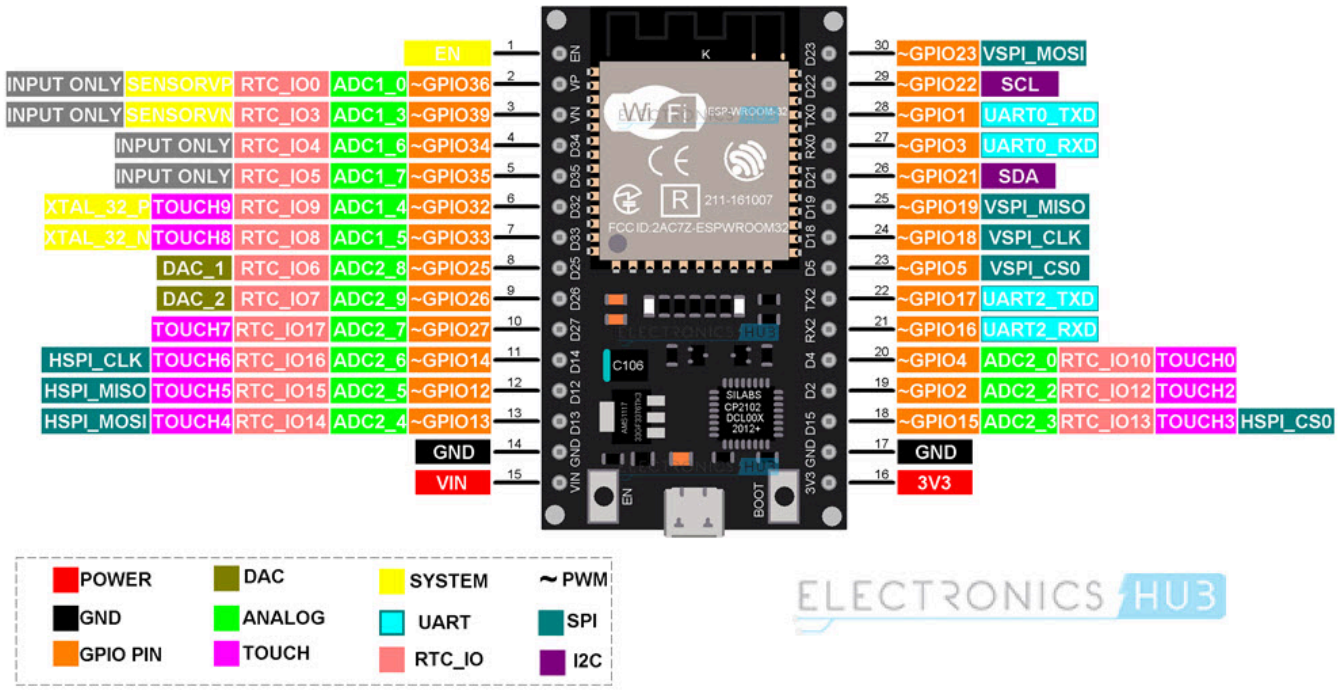
Projeto Monitoramento

ESP32 utilizado



https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf

Node-RED - https://nodered.org/

Node-RED é uma ferramenta de desenvolvimento baseada em fluxo para a integração de hardware, APIs e serviços online. Criada pela IBM, é especialmente popular no campo da Internet das Coisas (IoT), mas também é amplamente utilizada para automatização de processos e integração de sistemas em diversas áreas. Hoje é open Source, mantido pela JsFoundation.

Características Principais

1. Editor de Fluxo Baseado na Web:
    - Utiliza uma interface gráfica intuitiva baseada na web, onde os desenvolvedores podem arrastar e soltar "nós" (blocos de funcionalidade) para criar fluxos de trabalho.
2. Nós (Nodes):
    - Cada "nó" realiza uma função específica, como acessar uma API, processar dados ou controlar um dispositivo IoT.
    - Os nós são conectados por "links", formando fluxos que definem o caminho dos dados através do sistema.

3. Desenvolvimento Rápido:
    - Permite uma prototipagem rápida e um desenvolvimento ágil, pois muitos nós são pré-configurados para tarefas comuns.

4. Extensibilidade
    - Node-RED possui uma grande biblioteca de nós disponíveis para download, cobrindo uma ampla gama de funcionalidades.
    - É possível criar nós personalizados utilizando JavaScript.

5. Baseado em Node.js:
    - Construído sobre Node.js, o que permite uma integração fácil com um ecossistema vasto de módulos JavaScript.
    - Pode ser executado em qualquer lugar onde Node.js é suportado, como em servidores, desktops e dispositivos de borda (edge devices).

Indicação de referência: https://www.youtube.com/watch?v=y671EpUFM5s
Bibliotecas: https://flows.nodered.org/node/node-red-dashboard

ESP-IDF: Conceitos utilizados no projeto.

ADC

O ESP32-Wroom-32D possui dois conversores analógico-digital (ADC) de 12 bits: ADC1 e ADC2, que são utilizados para ler sinais analógicos.

- ADC1 - Possui 8 canais (GPIOs 32 a 39), não é compartilhado com o Wi-Fi, o que significa que pode ser usado simultaneamente com a funcionalidade Wi-Fi sem interferências.
- ADC2 - Possui 10 canais (GPIOs 0, 2, 4, 12 a 15, 25 a 27), é compartilhado com o Wi-Fi, isso significa que, quando o Wi-Fi está ativo, o ADC2 pode não estar disponível ou pode apresentar interferências. Portanto, para medições analógicas confiáveis enquanto o Wi-Fi está ativo, é recomendado usar o ADC1.

modo adc_one_shot

O adc_one_shot refere-se a uma operação onde uma leitura única de uma entrada analógica é realizada. Esta funcionalidade é útil para medições instantâneas em aplicações onde a leitura contínua não é necessária.

FREERTOS

FreeRTOS é um sistema operacional em tempo real (RTOS - Real-Time Operating System) para microcontroladores que oferece suporte a multitarefas, o que significa que você pode executar várias tarefas aparentemente ao mesmo tempo. O ESP32 é um microcontrolador popular que utiliza o FreeRTOS como seu sistema operacional padrão. O FreeRTOS permite a criação, gerenciamento e sincronização de múltiplas tarefas de maneira eficiente.

Principais Conceitos do FreeRTOS

- Task (Tarefa): Unidade básica de execução no FreeRTOS. Cada tarefa é uma função que executa em seu próprio contexto e pode ser executada de forma independente das outras.
- Scheduler (Escalonador): Responsável por gerenciar a execução das tarefas com base em suas prioridades.
- Queue (Fila): Mecanismo para comunicação entre tarefas.
- Semaphore (Semáforo): Ferramenta de sincronização entre tarefas.
- Mutex: Semáforo especial usado para evitar condições de corrida em seções críticas do código

Inclusão das bibliotecas:
- freertos/FreeRTOS.h
- freertos/task.h
- freertos/queue.h

Documentação: https://www.freertos.org/index.html

WIFI

Modos de Operação Wi-Fi do ESP32

- Station Mode (STA): No modo station, o ESP32 conecta-se a uma rede Wi-Fi existente, como um dispositivo cliente (semelhante ao funcionamento de um laptop ou smartphone). É o modo mais comum para aplicações que precisam acessar a internet ou uma rede local. Configuração típica: o ESP32 se conecta a um ponto de acesso (AP) fornecendo SSID e senha.

- Access Point Mode (AP): No modo AP, o ESP32 atua como um ponto de acesso Wi-Fi, permitindo que outros dispositivos se conectem a ele. Isso é útil para criar redes locais sem a necessidade de um roteador Wi-Fi adicional. O ESP32 pode ser configurado com seu próprio SSID e senha, e fornece endereços IP aos dispositivos conectados.

- Station + Access Point Mode (STA+AP): Este modo permite que o ESP32 funcione simultaneamente como um ponto de acesso e como um cliente. Ele pode se conectar a uma rede Wi-Fi existente e, ao mesmo tempo, permitir que outros dispositivos se conectem a ele. Este modo é útil em aplicações que exigem uma comunicação de dispositivo a dispositivo enquanto ainda têm acesso à rede externa.

- Recursos Avançados: Modo P2P (Peer-to-Peer), redes mesh, Modo Sniffer

Inclusão das bibliotecas:
- esp_mac.h
- esp_wifi.h

Documentação:
https://docs.espressif.com/projects/esp-idf/en/v5.1.2/esp32s2/api-guides/wifi.html

https://docs.espressif.com/projects/esp-idf/en/v5.1.2/esp32/api-reference/network/esp_wifi.html

MQTT

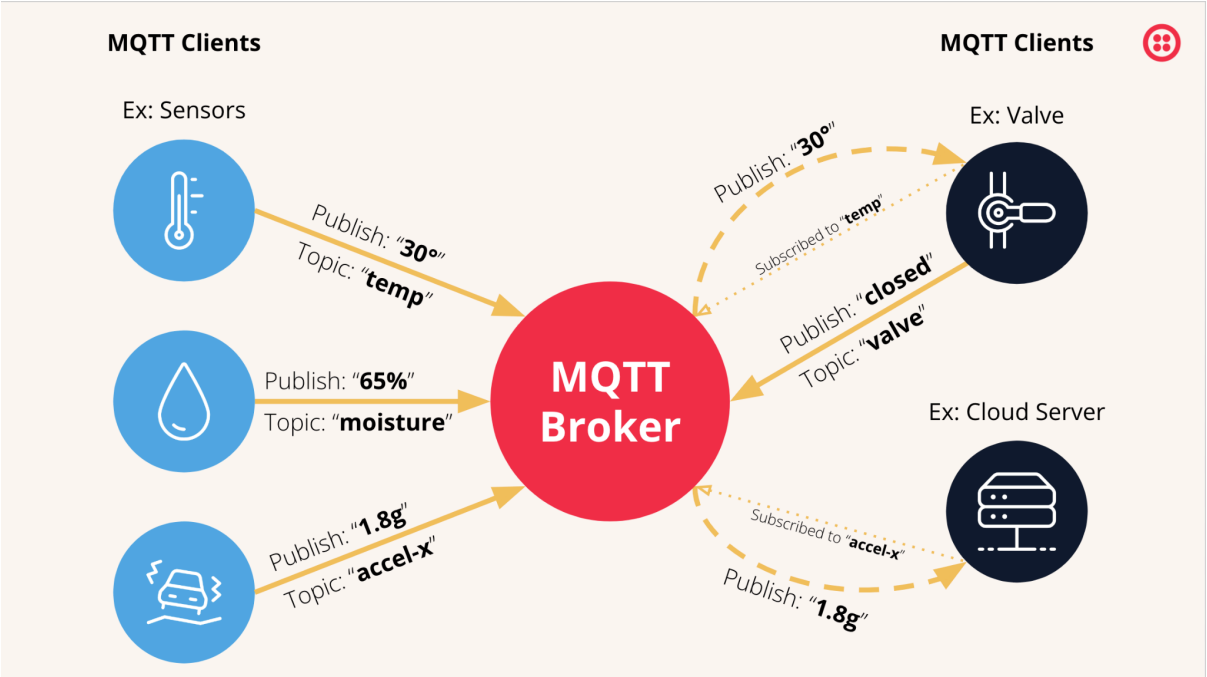
O MQTT (Message Queuing Telemetry Transport) é um protocolo de comunicação leve e eficiente, ideal para a Internet das Coisas (IoT). É amplamente utilizado para conectar dispositivos com recursos limitados, como o ESP32, a servidores ou outros dispositivos em uma rede. Aqui estão os pontos principais sobre o uso de MQTT no ESP32:

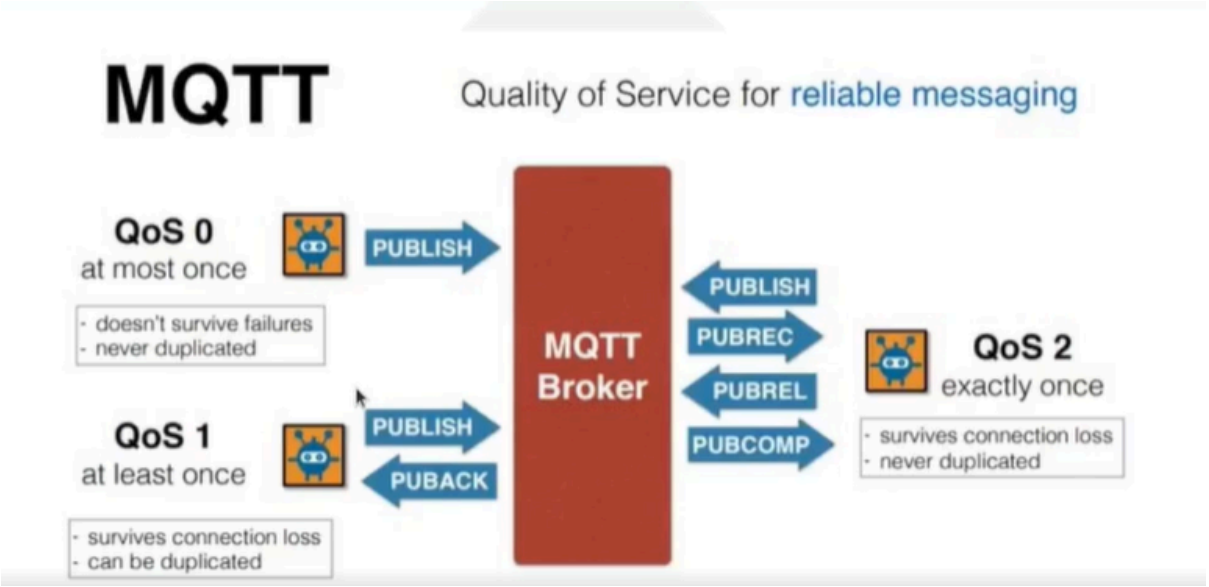
O que é MQTT?

- Protocolo de Mensagens: MQTT é um protocolo de publicação/assinatura (publish/subscribe) que facilita a comunicação entre dispositivos. Ele é baseado em tópicos, onde dispositivos podem publicar ou se inscrever para receber mensagens.
- Leve e Eficiente: Foi projetado para ser simples e de baixa sobrecarga, ideal para dispositivos com recursos limitados.
- Confiável: Oferece diferentes níveis de QoS (Quality of Service), garantindo que as mensagens sejam entregues de maneira confiável.

Como Funciona o MQTT?

- Broker: O MQTT utiliza um broker (servidor) que gerencia todos os tópicos e mensagens. Exemplos de brokers populares incluem Mosquitto, HiveMQ, e o AWS IoT.
- Cliente: Dispositivos como o ESP32 atuam como clientes que podem se conectar ao broker para publicar ou se inscrever em tópicos.
- Tópicos: São caminhos hierárquicos (strings) usados para organizar e endereçar mensagens. Exemplo: casa/sala/temperatura.





Inclusão das bibliotecas:

- mqtt_client.h

Documentação:
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mqtt.html

CRIAÇÃO COMPONENTE

Criar componentes no ESP32 envolve desenvolver módulos de software que encapsulam funcionalidades específicas, permitindo a reutilização de código e a organização do projeto. No contexto do ESP32, "componentes" geralmente se referem a bibliotecas ou módulos que podem ser integrados em aplicações maiores.
O que é necessário?

1. Criar os arquivos .c e .h com mesmo nome
2. Criar o CMakeLists.txt

```
df_component_register(SRCS "src/my_component.c"
INCLUDE_DIRS "include")
```

3. No main inclua o .h

```
#include "my_component.h"
```