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

Mateus de Sousa

Desenvolvimento de Sistema de Coleta e Transmissão de Dados utilizando Drones e Sensores IoT

João Pessoa - PB
2024

Mateus de Sousa

Desenvolvimento de Sistema de Coleta e Transmissão de Dados utilizando Drones e Sensores IoT

Trabalho de Conclusão de Curso apresentado 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.

Universidade Federal da Paraíba

Centro de Energias Alternativas e Renováveis

Curso de Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Juan Moises Mauricio Villanueva

João Pessoa - PB

2024

Catalogação na publicação
Seção de Catalogação e Classificação

S725d Sousa, Mateus de.
Desenvolvimento de Sistema de Coleta e Transmissão de Dados utilizando Drones e Sensores IoT / Mateus de Sousa. - João Pessoa, 2024.
64 f. : il.

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

1. Sensores. 2. AES. 3. ESP32. 4. ESP-NOW. 5. Drone. 6. Edge Computing. 7. Computação de borda. 8. TEDA. I. Villanueva, Juan Moises Mauricio. II. Título.

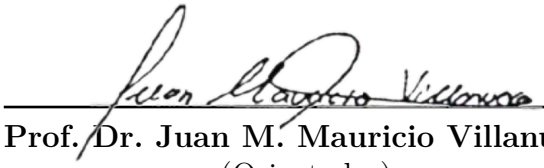
UFPB/BSCT CDU 621.3(043.2)

Elaborado por ROSANGELA GONCALVES PALMEIRA - CRB-216

Mateus de Sousa

Desenvolvimento de Sistema de Coleta e Transmissão de Dados utilizando Drones e Sensores IoT

Trabalho de Conclusão de Curso apresentado 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.



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

Prof. Dr. Alexsandro J. V. dos Santos
(Avaliador)
Universidade Federal da Paraíba

Prof. Dr. Cleonilson P. de Souza
(Avaliador)
Universidade Federal da Paraíba

João Pessoa - PB
2024

Dedico este trabalho a meus pais.

Agradecimentos

Gostaria de expressar minha profunda gratidão aos meus pais, Renato e Ana, pelo apoio e incentivo dados. À minha irmã, Luana. À minha namorada, Beatriz, pela compreensão e pelo carinho constantes, que foram essenciais durante todo o processo. E, claro, a meus quatro queridos companheiros de quatro patas, cuja alegria e presença tornam tudo mais leve. Agradeço em especial a todos os meus professores e, em especial, a meu orientador, Juan, pelo suporte e ensino ao longo deste trabalho. Aos colegas e amigos conquistados ao longo da universidade, que muito contribuíram para esta jornada, a todos meu mais sincero agradecimento.

"O sucesso não é definitivo, o fracasso não é fatal: o que conta é a coragem de continuar."
(Winston Churchill)

Resumo

A coleta de dados em áreas remotas e de acesso limitado apresenta desafios significativos para projetos de Internet das Coisas (IoT - Internet of Things). Este trabalho de conclusão de curso propõe uma solução inovadora utilizando drones e dispositivos ESP32 (Espressif Systems). O sistema compreende sensores ambientais baseados em ESP32, distribuídos no campo, e um drone equipado com ESP32 para coleta e processamento de dados. Os sensores de campo utilizam o framework ESP-IDF (Espressif IoT Development Framework) para otimizar o consumo energético, empregando modos de deep sleep e RTC (Real-Time Clock). Alimentados por energia solar, eles armazenam dados na memória NVS (Non-Volatile Storage), garantindo operação autônoma prolongada. O drone sobrevoa a área periodicamente, coletando dados via protocolo ESP-NOW (Espressif Wireless Communication Protocol) com criptografia AES (Advanced Encryption Standard). O sistema implementa edge computing no drone através do algoritmo TEDA (Typicality and Eccentricity Data Analytics), realizando processamento inicial dos dados. Posteriormente, os dados são enviados para a AWS IoT Core (Amazon Web Services Internet of Things Core) via MQTT (Message Queuing Telemetry Transport), armazenados no S3 (Simple Storage Service), processados pelo Athena (AWS Interactive Query Service) e visualizados no Grafana (Open-Source Analytics and Monitoring Platform). Esta abordagem integra eficientemente tecnologias IoT, drones e edge computing, oferecendo uma solução flexível e eficaz para monitoramento ambiental em áreas remotas, com foco em eficiência energética e processamento de dados em tempo real. O sistema foi testado em ambiente urbano, buscando uma linha de visada consistente com a localização do coletador de dados, já que irá utilizar energia solar. Foram coletados dados de autonomia, efetividade do TEDA e distância máxima para o protocolo utilizado durante a comunicação, posteriormente utilizando os dados para serem mostrados no ambiente Grafana funcionando uma uma máquina virtual da AWS.

Palavras-chave: Sensores, AES, ESP32, ESP-NOW, Drone, Edge Computing, Computação de borda, TEDA.

Abstract

Data collection in remote and hard-to-reach areas presents significant challenges for Internet of Things (IoT) projects. This thesis proposes an innovative solution using drones and ESP32 devices. The system consists of ESP32-based environmental sensors distributed in the field, and a drone equipped with ESP32 for data collection and processing. The field sensors use the ESP-IDF (Espressif IoT Development Framework) to optimize energy consumption, employing deep sleep modes and RTC (Real-Time Clock). Powered by solar energy, they store data in NVS (Non-Volatile Storage), ensuring prolonged autonomous operation. The drone periodically flies over the area, collecting data via the ESP-NOW (Espressif Wireless Communication Protocol) with AES (Advanced Encryption Standard) encryption. The system implements edge computing on the drone using the TEDA (Typicality and Eccentricity Data Analytics) algorithm, performing initial data processing. Subsequently, the data is sent to AWS IoT Core (Amazon Web Services Internet of Things Core) via MQTT (Message Queuing Telemetry Transport), stored in S3 (Simple Storage Service), processed by Athena (AWS Interactive Query Service), and visualized in Grafana (Open-Source Analytics and Monitoring Platform). This approach efficiently integrates IoT technologies, drones, and edge computing, providing a flexible and effective solution for environmental monitoring in remote areas, with a focus on energy efficiency and real-time data processing. The system was tested in an urban environment, seeking a consistent line of sightd with the location of the data collector, as it will use solar energy. Data on autonomy, effectiveness of the TEDA, and maximum distance for the protocol used during communication were collected, and later these data were used to be displayed in the Grafana environment running on an AWS virtual machine.

Keywords: Sensores, AES, ESP32, ESP-NOW, Drone, Edge Computing, Computação de borda, TEDA.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UAVs	*Unmanned Aerial Vehicles*
IoT	*Internet of Things*
TEDA	*Tipicity and Eccentricity Data Analysis*
MQTT	*Message Queuing Telemetry Transport*
AES	*Advanced Encryption Standard*
AWS	*Amazon Web Services*
RTC	*Real Time Clock*
BLE	*Bluetooth Low Energy*
SoC	*System on Chip*
Wi-Fi	*Wireless Fidelity*
LoS	*Line of Sight*
GPS	*Global Positioning System*
ADC	*Analog-to-Digital Converter*
HTTP	*Hypertext Transfer Protocol*
NTP	*Network Time Protocol*
ESP-NOW	*Espressif Proprietary Wireless Protocol*
JSON	*JavaScript Object Notation*
CPU	*Central Processing Unit*
LED	*Light Emitting Diode*
LDO	*Low Dropout Regulator*
GPIO	*General Purpose Input Output*
AI	*Artificial Intelligence*

Sumário

1 Introdução

Com o rápido avanço da tecnologia, a coleta de dados em áreas remotas se tornou uma prática fundamental em diversos setores, como agricultura de precisão, monitoramento ambiental e gestão de desastres naturais. A capacidade de obter dados precisos em locais de difícil acesso é essencial para tomadas de decisão eficazes e para o desenvolvimento de soluções que atendam às demandas desses setores. No entanto, a coleta de dados nessas regiões enfrenta desafios substanciais, incluindo a falta de infraestrutura adequada, a latência na transmissão de informações e a necessidade de garantir a segurança dos dados obtidos (ZHANG; LIU, 2020).

Atualmente, as soluções para a coleta de dados em áreas remotas frequentemente dependem de sensores distribuídos e redes de comunicação, como a Internet das Coisas (IoT), que facilita a troca de informações entre dispositivos de forma autônoma. A IoT tem se mostrado uma alternativa promissora ao permitir a automação de processos de coleta e análise de dados, superando limitações tradicionais de infraestrutura Mohammed e Smith (2021). Em locais de difícil acesso ou em áreas sem cobertura de rede, a implementação de sistemas autônomos de coleta de dados, combinada com drones (Veículos Aéreos Não Tripulados, UAVs) atuando como coletores móveis, pode ampliar significativamente a capacidade de monitoramento, eliminando a necessidade de conexão permanente com a rede (LIANG; YU; NGUYEN, 2020).

O uso de drones como coletores móveis de dados revoluciona a forma de obtenção de informações em áreas remotas. Nesta abordagem, dispositivos IoT autônomos são distribuídos estrategicamente pela área de interesse, operando continuamente na coleta e armazenamento local de dados. O drone, equipado com um sistema especializado de comunicação, sobrevoa periodicamente estes pontos de coleta, realizando a captação dos dados acumulados em cada dispositivo. Esta metodologia não apenas aumenta a área de cobertura, mas também otimiza o uso de recursos, já que cada dispositivo opera de forma independente até o momento da coleta Santos e Almeida (2022). Os dados coletados pelo drone podem ser processados localmente através de tecnologias de *edge computing*, permitindo análises preliminares antes da transferência final para a nuvem (ALMEIDA; SILVA, 2022).

Além disso, técnicas avançadas de análise de dados, como o uso de algoritmos para detecção de anomalias, como TEDA (*Typicality and Eccentricity Data Analytics*), podem ser utilizadas para filtrar ruídos e *outliers* nos dados antes de serem transmitidos para plataformas em nuvem como o AWS IoT Core. Isso garante que apenas os dados mais relevantes sejam transmitidos, economizando recursos de comunicação e energia, algo

fundamental em operações *off-grid* (GOULART; PINTO; BOAVA, 2022).

1.1 Objetivos

O objetivo central deste trabalho é desenvolver uma metodologia integrada para coleta de dados em áreas remotas utilizando dispositivos IoT autônomos distribuídos e drones como coletores móveis de dados. A proposta envolve dois componentes principais: dispositivos coletores fixos baseados em microcontroladores de baixo consumo, distribuídos estrategicamente para monitoramento contínuo, e um sistema embarcado no drone responsável pela coleta periódica dos dados armazenados nestes dispositivos. Esta abordagem visa criar uma solução robusta para cenários que exigem monitoramento remoto contínuo, mesmo em áreas sem infraestrutura de comunicação permanente.

O sistema é projetado de forma que cada dispositivo coletor opere de maneira autônoma, realizando medições constantes e armazenando dados históricos localmente. O drone, equipado com um microcontrolador especializado, executa voos programados sobre os pontos de coleta, estabelecendo comunicação sem fio com cada dispositivo para transferência eficiente dos dados acumulados. Após completar sua rota de coleta, o drone retorna à base para descarregar os dados obtidos, que são então processados e transmitidos de forma segura para a nuvem.

1.1.1 Objetivos específicos

Os objetivos específicos envolvem os seguintes itens:

- Desenvolvimento de um sistema de mapeamento eficiente: Implementação de um algoritmo de mapeamento que maximize a cobertura da área alvo e otimize a coleta de dados. O sistema deve minimizar o tempo de operação do drone e o consumo de energia, garantindo que a missão seja concluída com o menor uso de recursos possível. Isso inclui a definição de rotas inteligentes que considerem a autonomia da bateria e as áreas de maior prioridade para coleta.

- Integração de *edge computing* no drone: Durante o deslocamento do drone, o uso de edge computing será empregado para processar os dados coletados em tempo real. Esse processamento local permitirá a filtragem inicial dos dados, como a remoção de ruídos ou outliers, e a priorização das informações mais relevantes. Ao pré-processar os dados ainda no drone, será possível reduzir o volume de informações a serem transmitidas para a nuvem, economizando largura de banda e recursos de comunicação.

- Garantia de transmissão segura e eficiente dos dados: A implementação de mecanismos de segurança robustos garantirá que os dados sejam transmitidos de maneira segura,

utilizando protocolos de criptografia para proteger as informações sensíveis durante o envio. Além disso, o sistema será configurado para assegurar que os dados cheguem à nuvem sem perda de qualidade ou integridade, essencial para garantir a precisão das análises subsequentes. A segurança será combinada com a eficiência no envio, priorizando uma comunicação rápida e otimizada, que respeite as limitações do ambiente de transmissão.

1.2 Organização do trabalho

Dentro deste estudo, para além deste primeiro capítulo introdutório, estão dispostos mais quatro capítulos que serão apresentados a seguir.

No segundo capítulo, está reservado o espaço para a exposição da base teórica a respeito do tópico abordado. Neste capítulo, aprofundam-se os conceitos envolvendo a *arquitetura e protocolos de comunicação em IoT*, ao mesmo tempo em que se realiza uma análise mais detalhada sobre a segurança na tranmissão de dados e a consideração no uso de drones e UAVs na coleta de dados.

O terceiro capítulo engloba a exposição do estudo de caso, onde serão fornecidas as informações dos locais de teste e dos sensores utilizados na captura dos dados. Também sendo analisado quais os protocolos de comunicação serão selecionados em todo o sistema, de forma a obter o melhor resultado, sendo um passo crucial na obtenção de dados de forma rápida e eficiente. Assim como, são apresentados os códigos utilizados nos microcontroladores e o porquê foram implementados da forma que são apresentados.

O quarto capítulo é destinado à exposição da implementação prática e dos resultados obtidos no projeto de integração entre drones, IoT e edge computing. Neste capítulo, são descritos os processos de captura de dados pelos drones, desde a configuração dos sensores e dispositivos IoT até o mapeamento das áreas-alvo.

É detalhado como os dados foram processados em tempo real durante o tempo de voo do drone, abordando os desafios enfrentados e as soluções implementadas. Além disso, a transmissão dos dados para a nuvem é explicada, assim como os procedimentos de armazenamento e organização das informações coletadas.

Por fim, são discutidos os resultados obtidos, destacando as vantagens e limitações das abordagens adotadas, bem como as incertezas associadas aos modelos preditivos. Sugestões para melhorias futuras e possíveis aplicações adicionais da tecnologia também são incluídas.

2 Revisão Bibliográfica

Esta revisão bibliográfica aborda os principais aspectos tecnológicos e desafios relacionados à integração de drones, Internet das Coisas (IoT) e sistemas embarcados para coleta de dados em áreas remotas. Serão discutidos temas como: a evolução dos VANTs na coleta de dados, as aplicações da IoT em sistemas autônomos, a importância do edge computing no processamento de dados, os protocolos de segurança em transmissões wireless e as tendências futuras neste campo de estudo.

2.1 Drones e VANTs na Coleta de Dados

Os Veículos Aéreos Não Tripulados (VANTs), popularmente conhecidos como drones, revolucionaram a coleta de dados em áreas de difícil acesso. Segundo Makam et al. (2024), estes dispositivos têm se destacado especialmente na agricultura inteligente, onde permitem o monitoramento preciso de grandes áreas cultivadas. A evolução dos sistemas autônomos de navegação, conforme apresentado por Rodriguez e Smith (2023), possibilitou o desenvolvimento de algoritmos otimizados para planejamento de rotas, considerando fatores críticos como consumo de energia e cobertura eficiente da área alvo.

Em aplicações ambientais, Manfreda et al. (2018) demonstra como os drones equipados com sensores IoT podem realizar monitoramento contínuo sem intervenção humana, coletando dados cruciais sobre condições ambientais em áreas remotas. Esta capacidade tem se mostrado particularmente valiosa em cenários urbanos, onde Mohamed et al. (2020) destaca aplicações inovadoras na gestão de cidades inteligentes e monitoramento de infraestrutura.

2.2 Internet das Coisas (IoT) e Sistemas Embarcados

A Internet das Coisas (IoT) revolucionou a forma como dispositivos coletam e compartilham dados. Kim e Johnson (2024) apresenta avanços significativos na otimização do edge computing para aplicações IoT, destacando como o processamento local pode reduzir significativamente a latência e o consumo de banda.

No contexto de sistemas embarcados para aplicações IoT, é crucial avaliar diferentes plataformas para determinar a mais adequada. Atualmente, várias opções estão disponíveis no mercado, cada uma com suas vantagens e limitações. A Tabela 1 apresenta uma comparação entre algumas das plataformas mais populares.

A escolha do sistema embarcado adequado é determinante para a viabilidade de

projetos IoT, especialmente quando se trata de aplicações que exigem equilíbrio entre consumo energético, conectividade e capacidade de processamento local. Entre as várias opções disponíveis no mercado, cada uma apresenta características específicas que podem atender diferentes requisitos de projetos IoT. A seguir, são apresentadas as principais vantagens da ESP32 e uma comparação detalhada com outras plataformas.

Tabela 1 – Comparação de Sistemas Embarcados para Aplicações em IoT

Característica	**ESP32[1]**	**Raspberry Pi Zero W[2]**	**Arduino Nano 33 IoT[3]**	**STM32 F411RE[4]**
Frequência do CPU	240 MHz	1 GHz	48 MHz	100 MHz
RAM	520 KB	512 MB	32 KB	128 KB
Conectividade	Wi-Fi + BLE + ESP-NOW	Wi-Fi + BLE + P2P (Ad Hoc)	Wi-Fi + BLE	Necessita módulo externo
Consumo de energia (ativo)*	80 mA	230 mA	9 mA	10 mA
Preço aproximado (USD)	$3-10	$10-15	$15-20	$10-15
Preço aproximado (BRL)	R$15-50	R$50-75	R$75-100	R$50-75
Peso	7g	9g	5g	8g
Flash Memory (interno)	4 MB	MicroSD (dependente)	256 KB	512 KB
GPIOs disponíveis	34	40	14	50
Protocolos suportados	MQTT, HTTP, CoAP, ESP-NOW	MQTT, HTTP, P2P (Ad Hoc)	MQTT, HTTP, etc.	MQTT, HTTP, etc.

* Com módulos de comunicação desligados
[1] Fonte: Espressif Systems.
[2] Fonte: Raspberry Pi Foundation.
[3] Fonte: Arduino.
[4] Fonte: STMicroelectronics.

2.2.1 Análise da ESP32

A ESP32 apresenta características relevantes para projetos de IoT, embora também possua limitações específicas que devem ser consideradas conforme o contexto de aplicação. A seguir, são apresentadas suas principais características:

- **Desempenho e energia**: Com CPU de 240 MHz e consumo de 80 mA, a ESP32 oferece um equilíbrio entre processamento e eficiência energética, adequado para determinadas aplicações autônomas, embora existam opções mais eficientes em energia como o Arduino Nano IoT.
- **Conectividade**: A integração de Wi-Fi e Bluetooth permite conexões com redes e dispositivos IoT. O protocolo ESP-NOW possibilita comunicação entre dispositivos

sem redes Wi-Fi, embora com limitações de alcance e quantidade de dispositivos conectados.

- **Processamento e segurança**: Oferece recursos de criptografia em hardware e boot seguro, características importantes para transmissões de dados, ainda que com algumas limitações de processamento comparado a sistemas mais robustos como Raspberry Pi.
- **Custo**: Com preço entre R$15 e R$50, representa uma opção acessível, porém existem alternativas mais econômicas no mercado dependendo dos requisitos do projeto.
- **Suporte técnico**: Conta com documentação e comunidade ativa, facilitando o desenvolvimento, embora parte do material esteja disponível apenas em inglês e exista dependência das bibliotecas do fabricante.

2.2.2 Características e Limitações da ESP32

A ESP32, desenvolvida pela Espressif Systems, apresenta um conjunto de características que devem ser analisadas considerando os requisitos específicos de cada projeto:

2.2.2.1 Pontos Positivos

- **Integração**: Combina CPU, memória, Wi-Fi, Bluetooth e periféricos em um único chip, contribuindo para redução de tamanho e custo em determinadas aplicações.
- **Gerenciamento de Energia**: Disponibiliza diferentes modos de economia de energia, incluindo deep sleep, permitindo operação com fontes de energia limitadas.
- **Capacidade de Processamento Local**: O processador dual-core de 240 MHz permite execução de algoritmos básicos de edge computing para processamento local de dados.
- **Conectividade**: Integra Wi-Fi e Bluetooth LE (BLE), além do protocolo ESP-NOW para comunicações ponto a ponto em certas condições.
- **Recursos de Segurança**: Inclui funcionalidades básicas de criptografia em hardware e suporte a secure boot.
- **Armazenamento**: Compatível com sistemas de arquivos como LittleFS, permitindo organização de dados na memória flash.

2.2.2.2 Limitações e Desafios

- **Memória Limitada**: Com 520 KB de RAM, apresenta limitações significativas comparada a sistemas como Raspberry Pi, restringindo aplicações mais complexas.

- **Complexidade de Otimização**: A utilização eficiente dos modos de economia de energia requer conhecimento específico e pode aumentar a complexidade do desenvolvimento.
- **Processamento**: Limitações para aplicações que exigem alto poder computacional, como processamento de imagem em tempo real ou análises complexas de dados.
- **Dependência de Bibliotecas**: Necessidade de utilizar bibliotecas específicas do fabricante, podendo limitar a portabilidade do código.
- **Conectividade**: O protocolo ESP-NOW, embora útil, tem limitações de alcance e número de dispositivos conectados simultaneamente.
- **Desenvolvimento**: A necessidade de conhecimentos específicos da plataforma pode aumentar o tempo de desenvolvimento inicial.

2.2.3 Comparativo com Outras Plataformas

Cada plataforma possui características que a tornam mais adequada para determinados tipos de projeto:

- **ESP32**: Adequada para projetos que necessitam de conectividade wireless e baixo consumo energético, com processamento local moderado.
- **Raspberry Pi Zero W**: Ideal para aplicações que demandam maior poder de processamento e execução de sistemas operacionais completos.
- **Arduino Nano 33 IoT**: Mais apropriado para projetos que priorizam extrema eficiência energética e simplicidade de programação.
- **STM32 F411RE**: Recomendado para aplicações que necessitam de maior quantidade de GPIOs e precisam de processamento dedicado.

A escolha da plataforma deve considerar cuidadosamente os requisitos específicos do projeto, incluindo necessidades de processamento, consumo de energia, conectividade e orçamento disponível. Cada sistema possui seus pontos fortes e limitações, não existindo uma solução única ideal para todos os cenários.

2.3 Computação de borda e Processamento de Dados

O paradigma de computação de borda transformou o processamento de dados em sistemas IoT. Patel e Anderson (2023) demonstra como o processamento em tempo real em sistemas utilizados nos veículos aéreos não tripulados pode melhorar significativamente

a eficiência operacional. Esta abordagem é particularmente relevante em cenários onde a largura de banda é limitada ou o custo de transmissão é alto.

A implementação de algoritmos de processamento local, como destacado por Kim e Johnson (2024), permite:

- Redução significativa no volume de dados transmitidos
- Menor latência no processamento de informações críticas
- Maior autonomia operacional do sistema

2.4 Segurança e Transmissão de Dados

A segurança na transmissão de dados é um aspecto crítico em sistemas IoT baseados em drones. Sharma e Mehra (2023) apresenta protocolos de segurança específicos para redes UAV-IoT, enquanto Aljumah, Ahanger e Ullah (2023) propõe uma abordagem inovadora utilizando blockchain para garantir a integridade dos dados coletados.

Os principais aspectos de segurança considerados incluem:

- Criptografia fim-a-fim dos dados transmitidos
- Autenticação robusta dos dispositivos
- Proteção contra interceptação e manipulação de dados
- Garantia de integridade das informações coletadas

2.5 Desafios e Tendências Futuras

Os sistemas de coleta de dados baseados em drones enfrentam diversos desafios, como identificado por Makam et al. (2024) e Rodriguez e Smith (2023):

- Otimização da autonomia das baterias
- Confiabilidade da comunicação em áreas remotas
- Processamento eficiente de grandes volumes de dados
- Segurança e privacidade dos dados coletados

Tendências emergentes, conforme Mohamed et al. (2020), incluem:

- Integração com sistemas de inteligência artificial

- Desenvolvimento de drones com maior autonomia energética
- Implementação de redes mesh para comunicação entre múltiplos drones
- Uso de técnicas avançadas de processamento distribuído

A revisão bibliográfica apresentada expõe os principais desafios e oportunidades na integração de drones, IoT e sistemas embarcados para coleta de dados em áreas remotas. Entre os desafios destacados estão a otimização do consumo de energia, a confiabilidade das comunicações sem fio em ambientes de difícil acesso, e a segurança na transmissão de dados sensíveis. Em contrapartida, as oportunidades oferecidas pela evolução dos sistemas autônomos, o avanço das plataformas IoT, e a implementação de algoritmos de edge computing abrem caminho para soluções inovadoras e mais eficientes. As tendências futuras, como a integração com inteligência artificial e o uso de redes mesh para comunicação entre drones, reforçam a necessidade de pesquisa e desenvolvimento contínuos neste campo. Esses fatores, combinados com as lacunas identificadas na literatura, motivaram o presente trabalho, que busca explorar a ESP32 como uma das plataformas possíveis para coletar, processar e transmitir dados em aplicações de IoT, considerando aspectos como autonomia dos dispositivos e confiabilidade das informações.

3 Fundamentação Teórica

Neste capítulo, são apresentados os conceitos-chave que servirão de base para o desenvolvimento deste trabalho. Inicialmente, é abordado a utilização do protocolo ESP-Now, em seguida, discutido utilização de criptografia para envio de dados. Na sequência, a utilização de computação em borda e, mais especificamente, o conceito do TEDA (*Tipicity and Eccentricity Data Analytics*) e sua aplicação na detecção de *outliers*.

3.1 Protocolo ESP-NOW

O ESP-NOW é um protocolo de comunicação proprietário desenvolvido pela Espressif Systems, voltado para os microcontroladores ESP8266 e ESP32. Diferente dos protocolos tradicionais, como Wi-Fi ou Bluetooth, o ESP-NOW não depende de uma infraestrutura de rede, como roteadores ou pontos de acesso, o que o torna adequado para a criação de redes ponto a ponto (*peer-to-peer*, P2P) ou redes diretas em ambientes distribuídos. Essa abordagem é particularmente vantajosa em cenários onde a conectividade com a internet é limitada ou inexistente, como em áreas remotas ou fora de alcance de infraestrutura de rede (PASIC; KUZMANOV; ATANASOVSKI, 2021).

Esse protocolo de baixa latência e baixo consumo energético é amplamente utilizado em aplicações de Internet das Coisas (IoT), onde a eficiência no uso de energia e a simplicidade de implementação são fatores críticos para o sucesso. O ESP-NOW utiliza uma forma simplificada de comunicação sem fio, operando nas camadas física (PHY) e de enlace (MAC - Media Access Control) do modelo OSI(*Open Systems Interconnection*) do padrão IEEE 802.11, mas sem a complexidade associada à pilha de protocolos Wi-Fi (MAGZYM et al., 2023).

3.2 Características Principais do ESP-NOW

O ESP-NOW apresenta diversas vantagens em relação a outros protocolos, sendo especialmente útil em redes distribuídas e sem infraestrutura. Suas características principais incluem:

- **Comunicação Direta**: O ESP-NOW permite que os dispositivos ESP32 se comuniquem diretamente entre si, eliminando a necessidade de um ponto de acesso ou roteador. Isso reduz significativamente a latência e simplifica a comunicação em ambientes sem infraestrutura centralizada (PASIC; KUZMANOV; ATANASOVSKI, 2021).

- **Baixa Latência**: O protocolo é otimizado para fornecer comunicação rápida, com latência muito menor que a de protocolos como Wi-Fi ou Bluetooth, tornando-o adequado para aplicações em tempo real, como controle de dispositivos e monitoramento de sistemas críticos (MAGZYM et al., 2023).

- **Baixo Consumo Energético**: O ESP-NOW foi projetado para ser extremamente eficiente em termos de consumo de energia, o que o torna adequado para dispositivos que operam com baterias por longos períodos. Em combinação com modos de baixa energia, como *deep sleep*(modo de economia profunda de energia), a duração operacional dos dispositivos pode ser prolongada significativamente, o que é essencial para aplicações IoT em locais remotos (KOKAN, 2021).

- **Flexibilidade na Comunicação**: O ESP-NOW suporta tanto comunicação *unicast* (um para um) quanto *broadcast* (um para muitos), permitindo que um dispositivo mestre se comunique com vários dispositivos ao mesmo tempo. Isso é especialmente útil em redes IoT que envolvem múltiplos sensores espalhados por uma área de cobertura extensa (MAGZYM et al., 2023).

- **Suporte a Segurança**: O protocolo inclui suporte para criptografia com AES-128, protegendo a transmissão de dados entre dispositivos e garantindo a integridade e confidencialidade das informações trocadas, especialmente importante em aplicações sensíveis e em locais remotos sem infraestrutura confiável (PASIC; KUZMANOV; ATANASOVSKI, 2021).

3.3 Comparação com Outros Protocolos de Comunicação

Comparar o ESP-NOW com outros protocolos de comunicação comuns é crucial para entender suas vantagens em cenários específicos.

3.3.1 ESP-NOW vs. Wi-Fi

Comparando esses protocolos, que possuem bastante similaridade, já que o ESP-NOW deriva do protocolo de Wi-Fi com 2.4GHz, porém com mais simplificações.

- **Consumo de Energia**: O Wi-Fi, embora forneça alta taxa de transferência de dados, consome significativamente mais energia em comparação com o ESP-NOW. Isso o torna menos adequado para dispositivos IoT alimentados por bateria que operam em locais remotos.

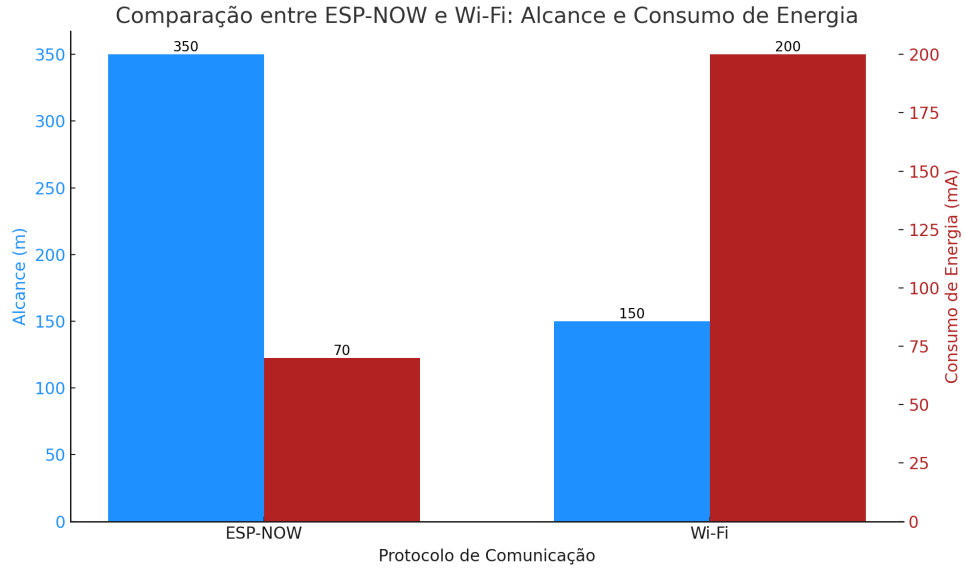
- **Complexidade de Rede**: Enquanto o Wi-Fi requer uma infraestrutura de rede (roteadores e pontos de acesso), o ESP-NOW é completamente descentralizado,

permitindo que os dispositivos operem de forma independente e sem a necessidade de configuração de rede complexa.

- **Latência**: O ESP-NOW oferece uma latência muito menor, sendo adequado para aplicações que requerem uma comunicação rápida e em tempo real, como monitoramento ambiental e controle de dispositivos (MAGZYM et al., 2023).

Como pode ser observado na figura 1:

Figura 1 – Comparativo de alcance e consumo médio entre ESP-NOW e WiFi



Fonte: Autoria Própria.

3.3.2 ESP-NOW vs. *Bluetooth/BLE (Bluetooth Low Energy)*

Comparando esses protocolos, que possuem foco na eficiência energética e menor densidade de informações transmitidas em decorrência dos protocolos.

- **Alcance**: O ESP-NOW pode alcançar distâncias maiores do que o *Bluetooth Low Energy* (BLE), especialmente em áreas abertas, com uma comunicação estável em até 350 metros, tornando-o mais eficiente em ambientes externos e distribuídos.

- **Número de Conexões**: Enquanto o BLE tem limitações no número de conexões simultâneas, o ESP-NOW pode gerenciar múltiplas conexões de forma eficiente, o que o torna adequado para redes com vários sensores ou dispositivos atuando simultaneamente.

- **Consumo Energético**: Apesar do consumo menor do BLE, ambos protocolos são eficientes em termos de consumo de energia, mas o ESP-NOW oferece maior

controle sobre o gerenciamento de energia, permitindo um consumo ajustável em implementações onde o modo de *deep sleep* pode ser utilizado (PASIC; KUZMANOV; ATANASOVSKI, 2021).

  - Para transmissão contínua: BLE é geralmente mais eficiente.
  - Para transmissões rápidas e esporádicas: ESP-NOW pode ser competitivo.

Segundo Alzahrani et al. (2021), o protocolo ESP-NOW demonstrou excelente suporte para longas distâncias em condições de Linha de Visada (LoS), com baixa latência. No entanto, consome mais energia e o sinal é rapidamente enfraquecido por obstruções. Wi-Fi apresentou a melhor velocidade de transmissão, enquanto o Bluetooth teve o menor consumo de energia, sendo mais adequado para dispositivos que precisam economizar energia em curtas distâncias.

3.3.3 ESP-NOW vs. Zigbee/Z-Wave

Comparando esses protocolos, que possuem similaridades na utilização de uma infraestrutura coordenada entre dispositivos diferentes conectadas à mesma rede.

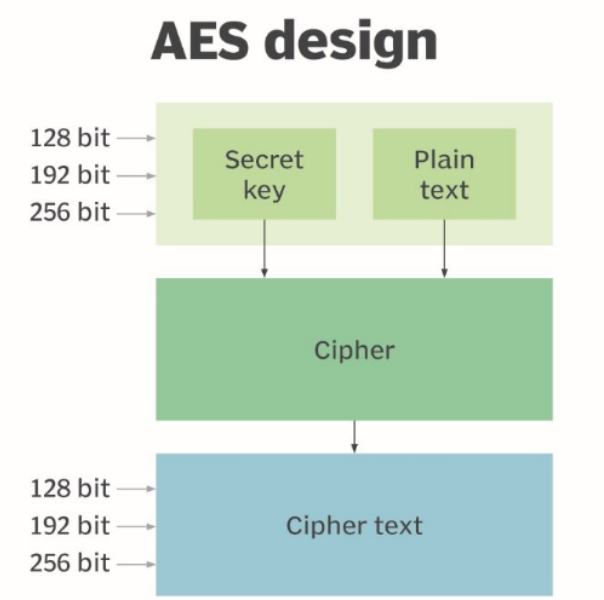
- **Infraestrutura**: Zigbee e Z-Wave são projetados para redes em malha mais complexas e requerem hardware especializado para comunicação, enquanto o ESP-NOW pode ser implementado em dispositivos ESP32, que são amplamente disponíveis e de baixo custo.(SANTOS et al., 2023)
- **Complexidade**: As pilhas de protocolo Zigbee e Z-Wave são mais complexas, o que pode aumentar a latência e o consumo de energia. O ESP-NOW, por outro lado, oferece uma solução mais simples e de menor consumo para redes de menor escala e com menos exigências de infraestrutura.(SANTOS et al., 2023)
- **Escalabilidade**: Embora Zigbee e Z-Wave ofereçam alta escalabilidade para redes maiores, o ESP-NOW se destaca em redes menores ou médias, onde a simplicidade e a rapidez são prioritárias (KOKAN, 2021).

3.4 Criptografia

A segurança de dados é um dos maiores desafios no contexto de redes de comunicação e, mais especificamente, na Internet das Coisas (IoT), onde dispositivos trocam dados sensíveis frequentemente. Como as comunicações em IoT podem ser interceptadas por agentes mal-intencionados, torna-se fundamental adotar medidas de segurança robustas para garantir a confidencialidade e integridade dos dados Stallings (2006). A criptografia

AES (*Advanced Encryption Standard*) de 16 bytes, observado o design na Figura 2, combinada com o preenchimento PKCS#7, é uma solução amplamente aceita que proporciona uma camada de proteção essencial para os dados enviados e recebidos entre dispositivos. Este trabalho explora o uso dessa criptografia para garantir a proteção da comunicação em um ambiente suscetível à interceptação.

Figura 2 – Esquema de funcionamento da criptografia AES



Fonte: ROUSE, Margaret. Advanced Encryption Standard (AES). TechTarget, 2021. Disponível em: https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard.

3.4.1 Criptografia AES de 16 Bits

3.4.1.1 Definição e Funcionamento do AES

O *AES (Advanced Encryption Standard)* é um algoritmo de criptografia simétrica padronizado pelo Instituto Nacional de Padrões e Tecnologia (NIST) dos Estados Unidos Daemen e Rijmen (2002). O AES opera em blocos de dados de tamanho fixo de 128 bits (16 bytes) e utiliza chaves de 128, 192 ou 256 bits, dependendo do nível de segurança necessário. Para o projeto em questão, foi utilizado o AES com blocos de 16 bytes e chave de 128 bits devido à sua eficiência e segurança.

3.4.1.2 Vantagens do AES de 16 Bits

- **Segurança**: O AES é reconhecido por sua robustez contra ataques de força bruta e criptoanálises, garantindo proteção mesmo em ambientes vulneráveis, como a IoT (KATZ; LINDELL, 2010).

- **Eficiência**: Em dispositivos embarcados com recursos limitados, como processadores de baixa potência, o AES com blocos de 16 bytes apresenta um ótimo balanço entre segurança e desempenho, permitindo criptografia em tempo real.
- **Aceitação Global**: O AES é amplamente utilizado e aceito, garantindo interoperabilidade entre dispositivos e sistemas (STALLINGS, 2006).

3.4.2 Preenchimento PKCS#7

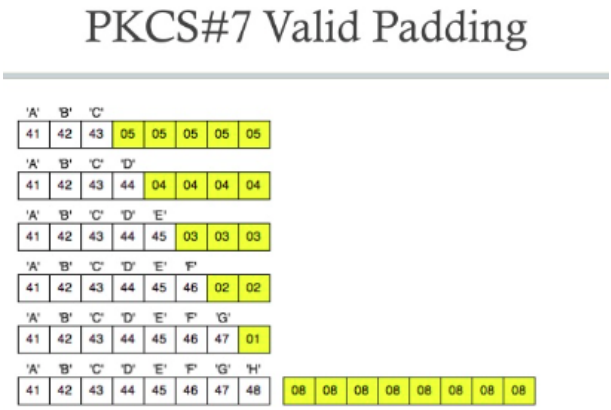
3.4.2.1 A Necessidade de Preenchimento (*Padding*)

O AES é um algoritmo de blocos que requer que os dados tenham tamanhos múltiplos de 16 bytes para funcionar corretamente. Quando o tamanho dos dados não é um múltiplo de 16, é necessário adicionar bytes extras para completar o bloco, processo conhecido como *padding* (KATZ; LINDELL, 2010).

3.4.2.2 O Que é o PKCS#7?

O PKCS#7 (Public-Key Cryptography Standards #7) é um dos métodos de padding mais utilizados. Ele adiciona bytes de preenchimento ao final do último bloco, como visto na Figura 3, onde o valor de cada byte indica a quantidade de bytes adicionados. Por exemplo, se faltam quatro bytes para completar um bloco, o padding será ‘04 04 04 04‘. Isso garante que o tamanho dos dados seja múltiplo de 16 (SONG; ZHANG, 2012).

Figura 3 – Exemplo de preenchimento PKCS#7



Fonte: Stack Overflow. What is the difference between PKCS5 padding and PKCS7 padding, 2019. Disponível em: https://stackoverflow.com/questions/20770072/what-is-the-difference-between-pkcs5-padding-and-pkcs7-padding.

3.4.2.3 Vantagens do PKCS#7

- **Simplicidade**: O PKCS#7 é simples de implementar e remover após a decriptação, sendo amplamente suportado por bibliotecas criptográficas (KATZ; LINDELL, 2010).

- **Segurança**: A simplicidade do PKCS#7 permite que o preenchimento seja facilmente identificado e removido, sem comprometer a segurança dos dados transmitidos.

3.4.3 Criptografia e Decriptação no Projeto

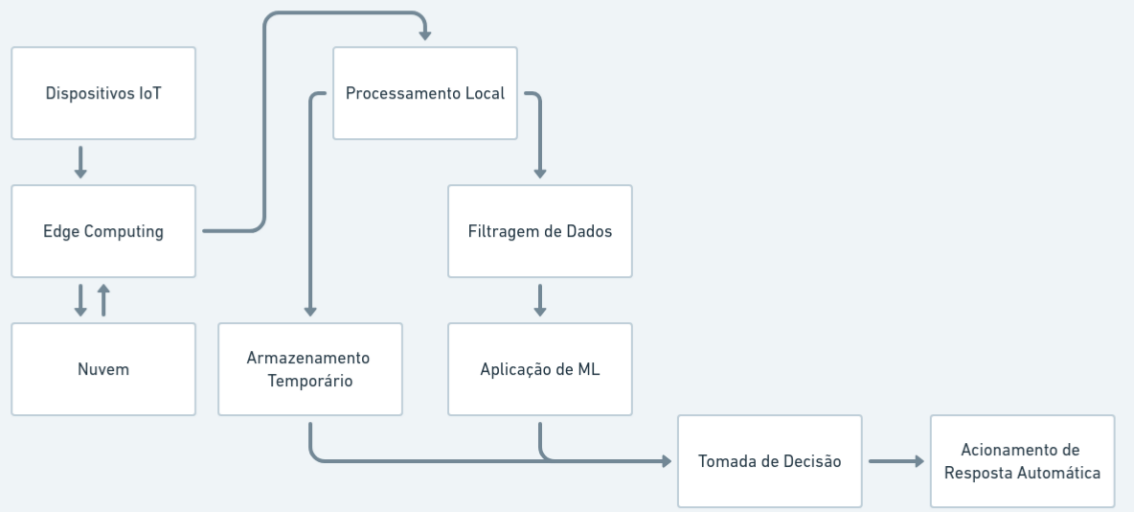
No presente projeto, o AES de 16 bits é utilizado para criptografar os dados antes que sejam enviados, garantindo que a comunicação entre dispositivos seja segura. Após a transmissão, os dados são decriptados utilizando a mesma chave simétrica, garantindo que somente dispositivos autorizados possam acessar as informações. O padding PKCS#7 assegura que blocos de dados de tamanhos variados possam ser adequadamente criptografados, sem falhas de operação. Dessa forma, o sistema consegue proteger a integridade e a confidencialidade das informações trafegadas, mesmo em ambientes vulneráveis à interceptação, como redes IoT.

3.5 Computação de borda

Com o crescimento exponencial de dispositivos conectados na Internet das Coisas (IoT), surgem desafios relacionados à latência, consumo de banda e segurança na transmissão de grandes volumes de dados para a nuvem. A computação de borda (ou edge computing) se apresenta como uma solução eficaz, ao processar dados localmente, nos dispositivos de borda, próximos à origem dos dados. Em vez de enviar todas as informações para a nuvem, a computação de borda permite que as decisões sejam tomadas localmente, reduzindo a latência e a dependência de redes externas, melhorando assim a eficiência e a confiabilidade dos sistemas IoT Azion (2021), sendo a infraestrutura exemplificada na figura 4.

Além disso, a computação de borda reduz a sobrecarga de transmissão de dados e aumenta a privacidade ao tratar informações sensíveis diretamente no dispositivo, sem a necessidade de transferi-las para servidores externos Hat (2022). Dispositivos de borda, que são os dispositivos intermediários do processo, podem realizar tarefas computacionais mais complexas com menor consumo de energia e recursos, tornando-se viáveis para aplicações em tempo real, como monitoramento e controle em sistemas distribuídos (IBERDROLA, 2021). A utilização de algoritmos eficientes é essencial para maximizar os benefícios da computação de borda. Na próxima seção, será apresentado o TEDA (*Typicality and Eccentricity Data Analytics*), um algoritmo robusto e leve que é adequado para detecção de anomalias em tempo real em dispositivos IoT com recursos limitados.

Figura 4 – Exemplificação do que significa a computação em borda.



Fonte: Autoria Própria.

3.6 Algoritmo TEDA

O algoritmo conhecido como TEDA (*Typicality and Eccentricity Data Analytics*) foi desenvolvido com base nos princípios de tipicidade e excentricidade, conforme discutido por Angelov (2014). Este método é projetado para detectar *outliers* em conjuntos de dados. A tipicidade refere-se à similaridade de uma amostra em relação às demais do conjunto, enquanto a excentricidade avalia o quão excepcional um dado é em comparação com os outros (SMITH, 2021).

O TEDA se destaca no campo de aprendizado de máquina devido à sua abordagem inovadora, que elimina a necessidade de:

- Suposições prévias sobre a distribuição dos dados, uma prática comum em muitos métodos de detecção de *outliers*;
- Definição antecipada de parâmetros específicos do problema, o que pode ser desafiador em diversas situações;
- Pressuposição de independência entre as amostras de dados, uma limitação presente em muitos algoritmos tradicionais;
- Necessidade de um número finito de observações; o TEDA demonstra eficácia com apenas três amostras, o que é notavelmente eficiente.

Dentro do contexto do TEDA, três conceitos fundamentais emergem: tipicidade (Γ), excentricidade (ξ) e proximidade acumulada (π). Esses conceitos formam a base para a análise de dados no TEDA.

Considerando um espaço de dados n-dimensional, representado como $X \in \mathbb{R}^n$, onde a distância entre pontos x e y pode ser definida através de várias métricas, como a distância Euclidiana, as amostras são organizadas como um conjunto de vetores ordenados:

$$X = \{X_1, X_2, \ldots, X_k, \ldots\},\ X_k \in \mathbb{R}^n,\ k \in \mathbb{N}$$

Cada amostra X_k representa o sistema em um momento específico k, permitindo que essa metodologia seja adaptada a conjuntos de dados com diferentes dimensionalidades n Smith (2021). Isso possibilita o cálculo de distâncias para fluxos de dados em diferentes instantes.

- A proximidade acumulada (π) de um ponto x é definida pela Equação (2.1) (ANGELOV, 2014).

$$\pi_k(x) = \sum_{i=1}^{k} d(x, x_i), k \geq 2 \tag{3.1}$$

- A excentricidade de uma amostra em um determinado momento k é calculada pela razão entre sua proximidade acumulada e a soma das proximidades acumuladas das demais amostras, conforme ilustrado na Equação (2.2) do trabalho de (ANGELOV, 2014).

$$\xi_k(x) = \frac{2\pi_k(x)}{\sum_{i=1}^{k} \pi_k(x_i)}, k \geq 2, \sum_{i=1}^{k} \pi_k(x_i) > 0 \tag{3.2}$$

- A tipicidade (Γ) é o complemento da excentricidade e é definida por:

$$\Gamma_k(x) = 1 - \xi_k(x) \tag{3.3}$$

  Vale ressaltar que tanto a tipicidade (Γ) quanto a excentricidade (ξ) são determinadas com base em um conjunto mínimo de três amostras distintas ($k \geq 3$), pois qualquer par de amostras não idênticas apresenta níveis iguais de excentricidade e tipicidade segundo os princípios do TEDA.

Armazenar amostras pode ser oneroso e há limitações na maioria dos dispositivos. Assim, tanto a excentricidade quanto a tipicidade podem ser calculadas recursivamente, eliminando a necessidade de manter um banco de dados com registros passados. Apenas a última amostra recebida e os valores representativos do sistema anterior são utilizados. A forma recursiva é expressa pela equação:

$$\xi_k(x) = \frac{1}{k} + \frac{(\mu_k - x_k)^T(\mu_k - x_k)}{k\sigma_k^2} \tag{3.4}$$

Em que a média (μ) e a variância (σ^2) são representadas pelas seguintes equações:

$$\mu_k(x) = \frac{k-1}{k}\mu_{k-1} + \frac{1}{k}x_k, \quad \mu_1 = x_1 \tag{3.5}$$

$$\sigma_k^2(x) = \frac{k-1}{k}\sigma_{k-1}^2 + \frac{1}{k-1}\|x_k - \mu_k\|^2, \quad \sigma_1^2 = 0 \tag{3.6}$$

Essa forma recursiva é fundamental para otimizar o uso de memória e poder computacional, especialmente em dispositivos de borda (edge devices) como os utilizados em aplicações IoT. Ela permite que o sistema calcule excentricidade e tipicidade de maneira eficiente e sem a necessidade de manter grandes volumes de dados.

3.6.1 Detecção de Outliers com TEDA

A Desigualdade de Excentricidade derivada do TEDA estabelece um limiar para a detecção de outliers, considerando o conceito estatístico da desigualdade de Chebyshev. Esta última assegura que não haverá mais do que $\frac{1}{n^2}$ amostras que excederão uma distância de $m\sigma$ em relação à média, onde n representa o parâmetro de sensibilidade que controla a identificação de valores atípicos (quanto maior seu valor, menor a sensibilidade do método e consequentemente menos outliers serão detectados) e m é o multiplicador do desvio padrão que determina a distância mínima, em termos de desvios padrão, que um ponto precisa estar da média para ser considerado um outlier. Entretanto, a Desigualdade de Excentricidade oferece resultados semelhantes sem impor suposições rigorosas sobre a distribuição ou independência dos dados. Essa desigualdade é definida pela equação apresentada em (DOE, 2020):

$$\zeta_k(x) > \frac{n^2+1}{2k} \tag{3.7}$$

Em que $\zeta_k(x)$ representa a excentricidade normalizada dos dados e é calculada pela equação seguinte:

$$\zeta_k(x) = \frac{\xi_k(x)}{2}, \quad \sum_{i=1}^{k} \zeta_i(x) = 1, k \geq 2 \tag{3.8}$$

Aqui, n indica o parâmetro que define a sensibilidade utilizada no limiar; quanto maior for o valor de n, menor será a sensibilidade do método e menos outliers serão detectados. A escolha adequada deste parâmetro é crucial para o desempenho do algoritmo, pois valores muito baixos podem resultar em falsos positivos (identificação incorreta de dados normais como outliers), enquanto valores muito altos podem levar a falsos negativos (não identificação de verdadeiros outliers). O pseudocódigo do TEDA, atualizado com base na abordagem recursiva, está apresentado no algoritmo abaixo:

Algoritmo 1 TEDA: IDENTIFICAÇÃO DE VALORES ATÍPICOS.

X está ativo Ler a amostra $x_k \in X$ $k == 1$ $\mu_k \leftarrow x_k$ Inicializa a média com a primeira amostra $\sigma_k^2 \leftarrow 0$ Variância inicial é zero Atualizar μ_k usando a média recursiva das amostras anteriores
Atualizar σ_k^2 usando a variância recursiva
Calcular a distância $d(x_k, \mu_k)$
Calcular a distância média d_{avg} das amostras em relação ao centróide
Calcular excentricidade: $\xi_k \leftarrow \frac{d(x_k,\mu_k)}{d_{avg}}$ $\xi_k > ECCENTRICITY\_THRESHOLD$ valor_atipico $\leftarrow$ verdadeiro Armazenar x_k em dados removidos valor_atipico $\leftarrow$ falso Adicionar x_k aos dados válidos $k \leftarrow k + 1$ Incrementa o contador de amostras

Neste pseudocódigo, foi incluída a abordagem recursiva para o cálculo da média e variância das amostras, de forma a otimizar o armazenamento e o tempo de processamento, especialmente em aplicações de borda (*edge computing*) onde os recursos são limitados. Esta abordagem recursiva permite que o algoritmo processe os dados de forma contínua, atualizando suas estatísticas sem a necessidade de armazenar todo o histórico de dados, o que é particularmente importante em sistemas com restrições de memória ou que necessitam de processamento em tempo real.

4 Materiais e Métodos

Este capítulo apresenta os componentes físicos utilizados no projeto e detalha os métodos empregados para seu desenvolvimento. A primeira seção lista e descreve todos os materiais necessários para a construção do sistema, incluindo o drone, sensores e componentes eletrônicos. A segunda seção explica como esses elementos foram integrados e programados para criar um sistema funcional.

4.1 Materiais

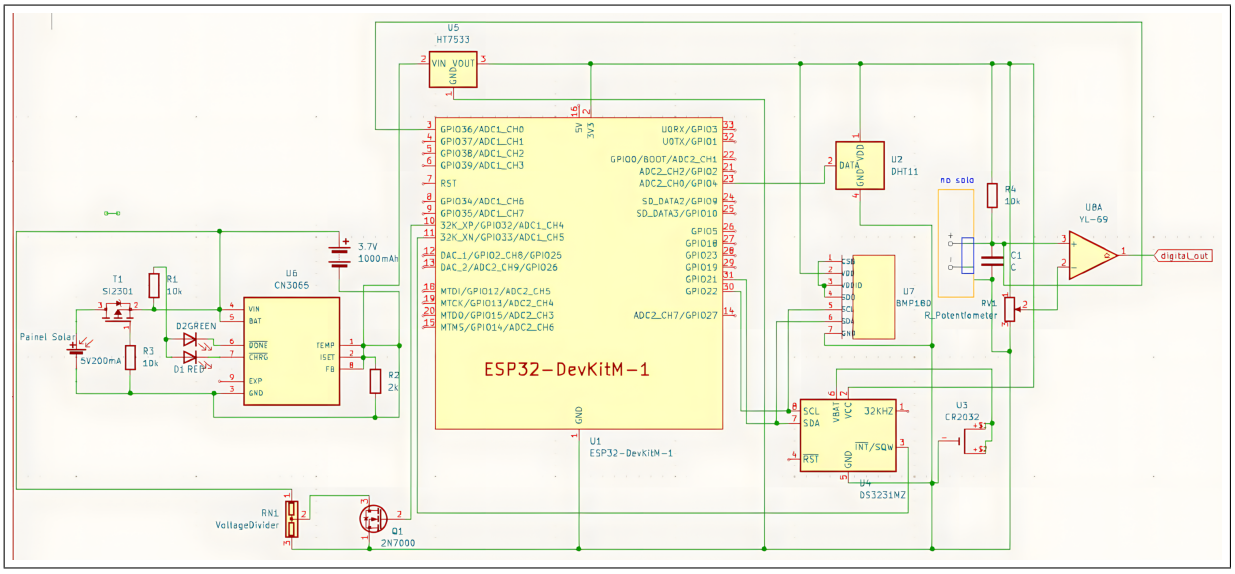
Os itens utilizados para o circuito que irá coletar os dados e os itens utilizados para o circuito que será fixado no drone:

1. **Drone E88 Pro Com Câmera**
2. **Microcontroladores ESP32**
    - Usados como nós sensores e como unidade de controle central no drone.
3. **Sensores**
    - **DHT11**: Sensor de temperatura e umidade.
    - **BMP180**: Sensor barométrico (pressão atmosférica e temperatura).
    - **YL-69**: Sensor de umidade do solo.
4. **Baterias de Íon-Lítio (3,7V)**
    - Fonte de alimentação para os dispositivos ESP32.
5. **Reguladores de Tensão (3,3V)**
    - **Conversor Buck** ou **LDO HT7833**: Reguladores utilizados para converter a tensão das baterias de 3,7V para 3,3V, adequada para a ESP32.
6. **Circuito de Monitoramento da Bateria**
    - **Transistor 2N7000**: Utilizado para habilitar a leitura do nível de bateria.
    - **Divisor de Tensão**: Implementado com resistores para adequar a tensão ao ADC da ESP32.
7. **Transistor 2N2222**
    - Utilizado para controlar o buzzer no drone, fornecendo indicações sonoras.

8. **Buzzer**
    - Dispositivo de alerta sonoro para indicar o status de comunicação ESP-NOW, Wi-Fi e envio de dados via MQTT.
9. **Placa Solar (5 V 200 mA)**
    - Usada para carregar as baterias de íon-lítio das ESP32 equipadas com sensores.
10. **Sistema de Gerenciamento de Carga - CN3065**
    - Controlador de carregamento de baterias através da energia solar.
11. **Diodo Schottky 1N5819**
    - Usado como bloqueio para evitar retorno de corrente no circuito de carregamento solar.

Resultando no circuito da figura 5, e montado em protoboard como na figura 6.

Figura 5 – Diagrama representando o circuito



Fonte: Autoria Própria

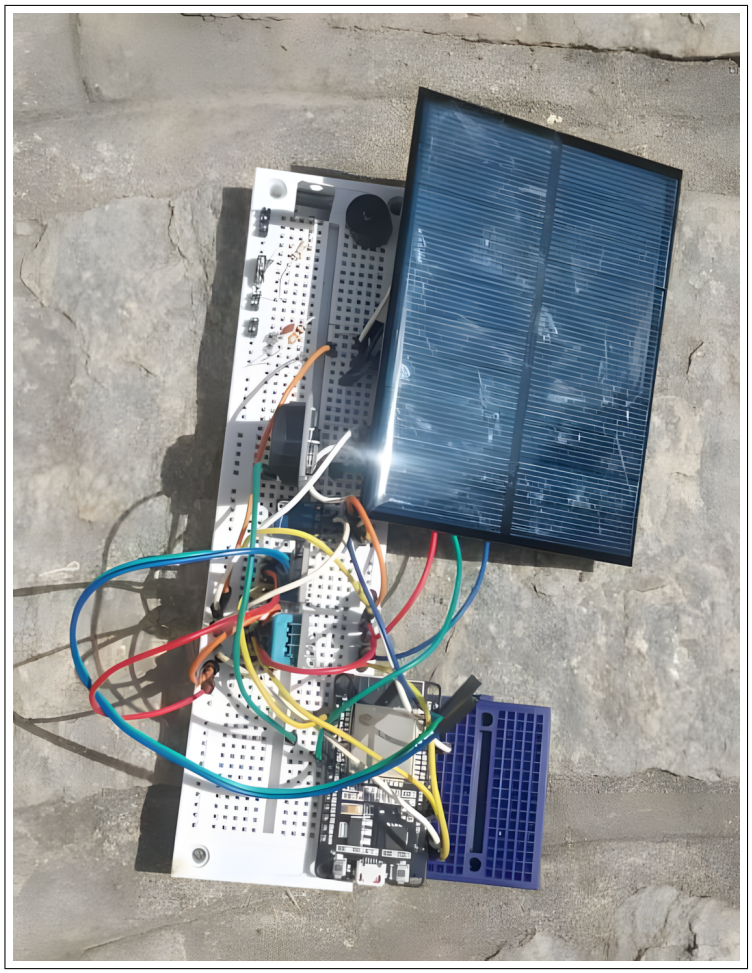


Figura 6 – Circuito montado do sistema com os sensores conectados à ESP32 para coletar dados

Fonte: Autoria Própria

4.2 Métodos

4.2.1 Coleta e Transmissão de Dados Ambientais com Rede de Microcontroladores ESP32 e Drone

A necessidade de monitoramento ambiental em áreas remotas exige soluções eficientes e autônomas para a coleta e transmissão de dados. Este projeto propõe o uso de microcontroladores ESP32 em rede, associados a sensores ambientais, para a aquisição de dados que são posteriormente transmitidos a um nó central embarcado em um drone. O drone coleta os dados via protocolo ESP-NOW e os envia para a AWS (Amazon Web Services) via MQTT para armazenamento e análise, o drone com o circuito e microcontrolador pode ser visualizada na Figura 7.

Figura 7 – Drone com circuitos fixados

Fonte: Autoria Própria

4.2.2 Arquitetura do Sistema

4.2.2.1 ESP32 como Armazenamento de Dados e Sensores

Cada nó sensor é composto por uma ESP32 equipada com diversos sensores ambientais, responsáveis pela coleta de dados. A ESP32 foi escolhida pela sua versatilidade, suporte a diferentes interfaces de comunicação e baixo consumo energético.

4.2.2.1.1 Sensores Utilizados

- **DHT11**: Sensor de temperatura e umidade, amplamente utilizado por sua simplicidade e precisão em leituras básicas de ambiente.
- **BMP180**: Sensor barométrico que permite medir a pressão atmosférica, além de fornecer dados adicionais de temperatura.
- **YL-69**: Sensor de umidade do solo, adequado para monitoramento de áreas agrícolas e ambientes que demandam controle da umidade.

4.2.2.1.2 Modularidade do Sistema

A modularidade do sistema é um aspecto crucial. Como consequência da estrutura de software desenvolvida, é possível implementar novos sensores com facilidade. Cada sensor é tratado por uma classe específica no código, permitindo que a adição de um novo sensor requeira apenas a inclusão do driver adequado para aquele dispositivo. Isso garante que o sistema seja flexível e expansível, acomodando diferentes tipos de sensores conforme a necessidade.

4.2.2.2 Alimentação dos Dispositivos

Para a alimentação das ESP32, utilizamos baterias de íon-lítio de 3,7 V. Como a ESP32 opera a 3,3 V, é necessário o uso de reguladores de tensão para adaptar a voltagem fornecida pela bateria.

4.2.2.2.1 Reguladores de Tensão Utilizados

Os reguladores utilizados são:

- **Conversor Buck ou LDO HT7833**: Reguladores eficientes para converter a tensão da bateria de 3,7 V para 3,3 V, garantindo o funcionamento estável da ESP32 sem sobrecarregar o circuito.

4.2.2.2.2 Sistema de Monitoramento da Bateria

As ESP32 equipadas com sensores contam com um sistema de monitoramento da bateria, composto por:

- **Divisor de Tensão e transistor 2N7000**: Implementado com resistores para reduzir a tensão medida para um nível seguro, compatível com a entrada analógica (ADC) da ESP32, sendo ativo por meio de um pino GPIO.

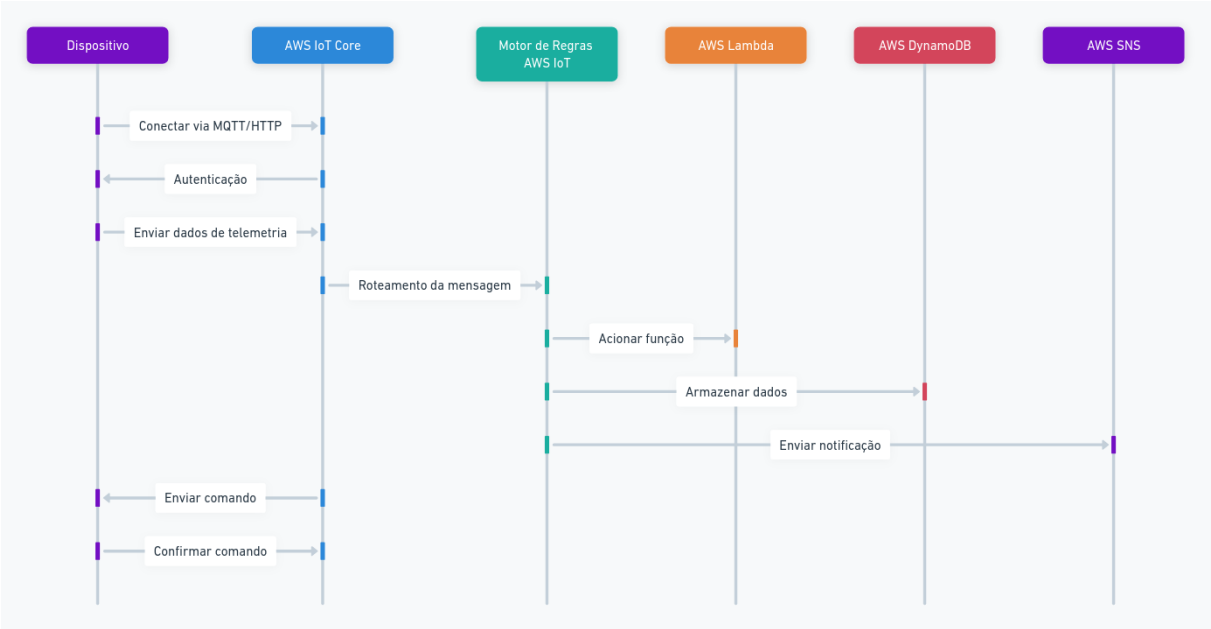
4.2.2.3 Configuração da ESP32 no Drone

A ESP32 instalada no drone atua como o nó central de comunicação. Ela é responsável por receber os dados transmitidos via protocolo ESP-NOW dos outros dispositivos ESP32 e, em seguida, retransmitir esses dados utilizando o protocolo MQTT para a AWS, onde serão armazenados e analisados.

4.2.2.3.1 Comunicação via ESP-NOW e MQTT

O protocolo ESP-NOW permite a comunicação direta entre as placas ESP32, sem a necessidade de um roteador ou ponto de acesso, garantindo baixo consumo de energia e alta eficiência em áreas sem infraestrutura de rede. Após receber os dados, a ESP32 no drone utiliza o protocolo MQTT para enviar os dados para a AWS IoT Core, como mostrado no diagrama de sequência da Figura 8, um serviço que permite conectar dispositivos IoT à nuvem de forma segura e escalável.

Figura 8 – Comunicação AWS



Fonte: Autoria Própria

4.2.2.3.2 Sistema de Alerta com Buzzer

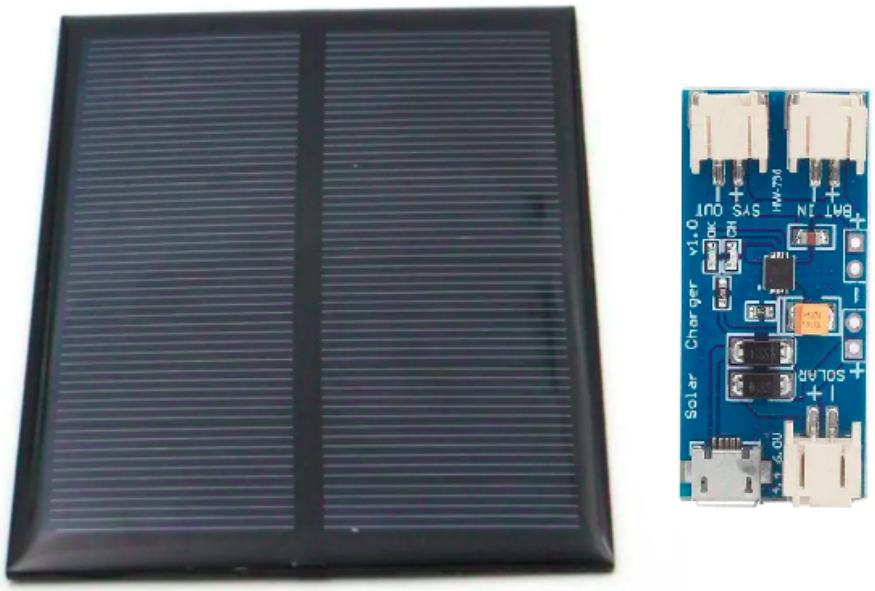
Para fornecer um feedback simples e eficiente sobre o status da comunicação, é implementado um sistema de alerta utilizando um buzzer. O controle do buzzer é feito através de um transistor 2N2222, que permite a ativação do dispositivo quando eventos específicos ocorrem, como:

- **Comunicações ESP-NOW**: Quando a ESP32 recebe dados dos sensores distribuídos.
- **Conexão Wi-Fi**: Quando a ESP32 no drone está conectada à internet para envio dos dados.
- **Envio via MQTT**: Quando os dados são enviados com sucesso para a AWS.

4.2.2.4 Monitoramento e Energia Sustentável

Para prolongar a autonomia dos nós sensores em campo, cada ESP32 é alimentada por uma placa solar de 5 V 200 mA. O sistema de gerenciamento de carga utilizado é o CN3065, oferecendo funções como controle de corrente de carga, proteção contra sobrecarga e monitoramento de tensão. Exemplificado tais componentes na Figura 9. Para proteger o circuito contra o retorno de corrente e possíveis danos, utilizamos um Diodo Schottky 1N5819 como bloqueio.

Figura 9 – Placa solar e circuito de gerenciamento de carga utilizado para alimentar as ESP32.

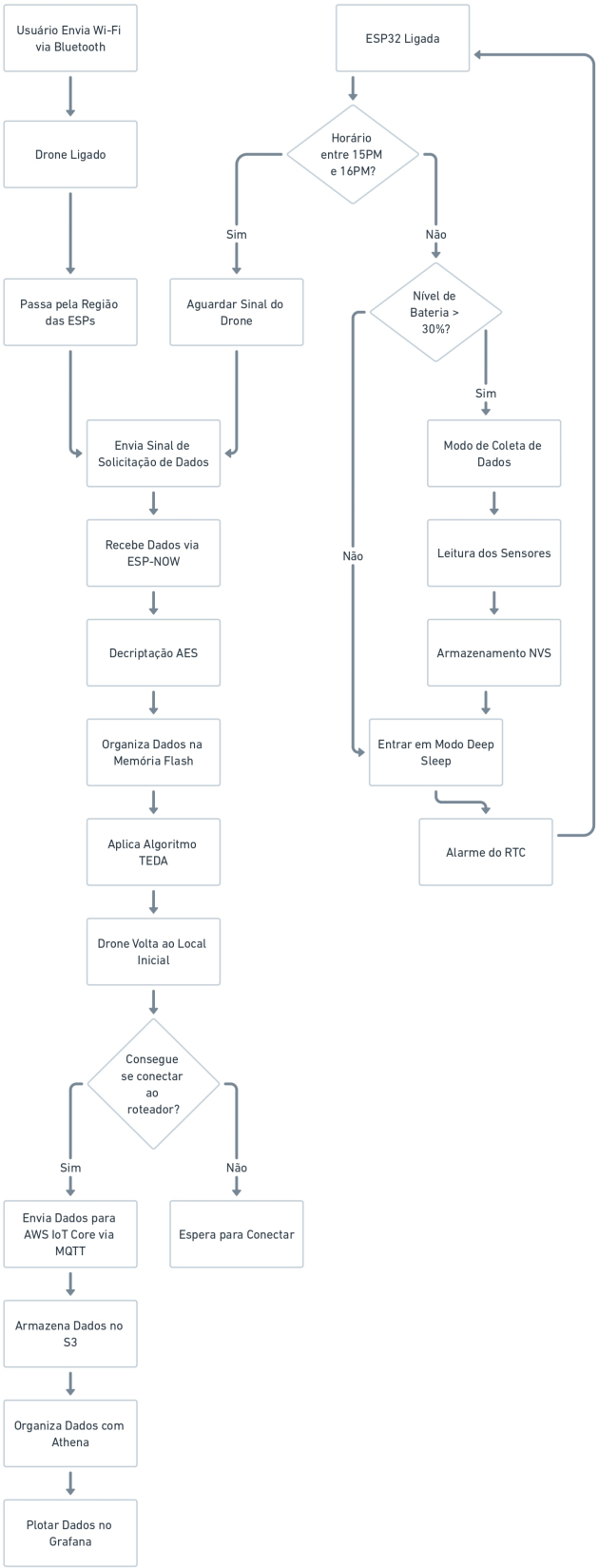


Fonte: Autoria Própria

4.2.3 Metodologia

Este projeto utiliza uma placa para o coletador de dados e outra placa fixada no drone, cada uma com uma função distinta, mas interconectadas para coletar, transmitir, processar e enviar dados ambientais. A seguir, foram detalhadas as etapas e os métodos utilizados em cada uma das ESP32, explicando o ciclo de operação, monitoramento de bateria, comunicação via ESP-NOW, processamento com TEDA e envio de dados para a AWS via MQTT. O fluxograma do sistema é como na Figura 10.

Figura 10 – Fluxograma do processo completo entre ESP32 e drone, desde a coleta de dados até o envio para a AWS.



4.3 Implementação dos Códigos

Nesta seção, são descritas as principais implementações dos códigos utilizados no projeto, separadas em duas partes principais: o código do **coletador de dados**, que envolve a coleta de dados de sensores, criptografia AES, e sincronização com RTC; e o código da **esp drone**, que implementa a comunicação via ESP-NOW com criptografia AES, integração com AWS IoT, e provisionamento via Bluetooth.

4.3.1 Código do *coletador de dados*

O código do *coletador de dados* é responsável pela coleta de dados de sensores conectados ao ESP32, além da implementação de criptografia com AES e sincronização de tempo utilizando RTC. Este código foi desenvolvido com modularidade, facilitando a adição de novos sensores e garantindo a segurança dos dados transmitidos.

4.3.1.1 Modularidade dos Sensores

A coleta de dados dos sensores é realizada de forma modular, com cada sensor encapsulado em classes ou funções individuais. A estrutura modular do código permite a fácil implementação de diferentes sensores e protocolos de comunicação. Isso é alcançado através de uma classe base `ESP32Base` e uma classe derivada `ESP32DerivedClass`.

- Facilidade de expansão: Novos sensores podem ser adicionados simplesmente criando uma nova função de leitura e adicionando o sensor à lista.
- Separação de responsabilidades: Cada sensor tem sua própria função de leitura, tornando o código mais organizado e fácil de manter.
- Flexibilidade: Diferentes protocolos de comunicação (I2C, SPI, etc.) podem ser implementados nas funções específicas de cada sensor.

Listing 4.1 – Estrutura da Classe ESP32DerivedClass

```
class ESP32DerivedClass : public ESP32Base {
public:
    void readSensorData() override;
    void formatData(const std::string& sensorName,
                    float sensorValue);
    void saveDataToNVS(const std::string& sensorName,
                       const std::string& sensorData);
    void readAllDataFromNVS();
private:
    // Static member variables para dados dos sensores
```

```
    static std::vector<std::string> bmp180Data;
    static std::vector<std::string> dht11Data;
    static std::vector<std::string> soilMoistureData;
    static std::vector<std::string> ky037Data;
    // Funcoes de leitura para cada sensor
    void readBMP180();
    void readDHT11();
    void readSoilMoisture();
    void readKY037();
};
```

As principais características desta estrutura modular incluem:

- **Polimorfismo**: A função `readSensorData()` é virtual na classe base e sobrescrita na classe derivada, permitindo comportamentos específicos para diferentes tipos de ESP32.
- **Funções específicas para sensores**: Cada sensor tem sua própria função de leitura (ex: `readBMP180()`, `readDHT11()`), o que facilita a adição de novos sensores.
- **Armazenamento de dados**: Vetores estáticos são usados para armazenar dados de cada sensor, permitindo fácil acesso e manipulação.
- **Flexibilidade**: A função `readSensorData()` usa uma lista de sensores, facilitando a adição ou remoção de sensores sem alterar o código principal.

Listing 4.2 – Implementação da função readSensorData()

```
void ESP32DerivedClass::readSensorData() {
    std::vector<SensorInfo> sensorList = getSensorsList();
    for (const auto& sensor : sensorList) {
        if (sensor.name == "BMP180") {
            readBMP180();
        } else if (sensor.name == "DHT11") {
            readDHT11();
        } // ... outros sensores
    }
}
```

4.3.1.2 Uso de Mutex e Queue

Para evitar conflitos de acesso ao barramento de comunicação I^2C, foram utilizados mutexes. Esses mutexes garantem que os sensores possam ser lidos sem interferência.

Além disso, filas (*queues*) são utilizadas para gerenciar a comunicação assíncrona entre as tarefas, permitindo que os dados dos sensores sejam processados e transmitidos de forma independente.

Listing 4.3 – Exemplo de Uso de Mutex no *coletador de dados*

```
// Toma controle do recurso
xSemaphoreTake(sensor_mutex, portMAX_DELAY);
// Realiza a leitura do sensor
bmp180_sensor.read_data();
// Libera o controle do recurso
xSemaphoreGive(sensor_mutex);
```

4.3.1.3 Sincronização de Tempo com RTC e NTP

O código também integra um RTC (DS3231) para manter a data e hora corretas, mesmo quando o dispositivo não está conectado à internet. Se o RTC não estiver corretamente configurado (ano menor que 2020), o dispositivo tentará sincronizar o tempo via NTP, conectando-se à internet e ajustando a hora automaticamente.

Listing 4.4 – Sincronização com NTP

```
void synchronize_time_with_ntp() {
    ESP_LOGI(TAG, "Connecting to Wi-Fi to sync with NTP...");

    // Configuracoes do NTP
    esp_sntp_setoperatingmode(ESP_SNTP_OPMODE_POLL);
    esp_sntp_setservername(0, "pool.ntp.org");
    esp_sntp_init();

    time_t now = 0;
    struct tm timeinfo = { 0 };
    int retry = 0;
    const int retry_count = 10;

    while (timeinfo.tm_year < (2020 - 1900) &&
           ++retry < retry_count) {
        ESP_LOGI(TAG, "Waiting for time to be set... (%d/%d)",
                 retry, retry_count);
        vTaskDelay(2000 / portTICK_PERIOD_MS);
        time(&now);
        localtime_r(&now, &timeinfo);
    }
```

```
}
```

4.3.1.4 Criptografia AES no *coletador de dados*

A criptografia AES foi implementada para proteger os dados sensíveis transmitidos entre os dispositivos. A chave de 128 bits e o vetor de inicialização (IV) são utilizados tanto no dispositivo transmissor quanto no receptor, garantindo que a comunicação seja encriptada e desencriptada de forma segura.

Listing 4.5 – Chaves e IV do AES no *coletador de dados*

```
unsigned char aes_key[16] = {
    0x12, 0x34, 0x56, 0x78, 0x90, 0xAB, 0xCD, 0xEF,
    0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF
};

unsigned char aes_iv[16] = {
    0xFE, 0xDC, 0xBA, 0x98, 0x76, 0x54, 0x32, 0x10,
    0xFF, 0xEE, 0xDD, 0xCC, 0xBB, 0xAA, 0x99, 0x88
};
```

4.3.2 Código da *esp drone*

O código da *esp drone* lida com a comunicação sem fio via ESP-NOW, a criptografia AES para proteger os dados, a integração com a AWS IoT para envio de dados à nuvem, e o provisionamento de credenciais Wi-Fi via Bluetooth.

4.3.2.1 Criptografia AES na *esp drone*

Assim como no *coletador de dados*, o código da *esp drone* também utiliza a criptografia AES para garantir a segurança dos dados transmitidos entre dispositivos ESP32. A mesma chave AES e vetor de inicialização são usados em ambos os dispositivos, permitindo que a transmissão e recepção de dados sejam feitas de maneira segura e confiável.

4.3.2.2 Integração com AWS IoT

Para enviar os dados do drone à nuvem, o código integra-se com a plataforma AWS IoT utilizando o protocolo MQTT. O cliente MQTT é configurado com os certificados e chaves da AWS, permitindo uma conexão segura com o serviço de nuvem.

Listing 4.6 – Configuração do Cliente MQTT para AWS IoT

```
void initialize_aws() {
```

```
    esp_mqtt_client_config_t mqtt_cfg = {
        .uri = "mqtts://<AWS_ENDPOINT>.amazonaws.com",
        .cert_pem = (const char *)ca_cert_pem_start,
        .client_cert_pem = (const char *)client_cert_pem_start,
        .client_key_pem = (const char *)client_key_pem_start,
    };

    mqtt_client = esp_mqtt_client_init(&mqtt_cfg);
    esp_mqtt_client_start(mqtt_client);
}
```

4.3.2.3 Provisionamento via Bluetooth (BLE)

Uma funcionalidade adicional importante é o uso de Bluetooth para provisionamento de Wi-Fi. Isso permite que o usuário configure o SSID e a senha da rede Wi-Fi sem precisar reprogramar o dispositivo. Essa configuração é feita via BLE (Bluetooth Low Energy), onde o dispositivo ESP32 escuta as credenciais e as armazena para futuras conexões. O aplicativo possui o seguinte ícone da figura 11.

Figura 11 – Aplicativo utilizado para comunicação entre bluetooth de smartphone e ESP32

Fonte: ESPRESSIF SYSTEMS. ESP BLE Provisioning. Google Play Store, 2024. Disponível em: https://play.google.com/store/apps/details?id=com.espressif.provble.

4.3.3 Uso do LittleFS para Organização de Dados

O LittleFS é utilizado para organizar as informações recebidas via ESP-NOW, proporcionando uma estrutura de arquivos eficiente e fácil de gerenciar. O *LittleFS* foi escolhido para o sistema de armazenamento devido às suas vantagens em dispositivos embarcados com memória Flash, como o ESP32, em comparação a outros sistemas de arquivos, como FAT ou SPIFFS. As principais razões para essa escolha incluem:

- **Eficiência em Flash**: O *LittleFS* foi projetado especificamente para otimizar operações de leitura e escrita em memórias Flash, com suporte a *wear leveling*, o que prolonga a vida útil da memória ao distribuir uniformemente as gravações.
- **Tolerância a Falhas**: O *LittleFS* oferece uma alta tolerância a falhas, garantindo a integridade dos dados, mesmo após desligamentos inesperados ou quedas de energia, o que é essencial em sistemas críticos.
- **Baixo Overhead**: O sistema possui um baixo consumo de memória e recursos, sendo adequado para dispositivos com recursos limitados, como o ESP32, permitindo a criação de sistemas de arquivos eficientes sem comprometer o desempenho.

4.3.3.1 Estrutura de Diretórios

A estrutura de diretórios no LittleFS é organizada da seguinte forma:

```
/
 AA:BB:CC:DD:EE:FF/
    data.json
 11:22:33:44:55:66/
    data.json
 ...
```

Cada endereço MAC de dispositivo ESP32 tem seu próprio diretório, contendo um arquivo `data.json` com os dados mais recentes daquele dispositivo.

Esta estrutura permite um gerenciamento eficiente dos dados de múltiplos dispositivos ESP32, facilitando o acesso e a atualização das informações de cada sensor individualmente.

5 Resultados

5.1 Consumo de energia dos coletadores de dados

Primeiramente, foi realizado um cálculo teórico de consumo de energia para o sistema de coleta de dados utilizando a ESP-WROOM-32 (DevKit), três sensores (BMP180, DHT11, YL-69), e módulo RTC DS3231 com bateria própria.

O primeiro passo foi calcular o consumo total do sistema utilizando apenas a bateria. Considerou-se o ciclo de operação da ESP32, incluindo tempos de *deep sleep* para economizar energia. Cada sensor foi ativado apenas nos períodos necessários, minimizando o tempo em que eles permanecem em operação.

Com base nesses dados, foi possível calcular o tempo de funcionamento total do sistema até que a bateria se esgote completamente. Esse cálculo forneceu uma estimativa do tempo de operação do sistema utilizando apenas a energia armazenada na bateria de 1000 mAh.

5.1.1 Consumo dos Sensores e do RTC

Para cada sensor, o consumo foi calculado com base no tempo em que estão ativos e inativos. Considerando o ciclo de trabalho de 15 segundos por hora, cada sensor consome energia apenas durante esse período.

- BMP180: Sensor de pressão e temperatura, ativado por 15 segundos por hora.
- DHT11: Sensor de umidade e temperatura, também ativado por 15 segundos por hora.
- YL-69: Sensor de umidade do solo, ativado no mesmo ciclo.
- O módulo RTC DS3231, que é responsável pela manutenção do tempo, consome aproximadamente 0.11mA continuamente, contribuindo de forma mínima para o consumo total.

5.1.2 Consumo do ESP32

Com base nas novas informações sobre o ciclo de trabalho:

- Modo Ativo (durante captura de dados): 15 segundos a cada 3600 segundos, será o tempo em que os sensores estarão processando os dados externos

- Modo Deep Sleep: 3585 segundos por hora (3600 - 15)
- Modo Wi-Fi Ativo: 1 hora por dia (entre 15:00 e 16:00)

Consumo em diferentes modos (SYSTEMS, 2021):

- Modo Ativo: 160 mA
- Modo Deep Sleep: 10 µA
- Modo Wi-Fi Ativo: 240 mA (estimativa conservadora para transmissão de dados)

5.1.3 Consumo Total e Duração da Bateria

Com base nos cálculos teóricos, estimou-se que a ESP32, junto aos sensores e o RTC, consome uma média de 14,11 mA durante o seu ciclo de trabalho. Este valor foi obtido somando o consumo médio em cada modo de operação da ESP32 e o consumo dos sensores e do RTC.

Considerando uma bateria de 1000 mAh e a eficiência do regulador de tensão HT7833, a autonomia teórica do sistema foi calculada em 2,66 dias. Essa autonomia leva em conta o uso otimizado do modo *deep sleep*, que reduz significativamente o consumo energético da ESP32 nos períodos em que os sensores não estão coletando dados.

- Consumo total médio = 13,3 mA (ESP32) + 0,6 mA (Sensores) + 0,1 mA (RTC) ≈ 14 mA
- Capacidade da bateria: 1000 mAh. Considerando uma bateria de íon-lítio de 3,7V/1000mAh, deve-se observar que a capacidade útil é reduzida devido ao limite mínimo de tensão do LDO (3,3V). Assim, utilizando aproximadamente 80% da capacidade nominal da bateria e considerando o consumo real com a eficiência do regulador:
  - Capacidade útil = 1000 mAh × 0,80 = 800 mAh
  - Consumo real = 14 mA ÷ 0,8 ≈ 17,5 mA
  - Duração teórica da bateria = 800 mAh ÷ 17,5 mA ≈ 45,7 horas ≈ 1 dia e cerca de 22 horas
- Considerando a eficiência do regulador HT7833 (90%): O regulador LDO HT7833, que fornece uma saída regulada de 3,3V, apresenta uma eficiência média de aproximadamente 87 a 90% Holtek Semiconductor Inc. (2021). Esta alta eficiência é alcançada devido à pequena diferença entre a tensão de entrada (bateria) e a tensão de saída regulada, minimizando as perdas por dissipação no regulador.
- Duração ajustada = 45,7 × 0,9 = 41,13 horas ≈ 1 dia e cerca de 17 horas

5.1.4 Inclusão da Placa Solar e Controlador de Carga

Em seguida, foi incluída uma placa solar de 5 V e 200 mA, com dimensões de 100 x 69mm, juntamente com um controlador de carga CN3065. Essa configuração permitiu que a energia solar fosse utilizada para carregar a bateria, prolongando assim o tempo de operação do sistema.

Para estimar com precisão a geração de energia solar em João Pessoa, foi realizada uma simulação detalhada considerando a latitude local (-7,115°), as variações sazonais e os efeitos atmosféricos. O modelo de simulação levou em conta a declinação solar, o ângulo horário e aplicou um fator de transmissão atmosférica de 0,7 para representar as perdas, adotando uma abordagem conservadora.

Os resultados da simulação indicaram que o sistema pode gerar, em média, aproximadamente 7,5 Wh por dia nos equinócios e solstícios, com variações ao longo do ano devido às mudanças sazonais. A corrente média gerada foi estimada em aproximadamente 32,42 mA.

Para validar o balanço energético diário do sistema, foi realizada uma simulação numérica com base nos dados obtidos. A função de geração de energia solar foi modelada levando em consideração o perfil de irradiância ao longo do dia, enquanto o consumo foi considerado constante, baseado nas estimativas anteriores de 14 mA.

O código utilizado para realizar a simulação foi o seguinte:

Algoritmo 2 Simulação de Balanço Energético Solar para Placa de 5V 200mA(100 x 69mm)

```
Importar bibliotecas numpy e matplotlib.pyplot
Definir função solar_irradiance(hour, day_of_year, latitude)
Definir função simulate_solar_production(date, latitude, panel_area, panel_efficiency)
Definir parâmetros: latitude, dimensões do painel, eficiência
Simular para equinócios e solstícios
Calcular média diária de produção de energia
```

5.1.5 Conclusões Parciais

A análise dos resultados mostra que o sistema, quando alimentado apenas pela bateria de 1000 mAh, pode operar por até 2,66 dias em condições ideais. Com a inclusão da placa solar, a autonomia do sistema é significativamente aumentada.

A geração média diária de aproximadamente 7,5 Wh pela placa solar, comparada ao consumo médio de 14 mA (equivalente a 338,64 mWh/dia), resulta em um excedente energético, permitindo que o sistema funcione de forma autossustentável em condições normais de exposição solar.

Essa energia é suficiente para carregar a bateria de 1000 mAh e ainda gerar um excedente energético, permitindo que o sistema funcione de forma autossustentável por períodos prolongados. Em dias nublados ou de menor exposição solar, a geração de energia será menor, mas a placa solar ainda contribui para aumentar a autonomia do sistema.

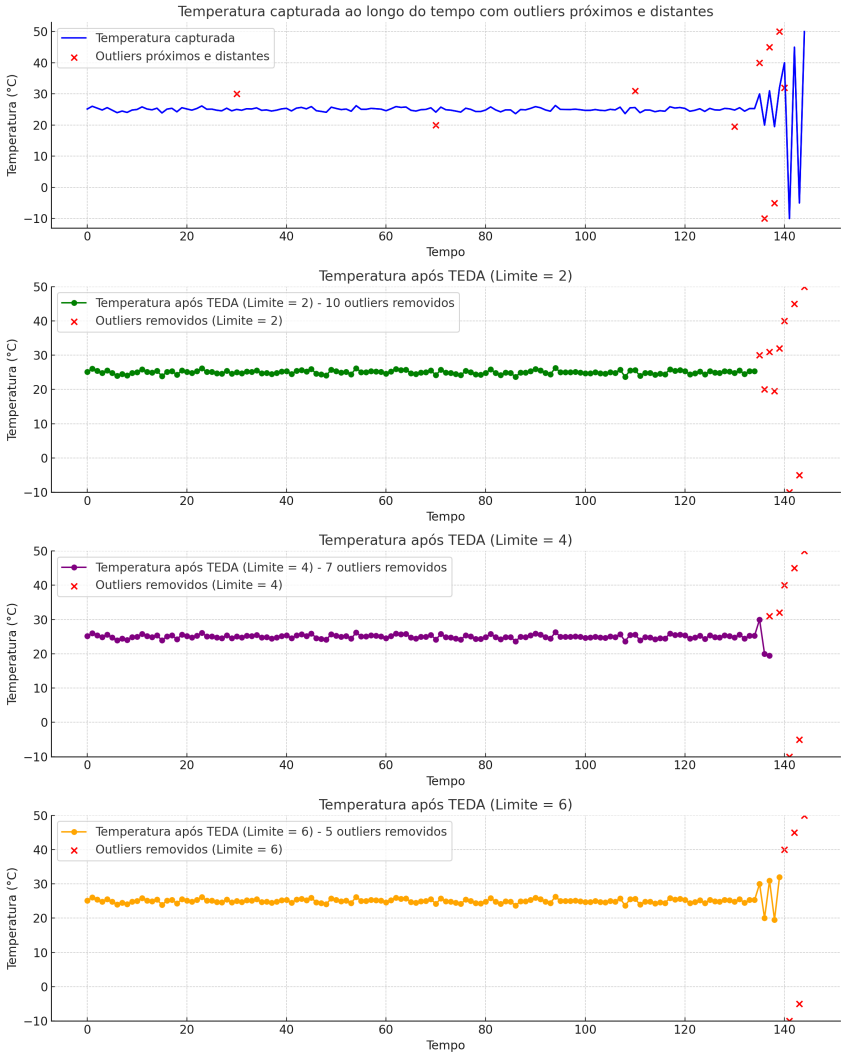
5.2 Efetividade do algoritmo TEDA

Para validar o algoritmo, foi realizada uma simulação de captura de temperatura, onde foram incluídos outliers visíveis no final da coleta de dados. A seguir, o TEDA foi aplicado com diferentes *thresholds*:

- **Gráfico 1**: Exibe a temperatura capturada com outliers.
- **Gráfico 2 (Threshold = 2)**: O TEDA removeu 10 outliers, mostrando alta sensibilidade.
- **Gráfico 3 (Threshold = 4)**: Com um *threshold* mais alto, 7 outliers foram removidos.
- **Gráfico 4 (Threshold = 6)**: Um *threshold* ainda maior resultou na remoção de apenas 5 outliers.

Os gráficos da Figura 12 demonstram como o TEDA ajusta sua sensibilidade à detecção de outliers com base no valor do *threshold* definido.

Figura 12 – Detecção de outliers com o algoritmo TEDA aplicado a dados de temperatura com diferentes *thresholds*.



Fonte: Autoria própria

5.2.1 Impacto do Threshold

Threshold baixo (2): Detecta e remove mais outliers, incluindo valores próximos do padrão.

Threshold médio (4): Remove uma quantidade moderada de outliers.

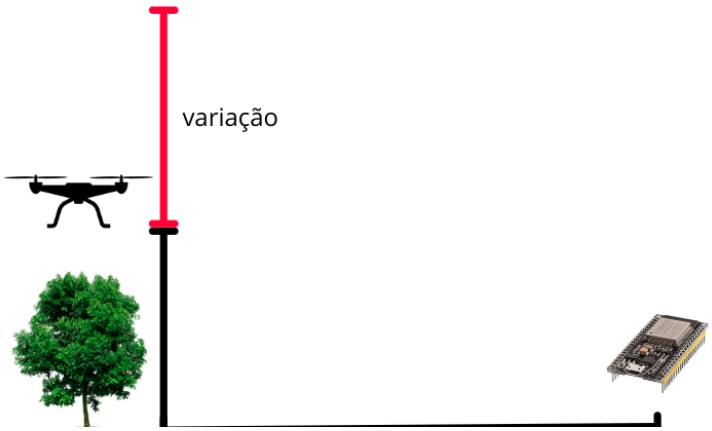
Threshold alto (6): Detecta apenas outliers extremos.

O valor do *threshold* deve ser ajustado conforme o nível de precisão necessário. Em aplicações que exigem rigor na análise de dados, um *threshold* menor pode ser mais adequado, enquanto um valor maior pode ser utilizado em contextos que toleram variações menores.

5.3 Teste de distância da ESP-NOW

Para testar a efetividade do protocolo em distâncias consideráveis, foi feito teste considerando a distância diagonal do coletador de dados, que estará em um local próximo ao chão, para o drone que estará com o dispositivo conectado por meio de travas. O drone subirá próximo a uma árvore para servir como referência de altura inicial, e posteriormente a única variação será a distância vertical do drone para essa mesma árvore e avaliar se a conexão foi efetivada ou não. Como representado de forma simplificada na Figura 13.

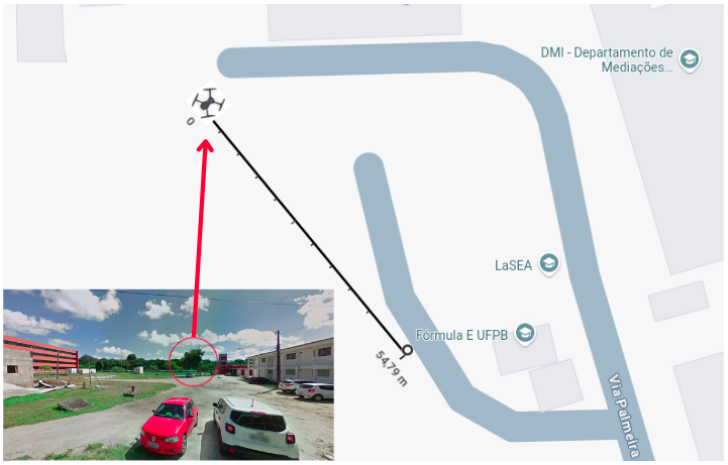
Figura 13 – Exemplificação do teste de distância



Fonte: Autoria própria

Para a distância horizontal entre o drone e o coletador de dados foi utilizado um tracker de GPS e a distância entre dois pontos verificando o Google MapsGoogle (2023), como há um erro inerente, foi feito o truncamento do valor para 50 metros. Como observado na imagem da Figura 14.

Figura 14 – Disposição no mapa e localização por foto



Fonte: Autoria própria

Para estimar a altura em que o drone se encontra em relação ao solo, foi utilizado a câmera de um celular em conjunto com os sensores de ângulo, como o acelerômetro e o giroscópio. A ideia é baseada no princípio de triangulação, onde o ângulo de inclinação do celular e a altura do observador são utilizados para calcular a altura do drone.

1. **Captura da Imagem do Drone**: O celular é posicionado de forma que a câmera aponte para o drone. A imagem da posição do drone é capturada em tempo real pela câmera.

2. **Leitura dos Sensores do Celular**: Simultaneamente, os sensores internos do celular (acelerômetro e giroscópio) medem o ângulo de inclinação do celular em relação ao solo. Esse ângulo, denotado por θ, é o ângulo entre o eixo do celular e a linha visual que aponta para o drone no céu.

3. **Altura do Usuário**: A altura do observador (altura do usuário que está segurando o celular) é um dado fixo no cálculo, denotado por H_{usuario}. Essa medida é a distância entre os olhos do observador e o solo.

4. **Cálculo da Altura do Drone**: Sabendo o ângulo de inclinação θ e a altura do observador H_{usuario}, podemos estimar a altura do drone, H_{drone}, através de uma relação trigonométrica simples utilizando a tangente do ângulo de inclinação. A fórmula para estimar a altura do drone é:

$$H_{\text{drone}} = H_{\text{usuario}} + (D \cdot \tan(\theta)) \tag{5.1}$$

   Onde:

   - H_{drone} é a altura estimada do drone,
   - H_{usuario} é a altura do observador,
   - D é a distância horizontal entre o observador e o ponto diretamente abaixo do drone,
   - θ é o ângulo de inclinação do celular em relação ao solo.

Com base na distância horizontal fixa de 50 metros, foram calculados a altura com base no valor de ângulo obtido da câmera (θ) e a distância da hipotenusa:

- Para um ângulo de inclinação (θ) de 3.78°:
  - Altura estimada: 5.00 m
  - Hipotenusa estimada: 50.11 m
  - RSSI estimado: −53.97 dBm

- Para um ângulo de inclinação (θ) de 5.6°:
  - Altura estimada: 6.60 m
  - Hipotenusa estimada: 50.24 m
  - RSSI estimado: −54.00 dBm
- Para um ângulo de inclinação (θ) de 22.11°:
  - Altura estimada: 22.01 m
  - Hipotenusa estimada: 53.97 m
  - RSSI estimado: −55.62 dBm
- Para um ângulo de inclinação (θ) de 35.88°:
  - Altura estimada: 37.87 m
  - Hipotenusa estimada: 61.71 m
  - RSSI estimado: −59.79 dBm
- Para um ângulo de inclinação (θ) de 47°:
  - Altura estimada: 55.32 m
  - Hipotenusa estimada: 73.31 m
  - RSSI estimado: −67.30 dBm
- Para um ângulo de inclinação (θ) de 64°:
  - Altura estimada: 104.22 m
  - Hipotenusa estimada: 114.06 m
  - RSSI estimado: −71.14 dBm

A fórmula utilizada para o cálculo do RSSI estimado em função da distância é baseada no modelo de perda de caminho em espaço livre, dado por:

$$RSSI = RSSI_{\text{ref}} - 10n \log_{10}\left(\frac{d}{d_{\text{ref}}}\right) \tag{5.2}$$

Em que:

- $RSSI_{\text{ref}}$ é o valor do RSSI de referência a uma distância $d_{\text{ref}} = 1\,\text{m}$ (assumido como $-40\,\text{dBm}$), A escolha de $-40\,\text{dBm}$ como valor de $RSSI_{\text{ref}}$ para uma distância de 1 metro foi baseada em características típicas de dispositivos ESP32 operando em ambientes de espaço livre, sem obstruções significativas. O valor é amplamente

utilizado em diversos cenários que envolvem comunicação de baixa potência, como a tecnologia ESP-NOW, e reflete a qualidade de sinal esperada a curtas distâncias. Em condições ideais, como em um ambiente aberto sem interferências, o $RSSI_{\text{ref}}$ de $-40\,\text{dBm}$ representa a intensidade de sinal para uma distância curta de 1 metro.

- n é o expoente de perda de caminho (variável que depende do ambiente, aqui assumido como 2 para espaço livre),
- d é a distância entre o drone e o receptor,
- d_{ref} é a distância de referência (normalmente 1 metro).

Após esses valores de distância houve um início de dados corrompidos, portanto a distância radial será de aproximadamente 100 metros, sendo um valor razoável, já que o usuário que pilota o drone saberá onde os coletadores de dados estão.

5.4 Infraestrutura da AWS

Quando o ESP32 do drone detecta o sinal do roteador correto, que foi condo anteriormente via *Bluetooth* (com o SSID e a senha da rede enviados de maneira segura), ele se conecta à rede Wi-Fi e estabelece comunicação com o *broker* do serviço AWS IoT Core.

Os dados coletados são enviados para o AWS IoT Core, mas, ao invés de serem enviados diretamente para o S3 (o que aumentaria os custos dependendo da quantidade de dados), eles passam por um *buffer*. Esse *buffer* controla o fluxo de dados e limita a quantidade que é escrita de uma vez no S3, ajudando a otimizar os custos de armazenamento.

Uma vez que os dados estão na AWS, eles podem ser acessados por outros serviços. O primeiro serviço a ser utilizado é uma função *serverless* que roda um código em Python. Esse código formata o JSON recebido via MQTT para um formato que pode ser mais facilmente manipulado. Como o JSON foi tratado inicialmente como uma *string* grande (já que passou por um processo de descriptografia), ele precisa ser corretamente convertido para o formato JSON estruturado.

Depois de formatado, os dados passam por outro serviço da AWS, o *Athena*, que é usado para executar *queries SQL* e transformar os dados em um formato tabular. Esses dados tabulares são então armazenados em outro *bucket* S3, organizando-os de forma mais eficiente.

5.4.1 Interface para observar os dados

O Grafana é uma plataforma de análise e visualização de dados de código aberto que se destaca por sua capacidade de criar *dashboards* interativos e dinâmicos. Esta ferramenta

é amplamente utilizada em ambientes de monitoramento e análise de dados em tempo real, oferecendo uma solução robusta para a visualização de métricas e logs.

5.4.1.1 Implementação no Projeto

Finalmente, é criada uma instância de máquina virtual que executa o Grafana. Esse Grafana é configurado com um *plugin* que se conecta ao *Athena*, permitindo acessar os dados tabulares. Com isso, é possível usar consultas SQL adicionais para visualizar os dados e configurar os painéis gráficos no *front-end*(interface de usuário) do Grafana, facilitando a análise e monitoramento. Os gráficos estão dispostos como na Figura 15, podendo ser facilmente modificados e dispostos de diferentes formas.

A integração do Grafana neste projeto oferece várias vantagens:

- **Visualização Intuitiva**: Permite a criação de *dashboards* intuitivos que facilitam a compreensão dos dados dos sensores.
- **Monitoramento em Tempo Real**: Possibilita o acompanhamento contínuo das métricas coletadas.
- **Flexibilidade na Análise**: Oferece diferentes formas de visualização e análise dos dados, permitindo identificar padrões e anomalias.
- **Compartilhamento de Informações**: Facilita o compartilhamento de *insights* através de *dashboards* interativos.

Figura 15 – Exemplo de como serão mostrados os valores para os diferentes sensores

Fonte: Autoria própria

A Figura 15 demonstra um exemplo da interface implementada, onde os diferentes sensores podem ser monitorados através de gráficos personalizados. Esta disposição pode ser facilmente adaptada conforme as necessidades específicas de análise, oferecendo uma solução versátil para o monitoramento dos dados coletados.

6 Conclusão

Este trabalho apresenta um sistema eficaz para coleta de dados em áreas remotas utilizando drones e microcontroladores ESP32, integrados a sensores IoT e a serviços de nuvem da AWS. A implementação demonstrou ser eficiente em termos de consumo de energia, autonomia e processamento de dados em tempo real, graças à integração de edge computing e algoritmos como o TEDA, que permitiram a detecção de anomalias e a otimização da transmissão de dados.

Os resultados obtidos indicam que, mesmo em ambientes off-grid, o sistema pode funcionar de maneira autônoma por longos períodos, especialmente quando alimentado por fontes de energia renováveis, como placas solares. A análise de consumo de energia e o teste de distância do protocolo ESP-NOW reforçam a viabilidade deste sistema em cenários de monitoramento ambiental e agrícola.

6.1 Perspectivas Futuras

Para futuras melhorias e expansões, propõe-se a integração de um coletor de dados mais robusto, baseado no SoC ESP32, capaz de lidar com volumes maiores de informações. Além disso, o uso de drones com sistemas de tracking pode automatizar ainda mais o processo, permitindo que o drone acompanhe o movimento de objetos ou áreas de interesse, sem a necessidade de intervenção manual.

Combinando essas tecnologias, seria possível desenvolver um sistema completamente automatizado, onde o drone realizaria a coleta de dados de forma contínua e autônoma, retornando à base apenas para recarregar ou transmitir grandes volumes de informações. Isso abriria portas para uma série de novas aplicações, como o monitoramento de áreas florestais, zonas urbanas, e até mesmo o controle de desastres naturais.

A integração de algoritmos de inteligência artificial (IA) e redes de drones cooperativos também pode ser uma linha de pesquisa futura, permitindo que vários drones trabalhem em conjunto para cobrir áreas maiores de forma eficiente. A automação completa do sistema, com o uso de tracking e drones autônomos, pode tornar o monitoramento de grandes áreas mais seguro, eficiente e preciso, trazendo soluções inovadoras para diversas indústrias.

Referências

ALJUMAH, A.; AHANGER, T. A.; ULLAH, I. Heterogeneous blockchain-based secure framework for uav data. *Mathematics*, v. 11, n. 6, 2023. ISSN 2227-7390. Disponível em: <https://www.mdpi.com/2227-7390/11/6/1348>. 22

ALMEIDA, F.; SILVA, R. Edge computing with esp32 for optimized data collection in remote areas. *Sensors*, v. 22, n. 1, p. 102–110, 2022. 15

ALZAHRANI, M. A. et al. Comparative performance study of esp-now, wifi, bluetooth protocols based on range, transmission speed, latency, energy usage and barrier resistance. *2021 IEEE 18th International Conference on Networking, Sensing and Control (ICNSC)*, v. 1, n. 1, p. 1–6, 2021. Disponível em: <https://ieeexplore.ieee.org/document/9573246>. 27

ANGELOV, P. Anomaly detection based on eccentricity analysis. p. 1–8, 2014. 31, 32

AZION. How edge computing solves four key iot challenges. 2021. Disponível em: <https://www.azion.com/en/blog/how-edge-computing-solves-iot-challenges/>. 30

DAEMEN, J.; RIJMEN, V. *The Design of Rijndael: AES - The Advanced Encryption Standard*. [S.l.]: Springer, 2002. 28

DOE, J. Desigualdade da excentricidade na detecção de outliers. *Data Analytics Review*, v. 5, p. 50–60, 2020. 33

GOOGLE. *Google Maps*. 2023. Acesso em: 20 out. 2023. Disponível em: <https://www.google.com/maps>. 54

GOULART, A.; PINTO, A.; BOAVA, A. Iot off-grid, data collection from a machine learning classification using uav. *Sensors*, v. 22, p. 111–122, 2022. 16

HAT, R. What is iot edge computing? 2022. Disponível em: <https://www.redhat.com/en/topics/edge-computing/iot-edge-computing-need-to-work-together>. 30

HOLTEK SEMICONDUCTOR INC. *HT78XX 100mA Low Power LDO*. [S.l.], 2021. Document No. DS78XX-05. 50

IBERDROLA. O que é edge computing, o futuro da conectividade. 2021. Disponível em: <https://www.iberdrola.com/inovacao/o-que-e-edge-computing>. 30

KATZ, J.; LINDELL, Y. *Introduction to Modern Cryptography: Principles and Protocols*. [S.l.]: Chapman and Hall/CRC, 2010. 28, 29

KIM, S.; JOHNSON, R. Edge computing optimization for drone-based iot applications. *IEEE Transactions on Mobile Computing*, v. 23, n. 1, p. 112–127, 2024. Acesso em: 10 out. 2024. Disponível em: <https://ieeexplore.ieee.org/document/9876545>. 18, 22

KOKAN, R. I. *Test ESP-NOW - Comunicação de longo alcance e baixo consumo de energia*. 2021. Faculty of Technical Sciences. Acesso em: 6 out. 2024. Disponível em: <https://www.researchgate.net/publication/369626626_Test_ESP-NOW>. 25, 27

LIANG, H.; YU, W.; NGUYEN, J. Internet of things data collection using unmanned aerial vehicles in infrastructure free environments. *IEEE Access*, v. 8, p. 7650–7660, 2020. 15

MAGZYM, Y. et al. Synchronized esp-now for improved energy efficiency. In: *IEEE International Conference on Smart Information Systems and Technologies*. [s.n.], 2023. Acesso em: 6 out. 2024. Disponível em: <https://orbit.dtu.dk/en/publications/synchronized-esp-now-for-improved-energy-efficiency>. 24, 25, 26

MAKAM, S. et al. Unmanned aerial vehicles (uavs): an adoptable technology for precise and smart farming. *Discover Internet of Things*, v. 4, n. 1, p. 12, 2024. ISSN 2730-7239. Disponível em: <https://doi.org/10.1007/s43926-024-00066-5>. 18, 22

MANFREDA, S. et al. On the use of unmanned aerial systems for environmental monitoring. *Remote Sensing*, v. 10, n. 4, 2018. ISSN 2072-4292. Disponível em: <https://www.mdpi.com/2072-4292/10/4/641>. 18

MOHAMED, N. et al. Unmanned aerial vehicles applications in future smart cities. *Technological Forecasting and Social Change*, v. 153, p. 119293, 2020. ISSN 0040-1625. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0040162517314968>. 18, 22

MOHAMMED, A.; SMITH, T. Iot-based solutions for data collection in remote areas. *International Journal of Smart Technology*, v. 9, n. 2, p. 45–57, 2021. 15

PASIC, R.; KUZMANOV, I.; ATANASOVSKI, K. Esp-now communication protocol with esp32. *Journal of Universal Excellence*, v. 6, n. 1, p. 53–60, 2021. Acesso em: 6 out. 2024. Disponível em: <https://www.researchgate.net/publication/369626626_Indoor_Performance_Evaluation_of_ESP-NOW>. 24, 25, 27

PATEL, V.; ANDERSON, T. Real-time data processing in uav-edge computing systems. *Journal of Network and Computer Applications*, v. 198, p. 103425, 2023. Acesso em: 10 out. 2024. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804523000XXX>. 21

RODRIGUEZ, M.; SMITH, P. Energy-efficient path planning for uav-based data collection in remote areas. In: *International Conference on Robotics and Automation (ICRA)*. [s.n.], 2023. p. 2145–2152. Acesso em: 10 out. 2024. Disponível em: <https://ieeexplore.ieee.org/document/9876544>. 18, 22

SANTOS, B. P. et al. A comparison of wireless communication protocols for iot applications. *IEEE Internet of Things Journal*, v. 10, n. 12, p. 11234–11248, 2023. 27

SANTOS, E.; ALMEIDA, F. Advanced iot applications using drones for data collection. *Sensors Journal*, v. 10, p. 100–115, 2022. 15

SHARMA, J.; MEHRA, P. S. Secure communication in iot-based uav networks: A systematic survey. *Internet of Things*, v. 23, p. 100883, 2023. ISSN 2542-6605. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2542660523002068>. 22

SMITH, M. Understanding typicality and eccentricity in data analysis. *International Journal of Machine Learning*, v. 15, p. 200–210, 2021. 31, 32

SONG, H.; ZHANG, Y. Practical security analysis of the pkcs7 padding scheme. *Proceedings of the 2012 International Conference on Security and Cryptography (SECRYPT)*, p. 1–6, 2012. 29

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. [S.l.]: Pearson Education, 2006. 27, 29

SYSTEMS, E. *ESP32 Series Datasheet*. 2021. Acesso em: 15 out. 2024. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. 50

ZHANG, W.; LIU, F. Data collection in remote areas: Challenges and solutions. *Journal of Remote Sensing*, v. 12, p. 123–135, 2020. 15