Universidade Federal do Rio Grande do Norte
Departamento de Engenharia de Computação e Automação

Aluno: Allyson Matheus Guedes de Oliveira
Matrícula: 20240001251

Trabalho individual - Projeto de Sistema Embarcado com Rede, MQTT, e NTP

MONITORAMENTO DE ESTUFA INTELIGENTE COM SENSOR DIGITAL

Natal, 2025

1. INTRODUÇÃO E PROBLEMÁTICA

A agricultura moderna e a horticultura de precisão dependem cada vez mais de ambientes controlados, como estufas, para maximizar a produção e cultivar espécies fora de suas épocas ou climas naturais. Dentro desses ambientes, fatores como temperatura, umidade e, crucialmente, a luminosidade, devem ser mantidos dentro de faixas ideais para garantir a saúde das plantas e a eficiência da fotossíntese. Este projeto foca no desenvolvimento de uma solução de monitoramento para otimizar esse controle.

O problema central abordado é a ineficiência e a suscetibilidade a erros do monitoramento manual. Em muitas operações, especialmente em pequena escala, o controle das condições da estufa depende da verificação humana, o que é trabalhoso e inconsistente. Flutuações na luminosidade, por exemplo, podem levar a um gasto energético desnecessário com iluminação artificial ou, inversamente, a períodos de luz insuficiente que prejudicam o crescimento e podem danificar plantas mais sensíveis.

Esta solução é destinada a um público que necessita de monitoramento preciso sem os altos custos de sistemas industriais complexos, como pequenos produtores, entusiastas de horticultura e laboratórios de botânica. Para estes grupos, o desperdício de energia com iluminação artificial, que muitas vezes opera em potência máxima mesmo quando a luz solar natural é adequada, representa um custo operacional significativo que pode ser mitigado com um monitoramento mais inteligente.

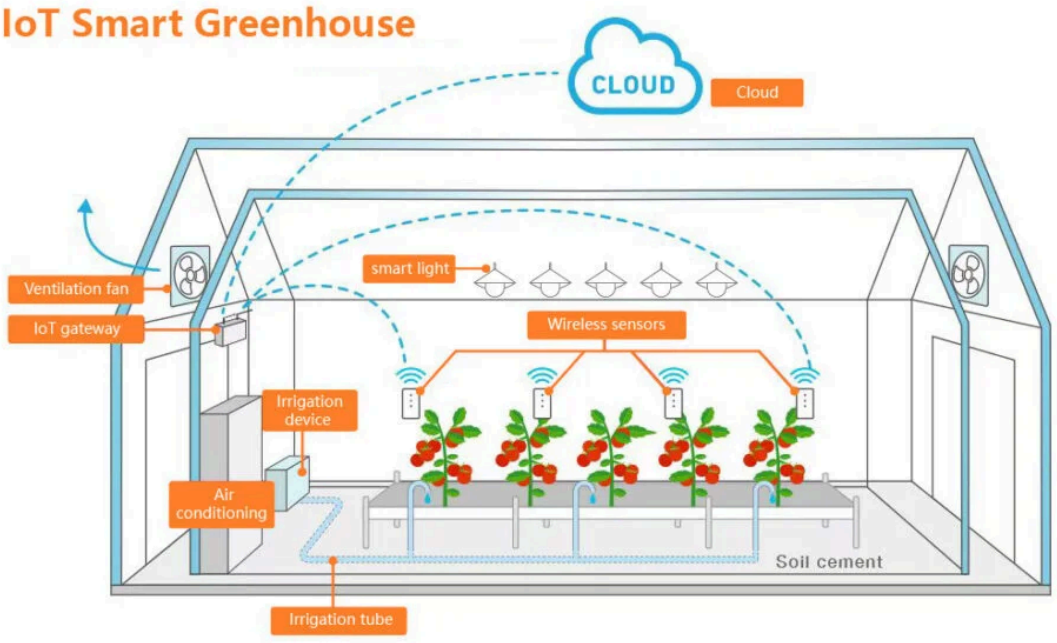

Figura 1: Exemplo de monitoramento de estufa inteligente (Disponível em https://www.dusuniot.com/blog/aiot-artificial-intelligence-of-things/)

Para solucionar esta problemática, este trabalho propõe o projeto e a simulação de um sistema embarcado de baixo custo conectado à Internet das Coisas (IoT). O dispositivo utilizará um sensor analógico de luminosidade (LDR)

para coletar dados do ambiente em tempo real. As leituras serão exibidas localmente em um display para verificação imediata, permitindo que o operador tenha feedback instantâneo das condições de iluminação.

A maior importância deste projeto reside na sua capacidade de registro e monitoramento remoto. Além da exibição local, o sistema foi projetado para se conectar a uma rede, sincronizar seu relógio com um servidor NTP para garantir a precisão temporal dos dados , e publicar as leituras de luminosidade em um broker MQTT. Isso não só cria um registro histórico confiável para análise, mas também estabelece a fundação para uma futura automação, onde sistemas de iluminação ou sombreamento possam ser acionados com base nos dados coletados, otimizando o crescimento das plantas e economizando recursos.

2. DESCRIÇÃO DO SISTEMA

O sistema de monitoramento de estufa foi projetado como um dispositivo IoT (Internet das Coisas) compacto e eficiente, capaz de operar de forma autônoma para coletar e transmitir dados ambientais. A plataforma de simulação escolhida foi o Wokwi, devido ao seu suporte robusto para simulação de rede com microcontroladores avançados. A programação foi desenvolvida inteiramente em linguagem C, utilizando o framework oficial ESP-IDF, para cumprir o requisito de não utilização das funções do Arduino. A seguir estão listados os componentes do sistema escolhido na configuração deste projeto. A Figura 2 a seguir ilustra o sistema completo simulado na plataforma Wokwi.

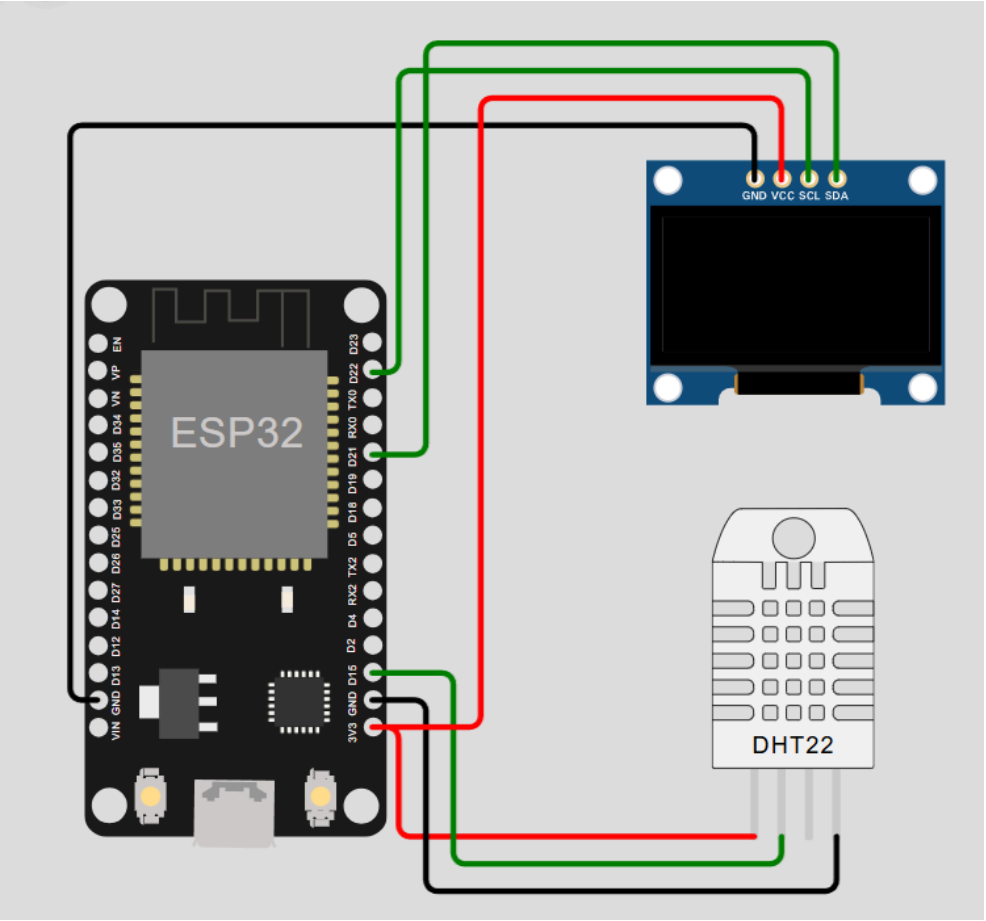

Figura 2: Sistema de monitoramento de estufa completo simulado (Autor, 2025).

2.1 Microcontrolador ESP32-WROOM-32

Este microcontrolador foi selecionado por ser uma solução completa e de baixo custo que integra um processador dual-core Xtensa LX6 e, o mais importante, conectividade Wi-Fi e Bluetooth nativa. A presença de Wi-Fi integrado elimina a necessidade de módulos de rede externos (como o W5500), simplificando o design do hardware e permitindo a comunicação direta com os serviços de NTP e MQTT usando bibliotecas C do ESP-IDF.

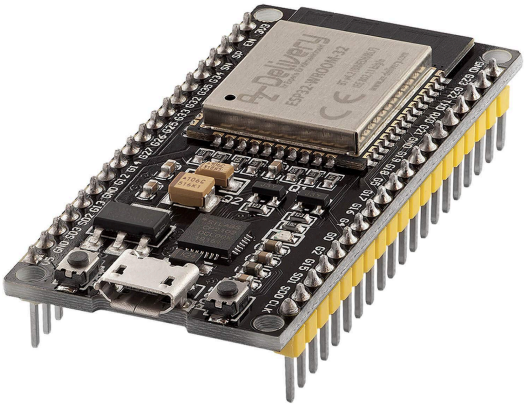

Figura 3: ESP32-WROOM-32 (Disponível em https://www.aranacorp.com/pt/programacao-de-um-nodemcu-esp32-com-a-ide-arduino/amp/)

2.2 Sensor DHT22

Este sensor é o do tipo sensor digital de dados compostos. Sua função é medir a temperatura e a umidade relativa do ar. A faixa de operação para umidade varia de 0% a 99,9% RH, enquanto a temperatura varia de -40°C a 80°C. O pino de conexão é conectado a um pino GPIO digital do ESP32. A comunicação é feita por um protocolo digital de fio único.

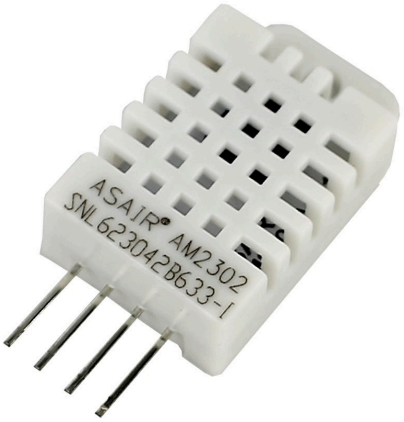

Figura 4: Sensor DHT22 (Disponível em https://l1nq.com/sensordht22).

2.3 Display OLED SSD1306

Foi escolhido um display OLED pela fácil leitura visual na tela, baixo consumo de energia e interface de comunicação simples, utilizando o protocolo I2C e resolução de 128x64 pixels. Esta interface utiliza apenas dois pinos de dados (SDA e SCL), economizando pinos GPIO do microcontrolador para outras funções. No ESP32, foi conectado aos pinos I2C padrão.

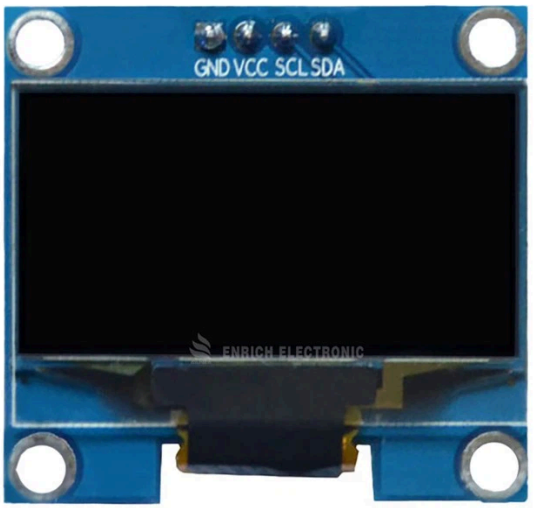

Figura 5: Display OLED SSD1306 (Disponível em https://l1nq.com/displayoled1306).

2.4 Serviços de Rede e Protocolos

A plataforma de simulação escolhida para este projeto foi a Wokwi (wokwi.com). Através do Broker MQTT foi possível utilizar o endereço público test.mosquitto.org. O MQTT é o protocolo central para a publicação dos dados, escolhido por sua leveza, o que o torna ideal para dispositivos embarcados com restrições de processamento e banda.

Para o servidor NTP foi utilizado o pool público pool.ntp.org. A sincronização de horário via NTP (Network Time Protocol) é um requisito do projeto, garantindo que todas as amostras de dados coletadas tenham um carimbo de data/hora preciso e universal.

2.5 Lógica de Coleta

O sistema está configurado para operar com um intervalo de amostragem de 1 segundo, realizando a leitura do sensor DHT22 com essa frequência. As leituras coletadas são então agrupadas em uma fila (buffer) com capacidade para 10 amostras. Desse modo, a cada 10 segundos (resultado de 10 amostras a cada 1 segundo por amostra), o sistema publica o grupo completo de dados no broker MQTT e reinicia o ciclo de coleta e armazenamento.

3. COMUNICAÇÃO E REDE

A conectividade é um pilar central deste projeto, transformando-o de um simples medidor local em um verdadeiro dispositivo IoT. Esta seção detalha os

protocolos e métodos utilizados para conectar o microcontrolador ESP32 à internet, permitindo a sincronização de horário e a publicação de dados.

3.1 O Protocolo MQTT e Importância em IoT

O MQTT (Message Queuing Telemetry Transport) é um protocolo de mensagens leve, projetado especificamente para a comunicação máquina-a-máquina (M2M) em redes com restrições, sendo hoje considerado um padrão de fato para aplicações de Internet das Coisas (IoT). Seu funcionamento é baseado em um modelo de publicar/subscrever. Diferente do modelo de requisição/resposta, como o HTTP, no MQTT os dispositivos (clientes) não se comunicam diretamente entre si, mas sim através de um servidor central chamado Broker. Um dispositivo, como o sensor deste projeto, publica uma mensagem para um tópico específico no broker, e qualquer outro dispositivo que tenha assinado este tópico recebe a mensagem do broker instantaneamente.

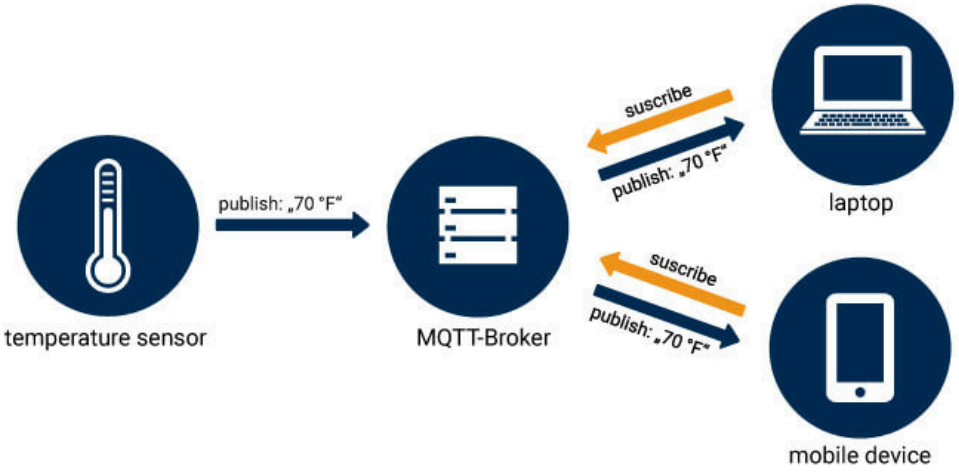

Figura 6: Ilustração do caminho do broker MQTT integrado ao sistema (Disponível em https://www.gta.ufrj.br/ensino/eel878/redes1-2023-1/trabalhos/Grupo01/).

A importância do MQTT em aplicações IoT deve-se a três fatores principais: leveza, desacoplamento e confiabilidade. Primeiramente, o cabeçalho de suas mensagens é extremamente pequeno, o que consome o mínimo de banda de rede e poder de processamento, sendo ideal para microcontroladores como o ESP32. Em segundo lugar, o protocolo promove o desacoplamento, significando que o publicador e o assinante não precisam se conhecer; um sensor pode publicar dados sem saber se há alguém (um aplicativo ou banco de dados) ouvindo. Por fim, o MQTT possui mecanismos de Qualidade de Serviço (QoS) que podem garantir a entrega de mensagens, mesmo em redes instáveis. Neste projeto, o ESP32 atua como um cliente MQTT que publica um pacote de 10 amostras no broker público test.mosquitto.org.

3.2 Sincronização de Horário via NTP

O NTP (Network Time Protocol) é um protocolo padrão da Internet usado para sincronizar os relógios de computadores e dispositivos com uma fonte de tempo universal precisa. O processo de sincronização ocorre em um modelo

cliente-servidor, onde o microcontrolador (o cliente) envia uma solicitação de rede para um servidor NTP, como o pool.ntp.org. O servidor, por sua vez, responde com o horário atual, geralmente no formato UTC (Tempo Universal Coordenado). O cliente NTP em execução no ESP32 é capaz de calcular o atraso da rede (latência) e ajustar seu relógio interno (RTC - Real-Time Clock) para corresponder com precisão ao horário recebido.

A importância da sincronização NTP neste projeto é crucial. Como o trabalho exige que as leituras sejam coletadas e publicadas, o dispositivo precisa saber "que horas são" para que os dados tenham valor contextual. O NTP fornece a capacidade de associar um carimbo de data/hora (timestamp) preciso a cada amostra de dado coletada. Isso permite que qualquer aplicação ou usuário que receba os dados saiba exatamente *quando* aquela temperatura ou umidade foi registrada na estufa, o que é fundamental para a análise de tendências e o registro histórico.

3.3 Conexão de Rede do Microcontrolador

O microcontrolador escolhido, o ESP32, é o componente central da conectividade do sistema. Diferente de microcontroladores mais simples (como o Atmega328), o ESP32 possui um módulo Wi-Fi integrado (IEEE 802.11 b/g/n).

O processo de conexão é gerenciado pelo framework ESP-IDF (em linguagem C). O código instrui o ESP32 a operar em modo "Station" (STA), escanear as redes disponíveis e se conectar à rede Wi-Fi (neste caso, a rede simulada "Wokwi-GUEST"). Após uma conexão bem-sucedida e a obtenção de um endereço IP via DHCP, o ESP32 ganha acesso à pilha de protocolos TCP/IP, permitindo que ele estabeleça conexões com o servidor NTP para sincronizar o relógio e com o broker MQTT para publicar os dados dos sensores.

3.4 Diagrama de Blocos

O diagrama ilustra a arquitetura do hardware e a interconexão dos protocolos de rede. A Figura 3 abaixo ilustra a representação gráfica do diagrama de blocos.

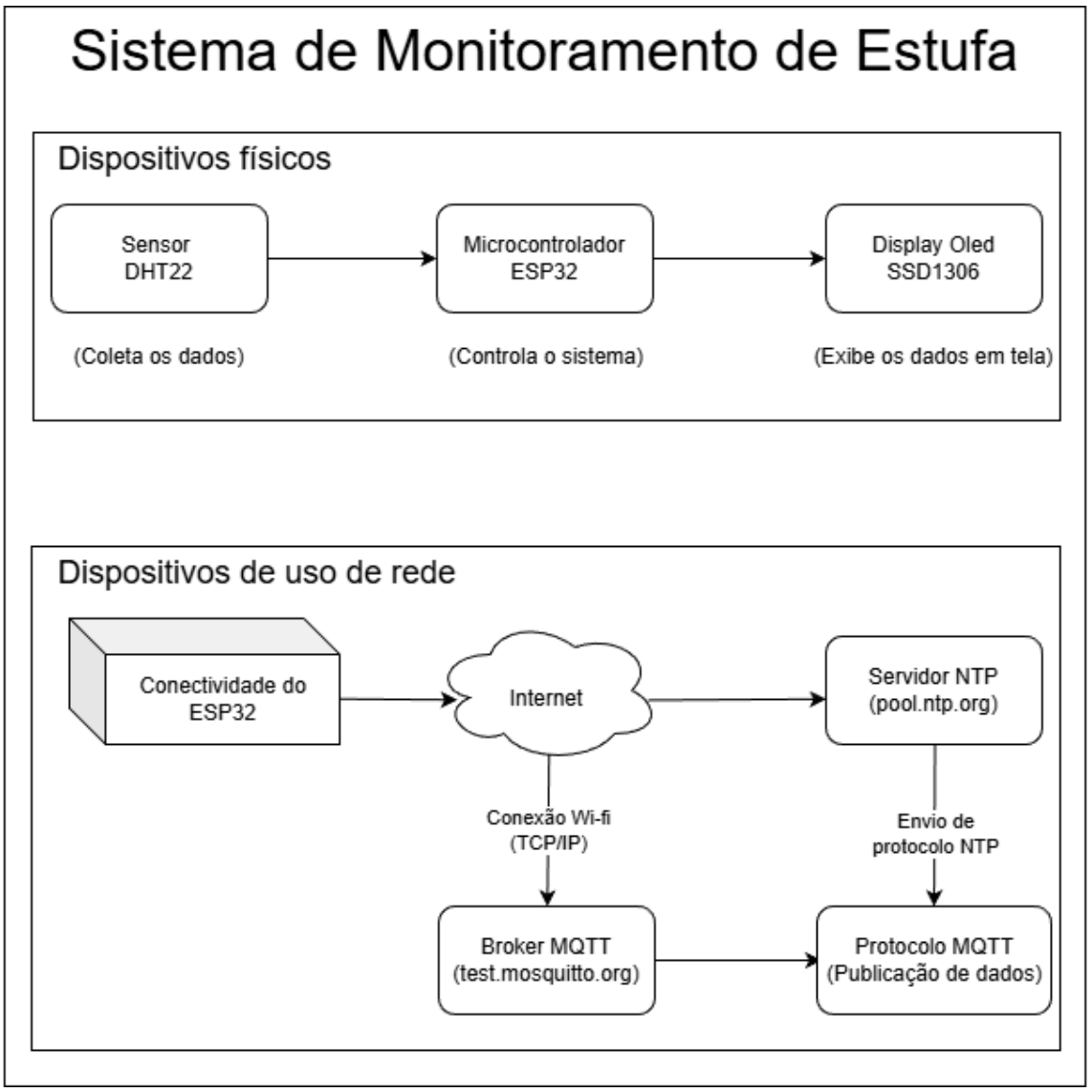

Figura 7: Diagrama de blocos do sistema (Autor, 2025).

O componente central do sistema é o microcontrolador ESP32, que funciona como o "cérebro" de toda a operação. Ele é o ponto de conexão para todos os outros blocos, sendo responsável por processar os dados recebidos, gerenciar as conexões e executar a lógica principal do sistema.

Para a coleta de informações, o sistema utiliza um bloco de entrada composto pelo sensor DHT22, que monitora a temperatura e a umidade do ambiente da estufa. Esses dados são enviados ao ESP32 através de uma conexão unidirecional (Dados Digitais). No que diz respeito à saída, o ESP32 envia os dados já formatados para um display OLED SSD1306. Esta comunicação também é unidirecional e utiliza comandos I2C para exibir as informações ao usuário.

A conectividade é gerenciada pelo módulo Wi-Fi integrado ao próprio ESP32. Este bloco de rede estabelece a conexão física (via TCP/IP) com a rede externa, representada como um bloco maior ou nuvem de "Internet / Rede Simulada". É através dessa conexão com a internet que o sistema acessa os serviços remotos necessários para seu funcionamento.

O sistema utiliza dois serviços remotos principais acessados pela Internet. O primeiro é um Servidor NTP (pool.ntp.org), com o qual estabelece uma comunicação

bidirecional via protocolo NTP para solicitar e receber a hora exata, mantendo o sistema sincronizado. O segundo serviço é um Broker MQTT (test.mosquitto.org), para onde o sistema publica os dados coletados da estufa. Neste caso, a conexão é unidirecional, enviando os dados para o broker remoto através do protocolo MQTT.

4. IMPLEMENTAÇÃO

Esta seção detalha a arquitetura de software, a lógica de controle principal e as bibliotecas utilizadas para implementar os requisitos do projeto em linguagem C, utilizando o framework ESP-IDF para o microcontrolador ESP32.

4.1 Pseudocódigo da Lógica Principal

O sistema é baseado em um loop principal contínuo que gerencia a coleta, a fila e a publicação dos dados. O pseudocódigo a seguir descreve o fluxo de operação do firmware:

```
// 1. Definições Globais
Definir Tamanho_Fila = 10
Criar Fila_de_Amostras[Tamanho_Fila]
Definir Contador_Amostras = 0

INÍCIO_PROGRAMA (app_main)

  // 2. Inicialização dos Periféricos
  Inicializar_Hardware_I2C(pino_SDA, pino_SCL)
  Inicializar_Display_OLED()
  Inicializar_Sensor_DHT22(pino_GPIO)

  // 3. Inicialização da Rede
  Conectar_WiFi("Wokwi-GUEST", "")
  Sincronizar_Relógio_NTP("pool.ntp.org")
  Conectar_Broker_MQTT("test.mosquitto.org")

  // 4. Loop Principal
  ENQUANTO (Verdadeiro)

    // 4.1 Coleta de Dados
    // Faz uma leitura a cada 1 segundo
    Esperar(1000 milissegundos)

    (Temp, Humid) = Ler_Sensor_DHT22()
    Timestamp_Atual = Obter_Horario_Atual()

    // 4.2 Atualização do Display
```

```
    // Mostra a última amostra coletada
    Limpar_Display()
    Escrever_no_Display(0, 0, "Estufa IoT - Online")
    Escrever_no_Display(0, 2, "Temp: " + Temp + " C")
    Escrever_no_Display(0, 3, "Humid: " + Humid + " %")

    // 4.3 Gerenciamento da Fila
    Amostra_Atual = Criar_Pacote(Temp, Humid, Timestamp_Atual)
    Adicionar_na_Fila(Fila_de_Amostras, Amostra_Atual,
Contador_Amostras)
    Contador_Amostras = Contador_Amostras + 1

    // 4.4 Verificação e Publicação
    // Verifica se a fila atingiu 10 amostras
    SE (Contador_Amostras == Tamanho_Fila) ENTÃO

      // Converte a fila de 10 amostras em uma única string (ex:
JSON)
      Mensagem_Completa = Formatar_Fila_Para_MQTT(Fila_de_Amostras)

      // Publica o grupo de 10 dados
      Publicar_MQTT("ufrn/dca3706/estufa/dados", Mensagem_Completa)

      // Reinicia o ciclo
      Contador_Amostras = 0

    FIM_SE

  FIM_ENQUANTO
FIM_PROGRAMA
```

4.2 Descrição da Fila de Dados e Loop Principal

O firmware opera em um loop principal infinito (while(1)), que constitui o ciclo de vida da aplicação após a inicialização dos periféricos e da rede. A lógica principal é regida por um temporizador (no ESP-IDF, implementado com vTaskDelay do FreeRTOS) que estabelece o intervalo de amostragem de 1 segundo.

A cada segundo, o sistema executa três ações: coleta, exibição e enfileiramento. Primeiro, ele lê os dados de temperatura e umidade do sensor DHT22. Imediatamente após, ele atualiza o display OLED com esta leitura mais recente, garantindo que o usuário tenha sempre a informação mais atualizada.

Para atender ao requisito de publicação em grupo, foi implementada uma fila de dados, estruturada como um array de 10 posições. Cada nova leitura (contendo temperatura, umidade e o timestamp NTP) é armazenada neste array. Um contador rastreia o número de amostras armazenadas. Quando o contador atinge 10, o

sistema pausa o enfileiramento, formata todo o array de 10 amostras (preferencialmente em formato JSON) em uma única string, e publica esta string no broker MQTT. Após a publicação, o contador é zerado, e o ciclo de coleta e enfileiramento recomeça continuamente.

4.3 Bibliotecas Utilizadas e Funções Principais

Para a implementação em linguagem C no ESP32 (na configuração ESP-IDF), o projeto utiliza componentes de driver do próprio framework, além de bibliotecas de terceiros para protocolos específicos. O requisito de não usar funções do Arduino foi estritamente seguido. Nas subseções a seguir estarão listas das bibliotecas utilizadas para controlar o ESP, bibliotecas adicionais incluídas e as principais funções implementadas, respectivamente.

4.3.1 Bibliotecas do microcontrolador ESP (drivers de hardware e rede)

driver/i2c.h: Biblioteca de C para controle do barramento I2C em nível de hardware, usada para a comunicação com o display OLED.

driver/gpio.h: Utilizada para configurar o pino de dados do sensor DHT22.

esp_wifi.h: Componente central do ESP-IDF para inicialização, configuração e conexão com a rede Wi-Fi.

esp_sntp.h: Biblioteca cliente para o protocolo NTP. Sua função principal, sntp_init(), é chamada para sincronizar o relógio interno (RTC) do ESP32 com um servidor de tempo (pool.ntp.org).

mqtt_client.h: Componente cliente para o protocolo MQTT. Ele gerencia a conexão com o broker (test.mosquitto.org), as reconexões automáticas e a publicação de mensagens.

freertos/task.h: Parte do sistema operacional (FreeRTOS) base do ESP-IDF. A função vTaskDelay(pdMS_TO_TICKS(1000)) é a função principal que implementa o intervalo de 1 segundo sem travar o processador.

4.3.2 Bibliotecas Adicionais

ssd1306.h: Biblioteca C para controle do display OLED, responsável por enviar os comandos de inicialização e os dados de pixel (texto) via I2C.

dht.h: Biblioteca C para controle do sensor DHT22, responsável por gerenciar a temporização precisa do protocolo de fio único para ler a temperatura e a umidade.

4.3.3 Funções Principais da Lógica

app_main(): Ponto de entrada principal do programa.

wifi_init_sta(): Função criada para encapsular a lógica de inicialização do Wi-Fi.

obter_horario_atual(): Função que utiliza o time.h (após a sincronização NTP) para buscar o timestamp atual.

mqtt_event_handler(): Função de *callback* que lida com eventos da conexão MQTT (ex: "conectado", "desconectado").

formatar_fila_json(): Função criada para percorrer o array de 10 amostras e construir a string JSON a ser publicada.

5. CUSTO ESTIMADO

Para avaliar a viabilidade do protótipo, foi realizado um levantamento do custo médio dos componentes de hardware em fornecedores nacionais. Os valores são estimativas e podem variar conforme o fornecedor e a data da cotação. A Tabela 1 abaixo detalha o custo de cada item necessário para a montagem do sistema de monitoramento.

Componente	Função	Preço Médio (R$)
Placa de Desenvolvimento ESP32	Microcontrolador e Módulo de Rede	R$ 55,00
Sensor DHT22	Sensor de Temperatura e Umidade	R$ 35,00
Display OLED SSD1306 (0.96" I2C)	Interface de exibição de dados	R$ 28,00
Jumpers (Macho-Fêmea / Macho-Macho)	Conexão dos componentes	R$ 15,00

Tabela 1: Custo estimado dos componentes do projeto

6. CONCLUSÕES

O desenvolvimento um protótipo simulado para o monitoramento de uma estufa inteligente, atendendo a todos os requisitos funcionais propostos foi bem-sucedido. O sistema demonstrou a capacidade de coletar dados de temperatura e umidade (via sensor DHT22) , exibi-los em tempo real (via display oled), sincronizar o relógio do sistema (via servidor NTP) e publicar grupos de 10 amostras para um broker MQTT remoto. A arquitetura baseada no microcontrolador ESP32 provou ser uma escolha robusta e eficiente, integrando o processamento de dados e a conectividade de rede (Wi-Fi) em um único componente.

O principal desafio encontrado durante a implementação foi a restrição de programar em linguagem C pura, sem o uso das funções de abstração do framework Arduino. A configuração do ESP32 usando o framework oficial ESP-IDF, embora mais complexa, permitiu um entendimento mais profundo do controle do hardware, da pilha de rede TCP/IP e do sistema operacional de tempo real (FreeRTOS) que serve de base para o ESP32. A implementação dos clientes NTP e MQTT, bem como o controle dos periféricos (I2C para o OLED e o protocolo de fio único do DHT22), exigiu um esforço significativamente maior na configuração manual dos drivers e no gerenciamento de *tasks* e *eventos*, em contraste com as bibliotecas simplificadas do Arduino.

Como possíveis melhorias, o projeto pode evoluir de um sistema de monitoramento passivo para um sistema de controle ativo. A adição de atuadores

(como relés) permitiria ao ESP32 controlar automaticamente a ventilação (baseado na temperatura) ou a iluminação artificial (baseado em um sensor de luminosidade e no horário NTP). Além disso, o sistema poderia implementar a funcionalidade de "subscribe" do MQTT, permitindo receber comandos remotamente. Por fim, a próxima etapa natural seria a montagem do protótipo físico para validação do seu funcionamento em um ambiente real de estufa.

7. ANEXOS

Segue o link do repositório do projeto no github com o desenvolvimento do projeto e os arquivos de bibliotecas mencionadas neste documento.

https://github.com/allysonliveira/projeto-de-sistemas-embarcados-iot