**Documentação Técnica – Estação Ciberfísica de Monitoramento Urbano**

Engenharia de Software – Fundamentos de Sistemas Ciberfísicos

Documentação Técnica – Estação Ciberfísica de Monitoramento Urbano

**Grupo:** Dominic, e os Torettos

**Data:** Outubro de 2025

**1. Sumário Executivo**

O projeto "Estação Ciberfísica de Monitoramento Urbano" é um protótipo de Sistema Ciberfísico (CPS) baseado em ESP32, que integra medições ambientais locais (temperatura, umidade e qualidade do ar simulada) com serviços de nuvem (NTP e Open-Meteo)2. Sua principal funcionalidade é o sistema de alerta proativo, que utiliza a API do Telegram para notificar o usuário remotamente sobre condições adversas de clima.

**2. Introdução e Visão Geral do Projeto**

| **Seção** | **Conteúdo** |
| --- | --- |
| **Nome do Projeto** | Estação Ciberfísica de Monitoramento Urbano (Smart City) |
| **Objetivo** | Desenvolver um protótipo de Sistema Ciberfísico (CPS) que integra monitoramento ambiental local com alertas remotos e dados de previsão de nuvem. |
| **Linguagem** | MicroPython |
| **Microcontrolador** | ESP32 (Wokwi/Simulação) |

**3. Especificação de Hardware e Pinagem**

**3.1. Descrição dos Componentes**

| **Componente** | **Tipo** | **Função no Projeto** |
| --- | --- | --- |
| **Microcontrolador** | Processamento | Unidade Central de Processamento (UCP), responsável pela execução do *firmware*, aquisição de dados, controle de atuadores e comunicação via protocolo TCP/IP. |
| **Sensor DHT22** | Sensor (Entrada) | Medição de Temperatura e Umidade. |
| **Potenciômetro** | Sensor (Entrada) | Simula a leitura de um sensor analógico de Qualidade do Ar (AQI). |
| **Display SSD1306** | Atuador (Saída) | Interface de usuário (UI) para exibição de dados e relógio. |
| **LED RGB** | Atuador (Saída) | Indicador visual para o status da Qualidade do Ar (Verde=Boa, Amarelo=Média, Vermelho=Ruim). |

**3.2. Mapeamento de Pinos (Pinout)**

| **Componente** | **Tipo de Conexão** | **Pino do ESP32** |
| --- | --- | --- |
| **Sensor DHT22 (DATA)** | Digital | D13 |
| **Potenciômetro** | Analógica (ADC) | D34 |
| **Display OLED (SCL)** | I2C | D22 |
| **Display OLED (SDA)** | I2C | D21 |
| **LED RGB (R)** | Digital OUT | D25 |
| **LED RGB (G)** | Digital OUT | D26 |
| **LED RGB (B)** | Digital OUT | D27 |

**4. Arquitetura de Software e Fluxo de Dados** 22

| **Elemento** | **Descrição da Funcionalidade** | **Implementação no Código** |
| --- | --- | --- |
| **Conectividade** | Gerencia a conexão Wi-Fi e a sincronização do relógio (NTP). | *network.WLAN, ntptime.settime(), gc.collect()* |
| **Alertas Telegram** | Função que envia notificações push para o usuário via API do Telegram. | Função: *enviar\_telegram()* |
| **Lógica de QA** | Lê o valor analógico, define a cor do LED RGB e o status no display. | Função: *mostra\_tela\_sensores()* |
| **API de Clima** | Faz requisição à API Open-Meteo, traduz o código do clima (WEATHER\_CODES) e aciona o alerta de chuva. | Função: *get\_weather\_forecast()* |
| **Gerenciamento de Memória** | A rotina *gc.collect()* é estrategicamente invocada antes das requisições de rede para liberar blocos de memória não utilizados, mitigando falhas por estouro de memória (OOM). | *gc.collect()* |
| **Ciclo de Telas** | Gerencia a exibição rotativa de 3 telas no OLED (relogio, sensores, previsão). | *Loop While* |

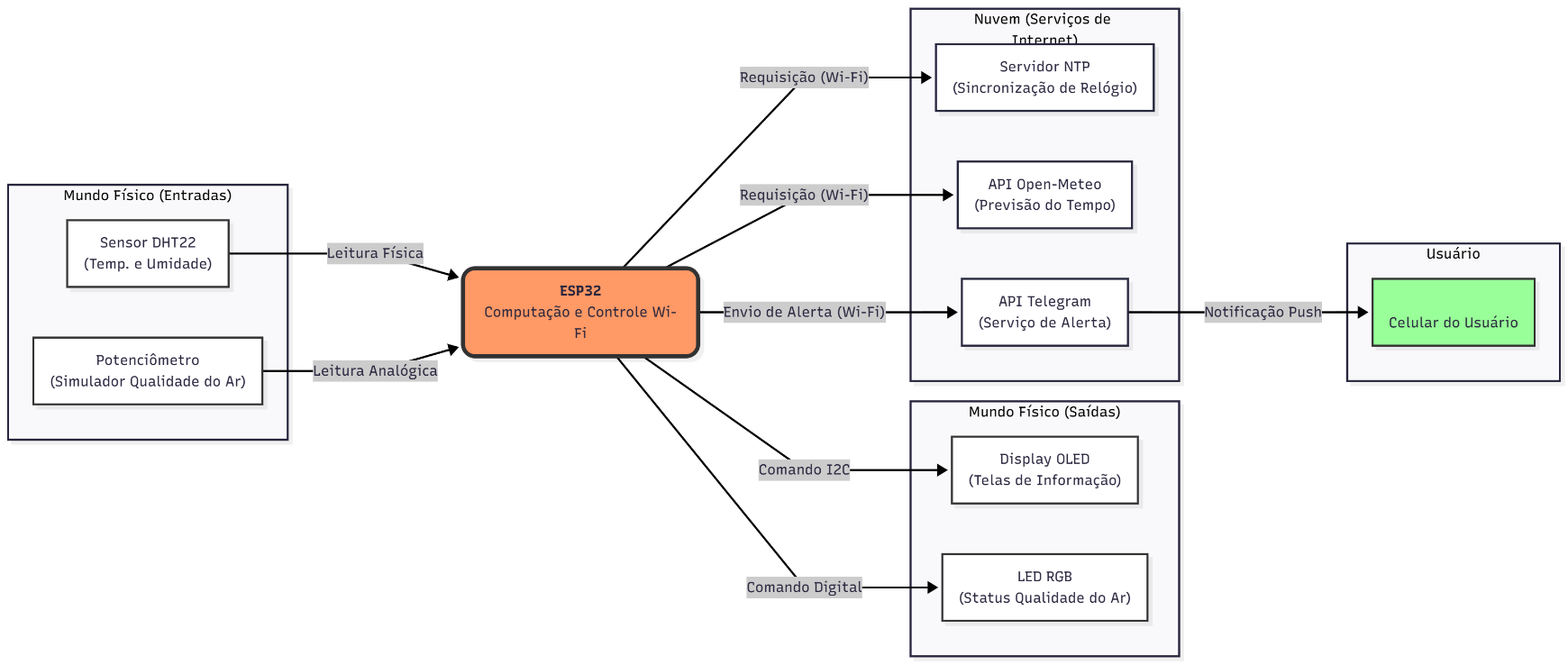
**5. Protocolos de Comunicação** 35

| **Protocolo** | **Camada/Tipo** | **Uso no Projeto** | **Detalhes** |
| --- | --- | --- | --- |
| **Wi-Fi (WLAN)** | Rede | Conectividade principal do ESP32 à Internet. | Conexão STA\_IF |
| **I2C** | Hardware | Comunicação serial entre o ESP32 e o Display OLED. | Pinos SDA (D21) e SCL (D22) |
| **NTP** | Aplicação | Sincronização do relógio (tempo) via internet. | Função *ntptime.settime()* |
| **HTTP(S)** | Aplicação | Usado para consumir dados (GET) e publicar alertas (GET) em serviços de nuvem. | Biblioteca *urequests* |

**6. Lógica de Alerta e Controle de Spammer**

| **Funcionalidade** | **Descrição do Funcionamento** | **Parâmetros de Controle** |
| --- | --- | --- |
| **Alerta de Chuva** | O sistema envia uma notificação se a descrição do clima da API incluir "Chuva", "Garoa", "Pancadas", ou "Trovejadas". | Condição no *get\_weather\_forecast()* |
| **Anti-Spam** | Utilização da *flag* booleana alerta\_enviado para evitar que o bot envie notificações repetidas enquanto a condição de chuva persistir. | A flag é True após o envio do alerta e resetada para False quando o tempo está "Ceu limpo" ou "Com nuvens". 48 |
| **Status da Qualidade do Ar** | Define o status de acordo com o valor lido do potenciômetro (0-4095, mapeado de 0-1023). | **Boa:** < 300 / **Moderada:** 300-700 / **Ruim:** > 700 |  |

**7. Diagrama Funcional**

O **Diagrama Funcional** (representado na imagem enviada) 51ilustra a arquitetura Ciberfísica completa, detalhando o fluxo de dados dos sensores (Entradas Físicas) para o **ESP32**, a comunicação com a **Nuvem** (NTP, Open-Meteo, Telegram) e o acionamento dos **Atuadores** (Display OLED e LED RGB).