparadas com a Figura 7. Como pode ser visto a melhor aproximação é a que utiliza 5 parâmetros pois possui um decaimento mais suave e seu erro médio é baixo quando comparada com a função média da Figura 15, portanto ela servirá como base para as análises posteriores. A Equação 4.1 representa o gráfico exposto na Figura 17.

Figura 18 – Curva Fit polinomial de 4ª ordem



Fonte: Autor

Figura 19 – Curva Fit polinomial de 3ª ordem



Fonte: Autor

$$d = p1 \cdot r^5 + p2 \cdot r^4 + p3 \cdot r^3 + p4 \cdot r^2 + p5 \cdot r + p6 \tag{4.1}$$

Onde,

$p1 = -1,887 \cdot 10^{-6}$
$p2 = -3,95 \cdot 10^{-4}$
$p3 = -0,03274$
$p4 = -1,342$
$p5 = -27,24$
$p6 = -219$

Foi estabelecido um novo método de aproximação, dessa vez utilizando uma função exponencial. É apresentado na Figura 20 a curva exponencial de segunda ordem. O número de parâmetros foi escolhido a fim de obter o menor erro e a melhor comparação com a figura e a Equação 4.2 os seus parâmetros. O número de elementos em cada função foi escolhido a partir do menor erro quadrático possível.

$$d = a \cdot exp(e1 \cdot r) + b \cdot exp(e2 \cdot r) \tag{4.2}$$

Onde,

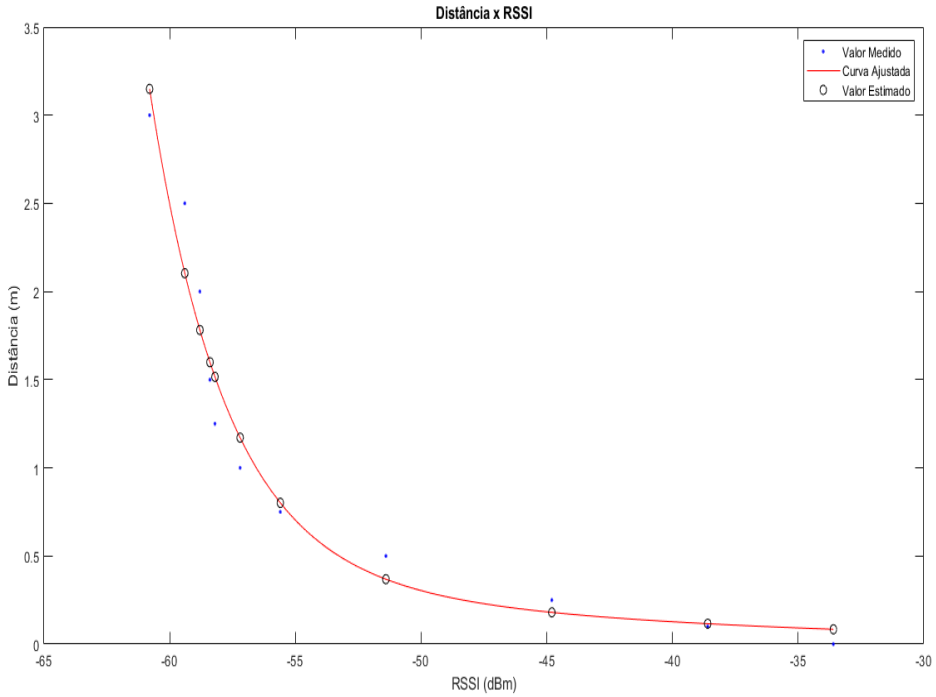
$a = 4,156 \cdot 10^{-9}$
$b = 0,01065$
$e1 = -0,3338$
$e2 = -0,06149$

Também pode-se ver na Tabela 7 o desvio padrão dos valores medidos, percebe-se que para distâncias de até 25 centímetros a variação tende a ser maior. Devido a isso as curvas tendem a não representar distâncias muito pequenas com precisão.

A fim de encontrar uma curva que se aproximasse da referência teórica dada pela Equação 2.4 foram feitos diversos testes com diferentes valores de β e calculados seus respectivos valores da raiz do erro quadrático médio (RMSE) e desvio padrão (σ). Os resultados obtidos para ambas as curvas podem ser vistos na Tabela 8.

Da tabela verifica-se que a aproximação polinomial é mais exata e precisa para valores de β próximos de 1, enquanto que para a exponencial acontece quando β tende a 1,75. O valor de A é sempre igual a 56,6 visto que ele é obtido a partir da melhor aproximação da medição em 1 metro das curvas fit. Sendo assim, a melhor aproximação encontrada pode ser vista na Figura 21na cor azul, no qual é utilizado um valor de $\beta = 1,35$. Para esta aproximação o erro e o desvio padrão encontrados estão descritos na Tabela 9.

Figura 20 – Curva Exponencial Estimada em Ambiente Controlado



Fonte: Autor

Tabela 8 – Erro Médio e Desvio Padrão para diferentes valores de β

Parâmetro	β	Polinomial	Exponencial
RMSE	1,00	2,97	4,73
σ	1,00	2,75	4,16
RMSE	1,25	3,10	3,64
σ	1,25	3,17	3,06
RMSE	1,5	4,41	2,79
σ	1,5	4,63	2,27
RMSE	1,75	6,19	2,47
σ	1,75	6,45	2,16
RMSE	2,00	8,14	2,85
σ	2,00	8,42	2,82

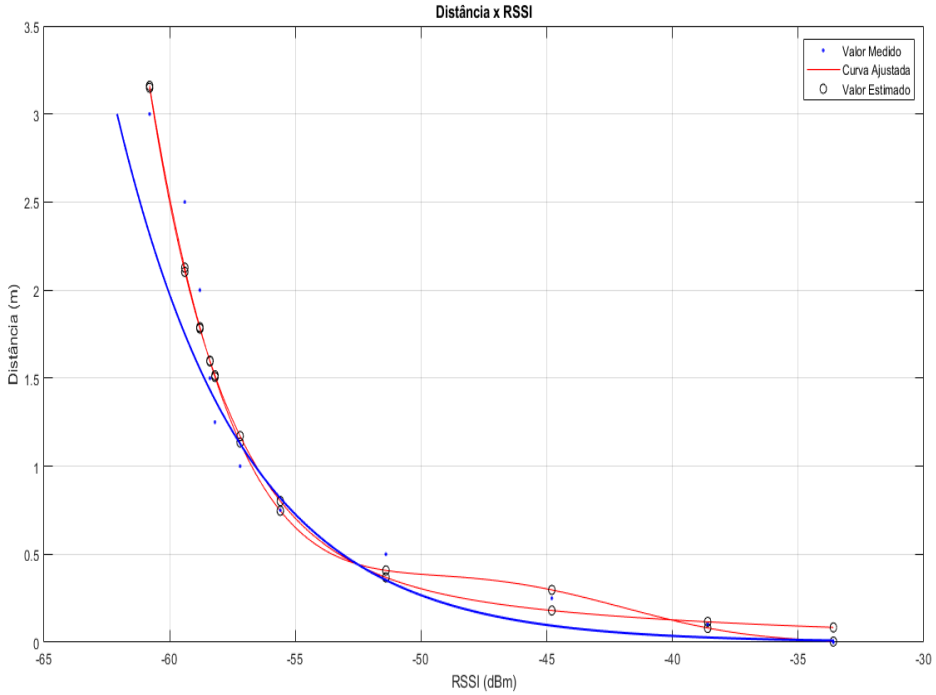
Fonte: Autor

Tabela 9 – Erro e Desvio Padrão para melhor aproximação teórica

	Polinomial	Exponencial
RMSE	3,29	3,44
Desvio Padrão	3,40	2,87

Fonte: Autor

Figura 21 – Comparativo entre curvas medidas e teórica estimada



Fonte: Autor

Conforme exposto na Tabela 3 o valor de β para ambientes residenciais é normalmente entre 1,6 e 1,8 metros, o que demonstra que o resultado obtido não está longe da realidade. Também se verifica que curva fit exponencial tende a se aproximar mais desse valor, o que a torna mais útil em cálculos de distâncias utilizando *bluetooth*.

Pode-se concluir através dos valores de desvio padrão que as curvas estão dentro da faixa esperada, tornando-as boas estimativas da distância real do localizador com uma resolução de poucos centímetros. Quando comparado a outras tecnologias, como exposto na seção 2.3, o *Bluetooth* LE ganha vantagem pelo seu baixo consumo de energia e baixo custo de implementação.

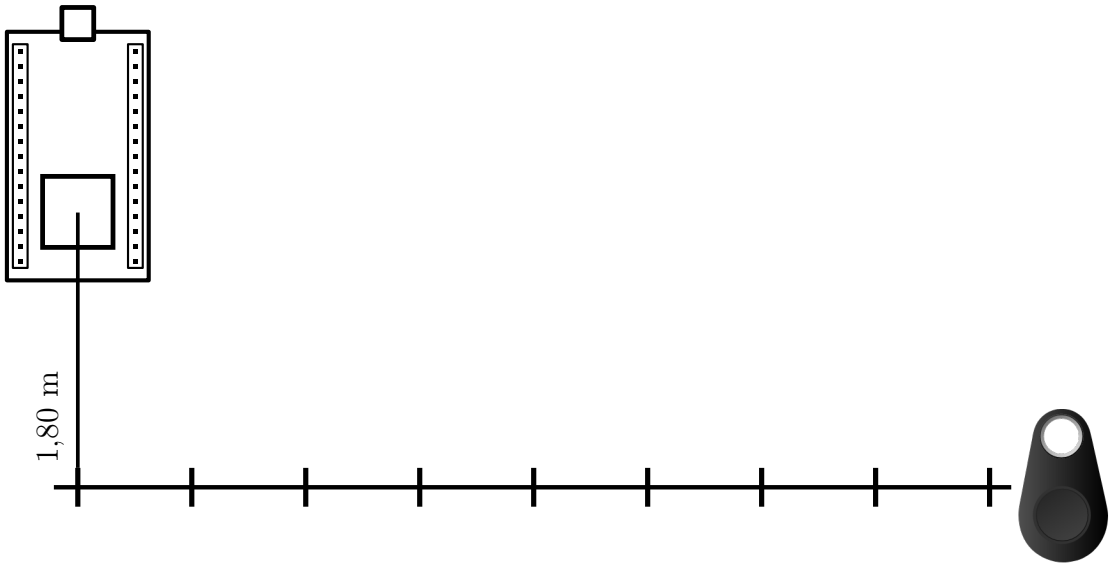
4.2 EXPERIMENTO 2

O segundo experimento foi realizado no Laboratório de Eficiência Energética e Hidráulica em Saneamento (LENHS/UFPB). O objetivo desse experimento é realizar o monitoramento do sinal RSSI através do ScadaBR.

Para fins de movimentação foi preciso fazer um arranjo em como realizar as medições de distância, como ilustrado no modelo do experimento na Figura 22.

Na Figura 23 é mostrado o ambiente em que os testes foram conduzidos. Os *post-its* amarelos no chão são demarcações com distância de 50 centímetros entre si, sendo o mais

Figura 22 – Modelo do experimento 2



Fonte: Autor

à esquerda da imagem a distância de 0 metros e o mais à direita a de 4 metros. A distância real entre o localizador e o módulo é dada pela fórmula de Pitágoras conforme a Equação 4.3, sendo d a distância percorrida em linha reta com as marcações, d_0 a distância de 1,8 metros entre o ponto $d = 0$ e um conjunto motobomba, assinalado com a seta vermelha na imagem, que é descrito no experimento 3. D é a distância real entre os componentes.

$$D = \sqrt{d^2 + d_0^2} \tag{4.3}$$

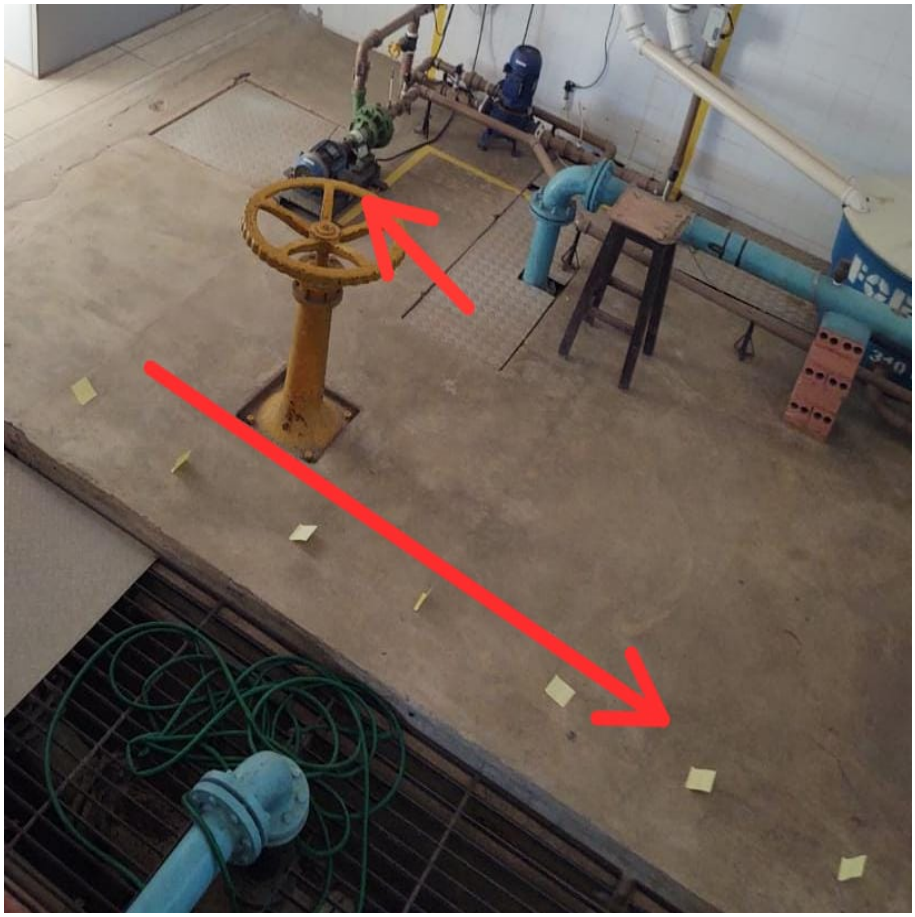
A placa foi colocada sobre o conjunto motobomba para verificar se a interferência eletromagnética seria forte o suficiente para causar uma mudança significativa no RSSI. O teste foi repetido posteriormente com o localizador BLE.

Foi observado que o localizador é bem mais sensível a interferência, muitas vezes chegando a falhar e enviar dados aleatórios mesmo com a bomba desligada, resultados similares foram encontrados ao utilizar uma trena para fazer a demarcação nos primeiros experimentos. Como visto na literatura objetos que obstruem a passagem do sinal fazem com que falhas possam ocorrer no protocolo, o metal, por exemplo, possui maior influência que outros materiais, como concreto e o corpo humano. Ademais, devido a potência ser maior no dispositivo transmissor a perda de sinal é maior que no receptor.

Com a conexão estabelecida com o ScadaBR é possível monitorar os dados. Na Figura 24 se ilustra os valores salvos no banco de dados do supervisório durante 10 minutos. Assim, foi possível recuperar valores antigos para construir a Tabela 10.

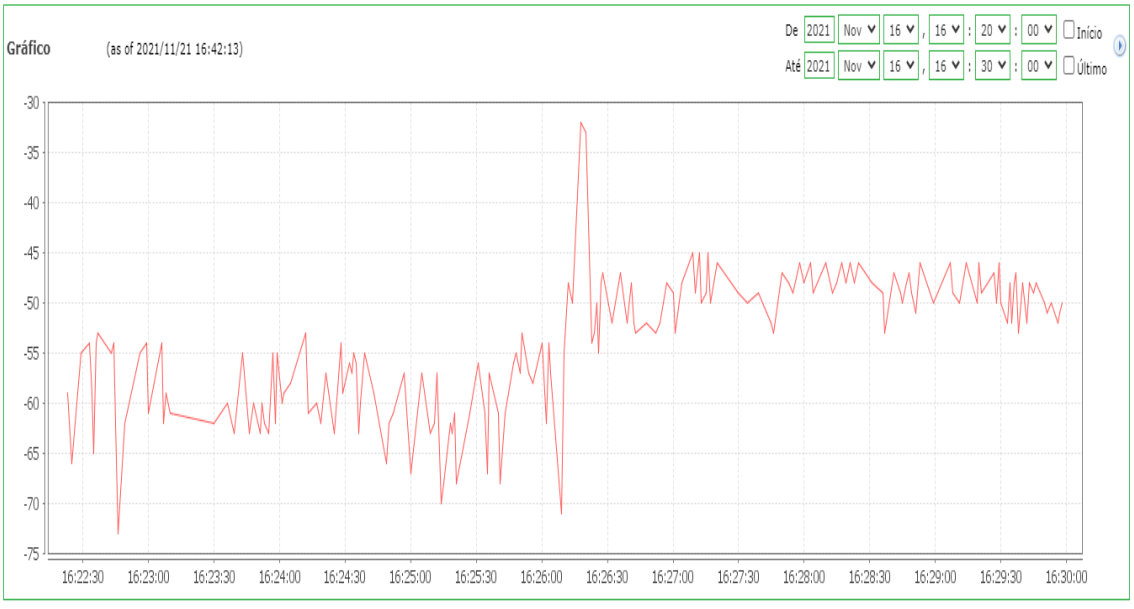
Da Tabela 10 verifica-se que há distâncias que possuem um valor médio de sinal maior que outros com distância menor, como por exemplo com d_0 entre 3,08 e 3,50 metros. Esse resultado é natural, uma vez que conforme a distância aumenta a variação no RSSI tende

Figura 23 – Ambiente dos experimentos no LENHS



Fonte: Autor

Figura 24 – Dados do RSSI monitorados no ScadaBR



Fonte: ScadaBR

a diminuir. A partir da curva teórica estimada na Figura 21 é mostrado que para esse mesmo intervalo a variação em dBm é de apenas -2, o que sequer compensa a variação natural do sinal. Para garantir resultados mais precisos seria necessário realizar mais pontos e medições para que a curva se tornasse similar à referência teórica.

Tabela 10 – RSSI medido no LENHS

d teórico(m)	D experimental (m)	1ª	2ª	3ª	Média RSSI	Desvio Padrão
0	1,8	-60	-63	-62	-61,67	1,53
0,50	1,87	-62	-65	-66	-64,33	2,08
1,00	2,05	-62	-69	-63	-64,67	3,79
1,50	2,34	-64	-64	-68	-65,33	2,31
2,00	2,69	-67	-63	-63	-64,33	2,31
2,50	3,08	-66	-70	-66	-67,33	2,31
3,00	3,50	-64	-68	-64	-65,33	2,31
3,50	3,93	-70	-69	-69	-69,33	0,58
4,00	4,38	-72	-74	-73	-73	1

Fonte: Autor

Também se pode concluir que, pelo fato do ambiente possuir objetos metálicos que possam refletir o feixe eletromagnético, a variação tende a ser menos precisa do que em um ambiente controlado. Apesar disso, para fins práticos chegou-se a um resultado positivo, no qual é possível ter uma precisão de até um metro da posição do localizador.

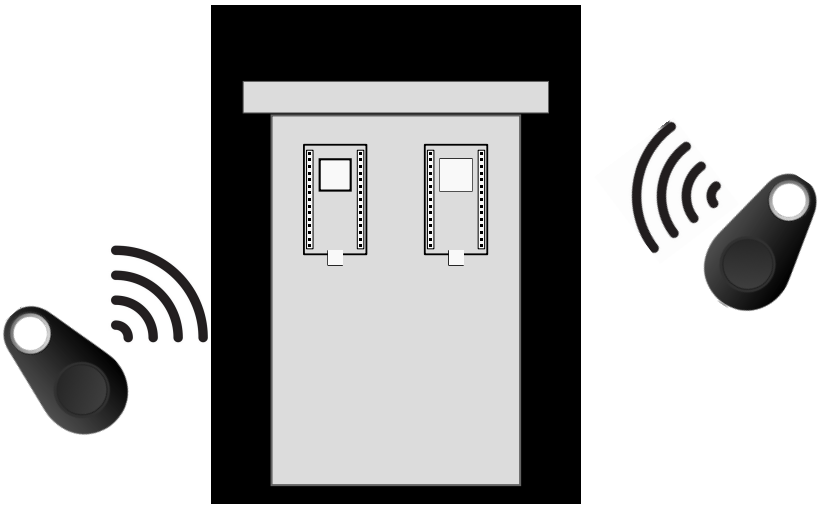
4.3 EXPERIMENTO 3

O terceiro experimento, também realizado no LENHS, consiste em colocar dois módulos ESP32 para receber os valores de RSSI de dois localizadores transmitindo, em condições normais, o mesmo valor, conforme a Figura 25. A distância entre os módulos e os localizadores não é necessariamente a mesma, uma vez que as condições de hardware do dispositivo BLE podem fazer as potências transmitidas serem bem diferentes. Outros testes realizados verificaram uma diferença de até 10dBm entre os localizadores para uma mesma distância de 1 metro.

A razão de se utilizar duas placas ao invés de apenas uma é que a biblioteca utilizada só permite uma única conexão. Outras bibliotecas foram testadas, porém não se chegou a um resultado aceitável. Um dos localizadores é colocado próximo a um conjunto motobomba para verificar a influência do ruído eletromagnético de uma máquina sobre a potência do sinal. O conjunto é formado por uma bomba centrífuga conectado a um motor de indução trifásico. As especificações do motor estão na Tabela 11.

Como observado no experimento 2 precisa-se de cautela no quão próximo os componentes podem estar do motobomba sem que haja perda de informações no protocolo, do

Figura 25 – Modelo do terceiro experimento



Fonte: Autor

Tabela 11 – Especificações do motor

Elemento	Especificação
Fabricante	WEG
Potência	380 V
Frequência	60 Hz
Corrente Nominal	4,86 A
Rotação Nominal	3450 rpm

Fonte: Autor

contrário não será possível verificar se houve interferência no RSSI. Por esta razão o BLE mais próximo da máquina estava a uma distância de cerca de 30 cm dela.

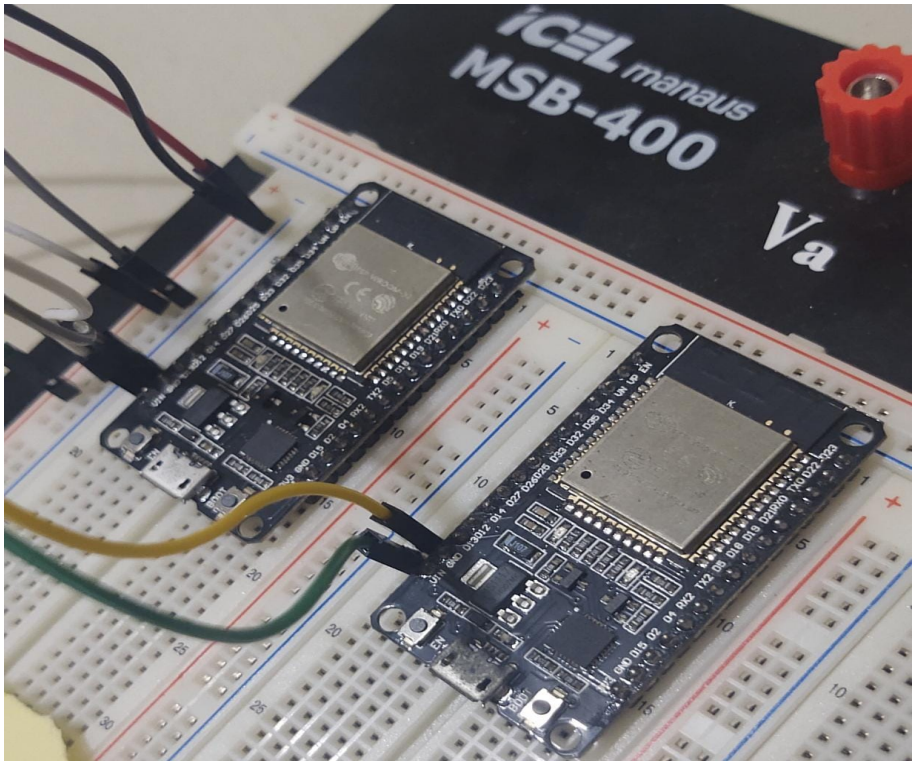
A alimentação das placas foi feita por uma fonte de alimentação de 12 V da GMI. A ligação entre os componentes é feita por uma protoboard e *jumpers* de conexão como é mostrado na Figura 26. Como não há conexão com a serial da IDE Arduino foi preciso realizar o monitoramento através do ScadaBR. Cada microcontrolador ESP32 gera um endereço de IP próprio o que requer a criação de dois *datasources*. Os gráficos dos *datapoints* podem ser vistos na Figura 27.

A principal dificuldade está em encontrar um ponto em que os valores médios dos dois localizadores sejam similares pois as variações do RSSI são altas e mesmo realizando a análise sobre o mesmo ponto haveria divergências entre as medidas.

O ScadaBR permite exportar os dados de seus *data points* através de um arquivo JSON que é então convertido para uma planilha no formato .XML. Na Tabela 12 são apresentados os valores médios e desvios dos dados coletados a partir do banco de dados do ScadaBR para um período de 10 minutos.

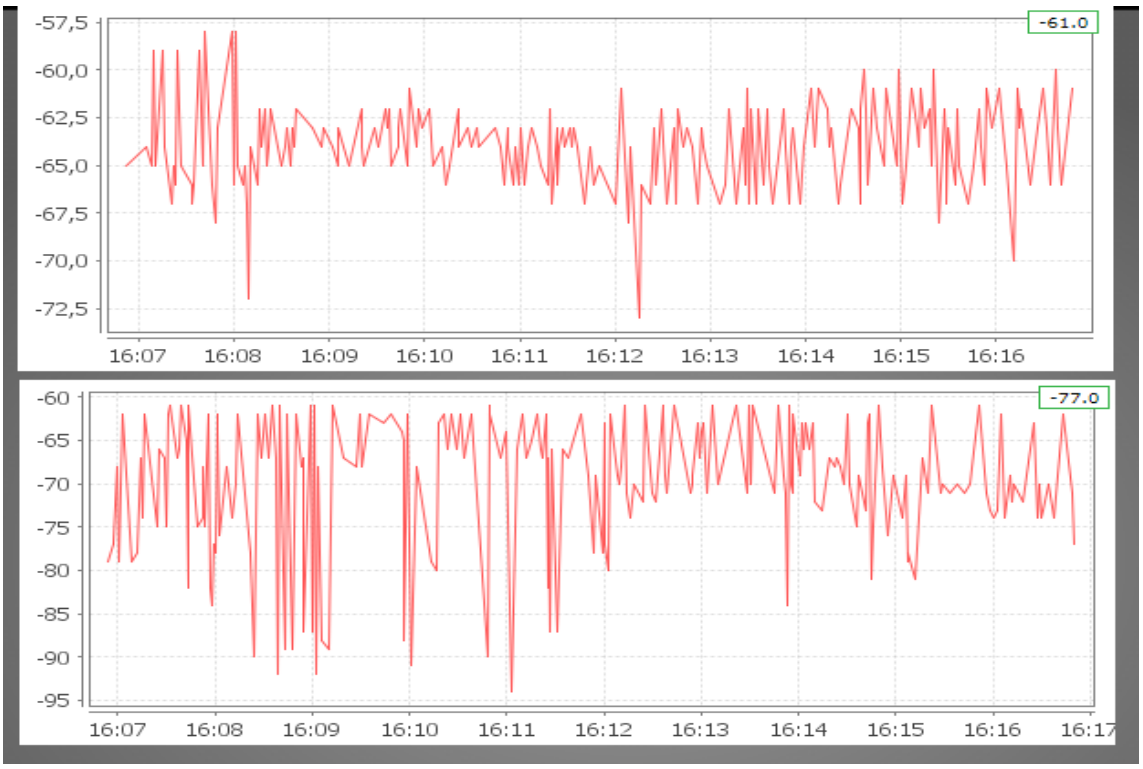
Apesar do localizador mais próximo do motobomba ser menor, os dois dispositivos

Figura 26 – Montagem utilizada para o terceiro experimento



Fonte: Autor

Figura 27 – Monitoramento dos localizadores no ScadaBR



Fonte: ScadaBR

Tabela 12 – Valor Médio e desvio padrão encontrados

	Localizador 1	Localizador 2
Distância (m)	1,20	1,10
Média RSSI	-63,93	-69,93
Desvio Padrão	2,265	7,700

Fonte: Autor

possuem um valor médio de RSSI razoavelmente próximo. O índice de desvio padrão no localizador 2 apesar de maior ainda está na faixa proposta na Tabela 5.

A interferência causada pela máquina não parece ser relevante sobre o valor de RSSI quando comparado com os resultados da literatura. Entretanto, devido ao seu alto grau de variação do BLE ainda pode ser pouco viável para aplicações industriais que necessitem de uma alta precisão.

A partir do exposto nos experimentos conclui-se que o BLE funciona bem com outras tecnologias *wireless*, não demonstrando sofrer interferência do *Wi-Fi* ou perdas de sinal devido a ambientes como muitas máquinas.

5 CONCLUSÃO

Neste trabalho foi realizado o estudo e a avalição do *bluetooth Low Energy* como alternativa para sistemas de localização em ambientes residenciais e industriais. Foram realizados experimentos em um ambiente residencial e no Laboratório de Eficiência Energética e Hidráulica em Saneamento (LENHS) localizado na UFPB.

Os resultados expostos no capítulo 4 chegaram a um valor de erro e desvio padrão de aproximadamente 3 dBm, para aproximações feitas a partir de uma curva polinomial e exponencial, sendo um valor menor do que o encontrado na literatura. É possível inferir a posição do localizador com uma variação de menos de um metro, o que o faz vantajoso quando comparado a outras tecnologias de posicionamento *indoor*. A variação do RSSI, que está em torno de 10 dBm, pode ser atenuada pela utilização do valor médio de 10 amostras e ainda enviar ao supervisório dados com tempo de atualização de um segundo.

Os valores da constante de propagação do sinal β encontrados divergem dos apresentados em outros trabalhos acadêmicos em torno de 0,3. Deve-se levar em conta, porém, as limitações de implementação, uma vez que a estimativa é feita com diversos localizadores funcionando simultaneamente, enquanto que nos experimentos deste trabalho foi utilizado apenas um. Além disso, fatores climáticos e ambientais podem resultar em variações no valor da constante β.

Não foi detectada perda de potência do sinal por interferência de máquinas adjacentes. Entretanto, a reflexão de sinal causada por objetos metálicos e máquinas operando em alta corrente causaram embaralhamento de dados no protocolo *bluetooth*. Isto pode ser um problema para locais em que haja um elevado número de dispositivos que utilizam variação de campo para realizar suas medições, como sensores de vazão eletromagnéticos e que seja preciso ficar próximo de tais equipamentos. Para casos em que o posicionamento a ser medido está distante ao menos um metro das máquinas a aplicação se torna eficaz.

O ScadaBR foi uma ferramenta útil no desenvolvimento deste trabalho, sendo capaz de monitorar os dados recebidos da placa com taxa de amostragem de até um segundo sem necessitar de um barramento serial. Em nenhum momento o supervisório emitiu alerta de perda de comunicação durante os experimentos. Isso demonstra a aplicabilidade da metodologia proposta em aplicações que envolvem um grande volume de dados (*Big Data*) e integração de diversos sensores e atuadores nos mais variados ramos industriais.

Como trabalhos posteriores se propõe a utilização de métodos de trilateração para estimar posições num determinado plano a partir de vários localizadores conectados a um mesmo microcontrolador. Dessa forma, será possível obter em pequena escala um sistema com precisão de centímetros e de baixo custo. Também o uso de outros métodos de cálculo, como o filtro de Kalman, podem ser aplicadas aos experimentos para análise da melhor forma de se obter a posição de cada localizador.

REFERÊNCIAS

AGHENTA, L. O.; IQBAL, M. T. Low-cost, open source iot-based scada system design using thinger. io and esp32 thing. **Electronics**, Multidisciplinary Digital Publishing Institute, v. 8, n. 8, p. 822, 2019.

AZEVEDO, C. L. d.; AZEVEDO, D. K. d. **Monitoramento de medidores de água via sistema supervisório**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2011.

CHOU, S.-J. et al. Application of bluetooth low energy indoor positioning to optical lenses manufacturing factory. In: IEEE. **2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)**. [S.l.], 2018. p. 1–6.

D'ALOIA, M. et al. Iot indoor localization with ai technique. In: IEEE. **2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT**. [S.l.], 2020. p. 654–658.

ELIZALDE, J. et al. Self-powered photovoltaic bluetooth® low energy temperature sensor node. **IEEE Access**, IEEE, v. 9, p. 111305–111314, 2021.

FARIA, M. H. M. d. et al. Estudo comparativo entre ferramentas de supervisão, controle e aquisição de dados e a importância destas para o ensino em engenharia. In: **Congresso Brasileiro de Educação em Engenharia (COBENGE). Belém–PA**. [S.l.: s.n.], 2012.

GIULIANO, R. et al. Indoor localization system based on bluetooth low energy for museum applications. **Electronics**, Multidisciplinary Digital Publishing Institute, v. 9, n. 6, p. 1055, 2020.

GOMES, J. P. d. T. Protótipo para monitoramento e aquisição de dados de pressão através de redes de sensores sem fio em sistemas de abastecimento de água com utilização da plataforma radiuino e scadabr. 2015.

KACZMAREK, M.; RUMINSKI, J.; BUJNOWSKI, A. Accuracy analysis of the rssi ble sensortag signal for indoor localization purposes. In: IEEE. **2016 Federated Conference on Computer Science and Information Systems (FedCSIS)**. [S.l.], 2016. p. 1413–1416.

LATHI, B. P.; DING, Z. Sistemas de comunicações analógicos e digitais modernos. **LTC, Rio de Janeiro**, 2012.

LI, G. et al. Indoor positioning algorithm based on the improved rssi distance model. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 18, n. 9, p. 2820, 2018.

LUGLI, A. B.; SANTOS, M. M. D. **Sistemas FIELDBUS para automação industrial: DeviceNet, CANopen, SDS e Ethernet**. [S.l.: s.n.], 2009.

MAIER, A.; SHARP, A.; VAGAPOV, Y. Comparative analysis and practical implementation of the esp32 microcontroller module for the internet of things. In: IEEE. **2017 Internet Technologies and Applications (ITA)**. [S.l.], 2017. p. 143–148.

MEKKI, K.; BAJIC, E.; MEYER, F. Indoor positioning system for iot device based on ble technology and mqtt protocol. In: IEEE. **2019 IEEE 5th World Forum on Internet of Things (WF-IoT)**. [S.l.], 2019. p. 787–792.

MODBUS, O. **MODBUS Messaging on TCP/IP Implementation Guide: V1. 0b**. [S.l.]: Modbus Organization, 2006.

MORAES, A. F. **Redes de Computadores: fundamentos, 5ª Edição**. [S.l.: s.n.], 2010.

MOUHAMMAD, C. S. et al. Ble indoor localization based on improved rssi and trilateration. In: IEEE. **2019 7th International Japan-Africa Conference on Electronics, Communications, and Computations,(JAC-ECC)**. [S.l.], 2019. p. 17–21.

NAIK, S. et al. Real time asset tracking using ble beacons. In: IEEE. **2019 Global Conference for Advancement in Technology (GCAT)**. [S.l.], 2019. p. 1–4.

PATERNA, V. C. et al. A bluetooth low energy indoor positioning system with channel diversity, weighted trilateration and kalman filtering. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 17, n. 12, p. 2927, 2017.

PEI, L. et al. Evaluation of fingerprinting-based wifi indoor localization coexisted with bluetooth. **The Journal of Global Positioning Systems**, Springer, v. 15, n. 1, p. 1–12, 2017.

POSTEL, J. et al. Transmission control protocol. STD 7, RFC 793, September, 1981.

QURESHI, U. M.; UMAIR, Z.; HANCKE, G. P. Indoor localization using wireless fidelity (wifi) and bluetooth low energy (ble) signals. In: IEEE. **2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)**. [S.l.], 2019. p. 2232–2237.

RAMIREZ, R. et al. A practice of ble rssi measurement for indoor positioning. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 21, n. 15, p. 5181, 2021.

RÖBESAAT, J. et al. An improved ble indoor localization with kalman-based fusion: An experimental study. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 17, n. 5, p. 951, 2017.

ROCHOL, J. **Sistemas de comunicação sem fio: conceitos e aplicações**. [S.l.]: Bookman Editora, 2018.

SALIMIBENI, M. et al. Iot-td: Iot dataset for multiple model ble-based indoor localization/tracking. In: IEEE. **2020 28th European Signal Processing Conference (EUSIPCO)**. [S.l.], 2021. p. 1697–1701.

SCADABR. **Manual do Software ScadaBR**. 2010. Disponível em: https://www.scadabr.com.br/index.php/2017/06/06/2939/.

SILVA, J. P. S. da; CUNHA, E. N. Supervisory system for hidraulic distribution apparatus scadabr application. In: IEEE. **2017 IEEE First Summer School on Smart Cities (S3C)**. [S.l.], 2017. p. 161–164.

SIQUEIRA, T. S. d. Bluetooth–características, protocolos e funcionamento. 2006. **Instituto de Computação. Universidade Estadual de Campinas**, 2006.

SURYAWANSHI, M. S. et al. Iot gateway design and implementation for modbus protocol. **International Journal of Engineering Applied Sciences and Technology**, 2019.

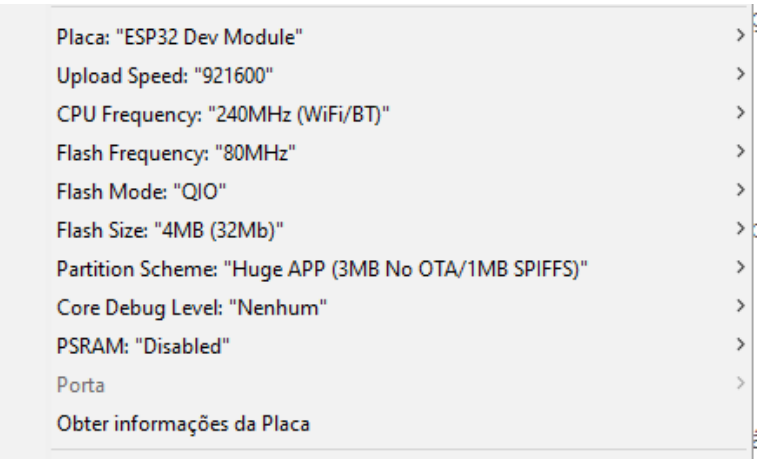
APÊNDICES

APÊNDICE A – CODIGO IMPLEMENTADO EM ARDUINO PARA COMUNICACAO BLE E WI-FI

Este apêndice apresenta o código implementado na IDE Arduino para comunicação do módulo ESP32 com o localizador BLE e o supervisório Scada-BR. A figura 28 apresenta uma modificação que dever ser feita antes de compilar o código. Por padrão a placa utilizada é a DOIT ESP32 DEVKIT V1, o que não permite acessar parâmetros internos como esquema de partição.

É preciso mudar a placa para ESP32 Dev Module e configurar a *Partition Scheme* para *Huge APP*, do contrário será apresentado um erro pois o código é maior que a memória disponibilizada.

Figura 28 – Configuração do Arduino para Pleno Funcionamento do Código



Fonte: Autor

Configuração BLE/Wi-Fi

```
1 #include <WiFi.h >
2 #include <BLEDevice.h>
3
4 //Definir nome e senha do WiFi que vai acessar
5 const char* ssid = "************"; // Nome da Rede Wi-Fi que
    vai conectar
6 const char* password = "************"; // Senha da rede Wi-Fi
7 int ModbusTCP_port = 502; // Porta que sera utilizada
8
9 #define ADDRESS "fc:58:fa:b5:09:c0" // Endereco do iTag,
    conseguido pelo proprio scan
10 #define SCAN_INTERVAL 100 // Intervalo entre cada scan
```

```
11 #define TARGET_RSSI -90 // Valor mais baixo de RSSI que pode
       ser adquirido.
12 #define MAX_MISSING_TIME 5000 // Tempo maximo de espera casa a
       tag nao seja encontrada
13
14 #define LED 2
15
16 BLEScan* pBLEScan; //Variavel que ira guardar o scan
17 uint32_t lastScanTime = 0; //Quando ocorreu o ultimo scan
18 boolean found = false; //Se encontrou o iTag no ultimo scan
19 uint32_t lastFoundTime = 0; //Tempo em que encontrou o iTag
       pela ultima vez
20
21 int rssi = 0; // Valor de potencia RSSI do sinal BLE. Varia de
       -90 a -27
22
23 //////// Requerido para Modbus TCP/IP /////////
24 #define maxInputRegister 10 // Numero maximo de registradores
       de input
25 #define maxHoldingRegister 10 // Numero maximo de registradores
        do tipo holding
26
27 // Ver apostila da WEG para mais informacoes
28 #define MB_FC_NONE 0
29 #define MB_FC_READ_REGISTERS 3 // Funcao de leitura de um unico
        registrador
30 #define MB_FC_WRITE_REGISTER 6 // Funcao de escrita de um unico
        registrador
31 #define MB_FC_WRITE_MULTIPLE_REGISTERS 16 // Funcao de escrita
       de multiplos registradores
32 //
33 // MODBUS Error Codes
34 //
35 #define MB_EC_NONE 0
36 #define MB_EC_ILLEGAL_FUNCTION 1 // O codigo da funcao eh
       invalido para escravos
37 #define MB_EC_ILLEGAL_DATA_ADDRESS 2 // O endereco eh invalido
       para escravos
38 #define MB_EC_ILLEGAL_DATA_VALUE 3 // Dados invalidos
```

```
39 #define MB_EC_SLAVE_DEVICE_FAILURE 4 // Um erro irrecuperavel
     ocorreu quando o servidor (ou escravo) tentava executar a
     acao solicitada
40 //
41 // MODBUS MBAP offsets
42 //
43 #define MB_TCP_TID 0 // Transaction Identifier (Identificador
     de transacao)
44 #define MB_TCP_PID 2 // Protocol Identifier (Identificador de
     protocolo)
45 #define MB_TCP_LEN 4 // Message Length (Tamanho da mensagem)
46 #define MB_TCP_UID 6 // Unity Identifier (Identificador de
     unidade)
47 #define MB_TCP_FUNC 7 // Function Code (Codigo da funcao)
48 #define MB_TCP_REGISTER_START 8 // The Data Address of the
     first register requested (Endereco do primeiro registrador
     solicitado)
49 #define MB_TCP_REGISTER_NUMBER 10 // The total number of
     registers requested (Numero total de registradores
     requisitados)
50
51 byte ByteArray[260];
52 unsigned int MBHoldingRegister[maxHoldingRegister];
53
54 //
     ////////////////////////////////////////////////////////////////


55
56 WiFiServer MBServer(ModbusTCP_port);
57
58 //Callback das chamadas ao scan
59 class MyAdvertisedDeviceCallbacks: public
     BLEAdvertisedDeviceCallbacks
60 {
61     void onResult(BLEAdvertisedDevice advertisedDevice)
62     {
63         //Se for o dispositivo que esperamos
64         if(advertisedDevice.getAddress().toString() == ADDRESS)
65         {
```

```
66              found = true;
67              advertisedDevice.getScan()->stop();
68              rssi = advertisedDevice.getRSSI();
69              Serial.print("RSSI: ");
70              Serial.println(rssi);
71
72          }
73      }
74 };
75
76 void setup() {
77  //Teste de coneccao
78  Serial.begin(115200);
79  delay(100) ;
80  WiFi.begin(ssid, password);
81  delay(100) ;
82  Serial.println(".");
83  while (WiFi.status() != WL_CONNECTED) {
84  delay(500);
85  Serial.print(".");
86  }
87  MBServer.begin();
88  Serial.println("Connected ");
89  Serial.print("ESP32 Slave Modbus TCP/IP ");
90  Serial.print(WiFi.localIP());
91  Serial.print(":");
92  Serial.println(String(ModbusTCP_port));
93  Serial.println("Modbus TCP/IP Online");
94
95 pinMode(LED, OUTPUT);
96
97 BLEDevice::init("");
98 pBLEScan = BLEDevice::getScan();
99 pBLEScan->setAdvertisedDeviceCallbacks(new
      MyAdvertisedDeviceCallbacks());
100 pBLEScan->setActiveScan(true);
101 }
102
103
```

```
104 void loop() {
105
106
107 // Check if a client has connected // Modbus TCP/IP
108 WiFiClient client = MBServer.available();
109 if (!client) {
110 return;
111 }
112
113
114 boolean flagClientConnected = 0;
115 byte byteFN = MB_FC_NONE;
116 int Start;
117 int WordDataLength;
118 int ByteDataLength;
119 int MessageLength;
120
121 // Modbus TCP/IP
122 while (client.connected()) {
123
124 if(client.available())
125 {
126 flagClientConnected = 1;
127 int i = 0;
128 while(client.available())
129 {
130 ByteArray[i] = client.read();
131 i++;
132 }
133
134 client.flush();
135
136
137 uint32_t now = millis(); //Tempo em milissegundos desde o boot
138
139    if(found){ //Se econtrou o iTag no ultimo scan
140        lastFoundTime = millis(); //Tempo em milissegundos de
              quando encontrou
141        found = false;
```

```
142
143     }
144
145     //Se nao encontrou e o tempo desde a ultima vez que
           econtrou for maior que o tempo maximo de desaparecido
146     else if(now - lastFoundTime > MAX_MISSING_TIME){
147         digitalWrite(LED, LOW);  //Desligamos o LED
148     }
149
150     if(now - lastScanTime > SCAN_INTERVAL){ //Se esta no tempo
           de fazer scan
151         //Marca quando ocorreu o ultimo scan e comeca o scan
152         lastScanTime = now;
153         pBLEScan->start(1);
154     }
155
156 if(rssi<-30&&rssi>-70){
157   if(now - lastFoundTime < MAX_MISSING_TIME){
158     digitalWrite(LED, HIGH);
159     }
160   }
161 else if(rssi>-30||rssi<-70){
162   digitalWrite(LED,LOW);
163   }
164
165  MBHoldingRegister[0] = rssi;
166  MBHoldingRegister[1] = random(0,12);
167  MBHoldingRegister[2] = random(0,12);
168  MBHoldingRegister[3] = random(0,12);
169  MBHoldingRegister[4] = random(0,12);
170  MBHoldingRegister[5] = random(0,12);
171  MBHoldingRegister[6] = random(0,12);
172  MBHoldingRegister[7] = random(0,12);
173  MBHoldingRegister[8] = random(0,12);
174  MBHoldingRegister[9] = random(0,12);
175
176
177  //// routine Modbus TCP
178  byteFN = ByteArray[MB_TCP_FUNC];
```

```
179 Start = word(ByteArray[MB_TCP_REGISTER_START],ByteArray[
      MB_TCP_REGISTER_START+1]);
180 WordDataLength = word(ByteArray[MB_TCP_REGISTER_NUMBER],
      ByteArray[MB_TCP_REGISTER_NUMBER+1]);
181 }
182
183 // Handle request
184
185 switch(byteFN) {
186 case MB_FC_NONE:
187 break;
188
189 case MB_FC_READ_REGISTERS: // 0x03 Ler todos os registradores
190 ByteDataLength = WordDataLength * 2;
191 ByteArray[5] = ByteDataLength + 3;
192 ByteArray[8] = ByteDataLength;
193 for(int i = 0; i < WordDataLength; i++)
194 {
195 ByteArray[ 9 + i * 2] = highByte(MBHoldingRegister[Start + i])
      ;
196 ByteArray[10 + i * 2] = lowByte(MBHoldingRegister[Start + i]);
197 }
198 MessageLength = ByteDataLength + 9;
199 client.write((const uint8_t *)ByteArray,MessageLength);
200
201 byteFN = MB_FC_NONE;
202
203 break;
204
205
206 case MB_FC_WRITE_REGISTER: // 0x06 Escrever em um unico
      registrador
207 MBHoldingRegister[Start] = word(ByteArray[
      MB_TCP_REGISTER_NUMBER],ByteArray[MB_TCP_REGISTER_NUMBER
      +1]);
208 ByteArray[5] = 6;
209 MessageLength = 12;
210 client.write((const uint8_t *)ByteArray,MessageLength);
211 byteFN = MB_FC_NONE;
```

```
212  break;
213
214  case MB_FC_WRITE_MULTIPLE_REGISTERS: //0x10 Escrever em
       multiplos registradores
215  ByteDataLength = WordDataLength * 2;
216  ByteArray[5] = ByteDataLength + 3;
217  for(int i = 0; i < WordDataLength; i++)
218  {
219  MBHoldingRegister[Start + i] = word(ByteArray[ 13 + i * 2],
       ByteArray[14 + i * 2]);
220  }
221  MessageLength = 12;
222  client.write((const uint8_t *)ByteArray,MessageLength);
223  byteFN = MB_FC_NONE;
224
225  break;
226  }
227  }
228
229
230 }
```