Nome do Aluno: Leonardo Bonifácio Vieira Santos
Polo: Ilheus/Itabuna
Data: 04/06/2025

Controla Chão de fábrica com Mqtt

Objetivo Geral

O projeto visa desenvolver um sistema de monitoramento inteligente para fábricas, utilizando o **Raspberry Pi Pico W** como unidade central. Seu propósito é **simular e visualizar o consumo de energia de múltiplos setores(através do mqtt panel)**, fornecendo alertas em tempo real sobre o uso excessivo e garantindo uma resposta automática para prevenção de sobrecarga. Isso é alcançado através da integração de comunicação **MQTT**, feedback visual via **LEDs PWM, display OLED e matriz de LEDs WS2812**, além de alertas sonoros por meio de um **buzzer**.

Descrição Funcional

O sistema opera em um ciclo contínuo de monitoramento e resposta, gerenciado principalmente pela função checaEstadoDeEnergiaDaFabrica(), que é executada repetidamente no loop principal.

Modos de Operação:

1. **Modo de Operação Normal:**
    - Quando o sistema está **monitorando** (esta_monitorando_energia = true) e o nivel_energia_total está abaixo de 500W, o sistema opera em modo normal.
    - Os LEDs de alerta (BitDogLab) e o buzzer permanecem desligados.
    - O display OLED exibe a quantidadeSetoresEnergizados.
    - A matriz de LEDs WS2812 mostra a porcentagem de energia consumida de 0% a 40% (com o limite de 500W sendo 50%).

2. **Modo de Alerta:**
    - O sistema entra em modo de alerta quando esta_monitorando_energia é true e o nivel_energia_total está entre 501W e 999W.
    - Os LEDs de alerta (BitDogLab) acendem em branco.
    - O buzzer começa a emitir bipes intermitentes (100ms ligado, 100ms desligado) para chamar a atenção.
    - O display OLED exibe uma mensagem de alerta.

    - A matriz de LEDs WS2812 mostra a porcentagem de energia consumida de 60% a 80%.
3. **Modo Crítico/Desligamento Automático:**

    - Este modo é ativado quando esta_monitorando_energia é true e o nivel_energia_total atinge ou excede NIVEL_ENERGIA_MAXIMA (1000W).
    - Quando o sistema **entra** neste estado (detectado por !ultimo_estado_desligamento_critico), todos os **LEDs dos setores são imediatamente desligados** (nível PWM 0), e as variáveis nivel_energia_total e quantidadeSetoresEnergizados são resetadas para zero.
    - O display OLED é limpo e uma mensagem de desligamento crítico pode ser exibida.
    - Os LEDs de alerta e o buzzer também são desligados (pois o sistema "sai" do estado de alerta ao desligar os setores).

Lógica por Trás de Cada Funcionalidade:

- **Controle de Nível de Energia dos Setores**

  A função controla_energia_setor() recebe um valor (0-125) via MQTT panel para cada setor. Esse valor é diretamente aplicado como ciclo de trabalho (duty cycle) ao LED PWM correspondente, simulando o consumo. Internamente, a energia_atual de cada setor é armazenada em uma struct SectorState_t.

- **Cálculo da Energia Total**

  Após qualquer alteração no nível de energia de um setor(através dos sliders panels do mqtt panel), o nivel_energia_total é **recalculado somando a energia_atual de todos os 8 setores**. Isso garante precisão e evita erros de acumulação.

- **Comunicação MQTT:**
    - O Pico W conecta-se ao broker MQTT com credenciais definidas.
    - Ele se **inscreve** nos tópicos energia/setor[1-8] para receber comandos de controle de energia, monitoramento/energia para ligar/desligar o modo de alerta e desligar/energia/setores para um reset manual.
    - Ele **publica** o *uptime* do sistema para o tópico fabrica/uptime_s periodicamente(1 em 1 segundo graças a `TEMP_WORKER_TIME_S)`.
    - Um tópico *Last Will and Testament* (/online) publica o status de conexão do dispositivo (1 para online, 0 para offline inesperado).

- **Bipes Intermitentes**

  A lógica do buzzer em checaEstadoDeEnergiaDaFabrica() é **não bloqueante**. Ela usa time_us_64() e uma flag ultima_troca_de_estado_buzzer_us para alternar o estado do buzzer (ligar/desligar) a cada BUZZER_INTERVALO_ATIVAMENTO (200ms) milissegundos, sem pausar o restante do programa.

- **Matriz de LEDs WS2812:**

  A função mostra_porcentagem_de_energia_pela_matriz_de_leds() calcula a porcentagem do nivel_energia_total. Para evitar o *flicker*, a matriz é atualizada (via set_one_led()) **apenas quando a porcentagem calculada cruza um novo limiar** (0%, 20%, 40%, 60%, 80%, 100%), garantindo que a atualização visual seja suave e eficiente.

Uso dos periféricos da bitdoglab

- **Protocolo WI-FI (CYW43):**
  - **Utilização:** O Raspberry Pi Pico W utiliza o chip CYW43439 para estabelecer a conexão Wi-Fi com uma rede local.
  - **Função no Projeto:** Permite a comunicação com o broker MQTT, enviando dados de telemetria (uptime) e recebendo comandos de controle (nível de energia dos setores, monitoramento, desligamento). É essencial para a funcionalidade de "fábrica inteligente" e controle remoto.

- **Potenciômetro do Joystick:**
  - **Utilização:** Não foi utilizado, mas melhorias futuras para este projeto incluem aumento ou diminuição dos brilho dos leds(energia do setor) pelo joystick da bitdoglab ou um potênciometro a parte, fazendo com que seja possível não só mudar os valores pelo mqtt panel mas sim também pela própria bitdoglab, além de calcular a média de energia gasta em determinado período de tempo.

- **Display OLED (SSD1306 - I2C):**
  - **Utilização:** O display OLED monocromático SSD1306 é conectado via protocolo I2C.

- **Função no Projeto:** Fornece feedback visual local sobre o estado do sistema. Ele pode exibir a quantidade de setores energizados ou mensagens de alerta (ssd1306_draw_alert_message), dependendo do estado da fábrica. A comunicação I2C (i2c_init, gpio_set_function(GPIO_FUNC_I2C)) é configurada a 400kHz.

- **Matriz de LEDs (WS2812 - PIO):**
  - **Utilização:** A matriz de leds ws2812 com leds Programáveis por PIO do Pico, são usadas para emitir alerta de porcentagem.
  - **Função no Projeto:** Atua como um medidor visual de porcentagem do nivel_energia_total. Diferentes padrões de LEDs (_0_porcento_de_energia, _20_porcento_de_energia, etc.) são acesos com cores específicas para indicar as faixas de consumo. A lógica de atualização inteligente (somente em mudança de limiar) evita o *flicker*, garantindo uma experiência visual suave.

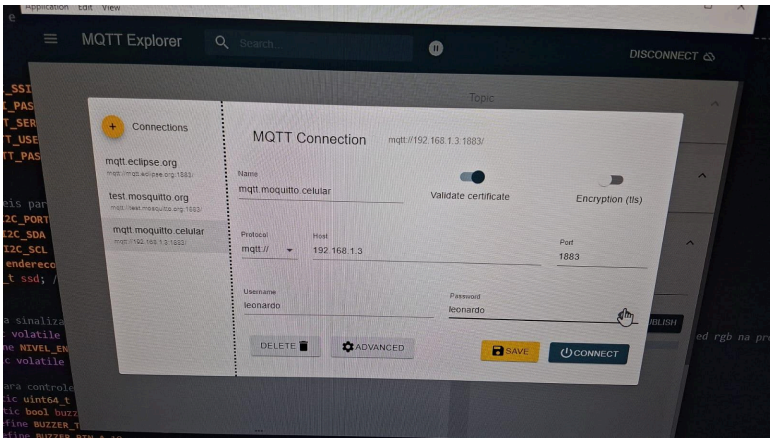
- **LED RGB (BitDogLab):**
  - **Utilização:** Os LEDs Azuis, Verdes e Vermelhos da BitDogLab (GPIOs 11, 12, 13) são usados como indicadores de estado de alerta.
  - **Função no Projeto:** São acesos em conjunto (resultando em luz branca) quando o sistema está em estado de alerta (energia total > 500W), fornecendo um aviso visual claro. São desligados quando o sistema retorna ao estado normal ou quando o sistema entra em estado crítico.

- **Buzzer (PWM):**
  - **Utilização:** O Buzzer da GPIO 10 foi configurado para PWM
  - **Função no Projeto:** Emite um alerta sonoro. Em estado de alerta, o buzzer "bipa" de forma intermitente (100ms ligado, 100ms desligado), utilizando um controle de tempo não bloqueante para não pausar o sistema. A frequência do tom é definida pelo wrap e clkdiv do PWM.

- **Tratamento de Debounce dos Botões:**
  - **Utilização:** Não foram utilizados botões da BitDogLab, somente botões e sliders do mqtt panel

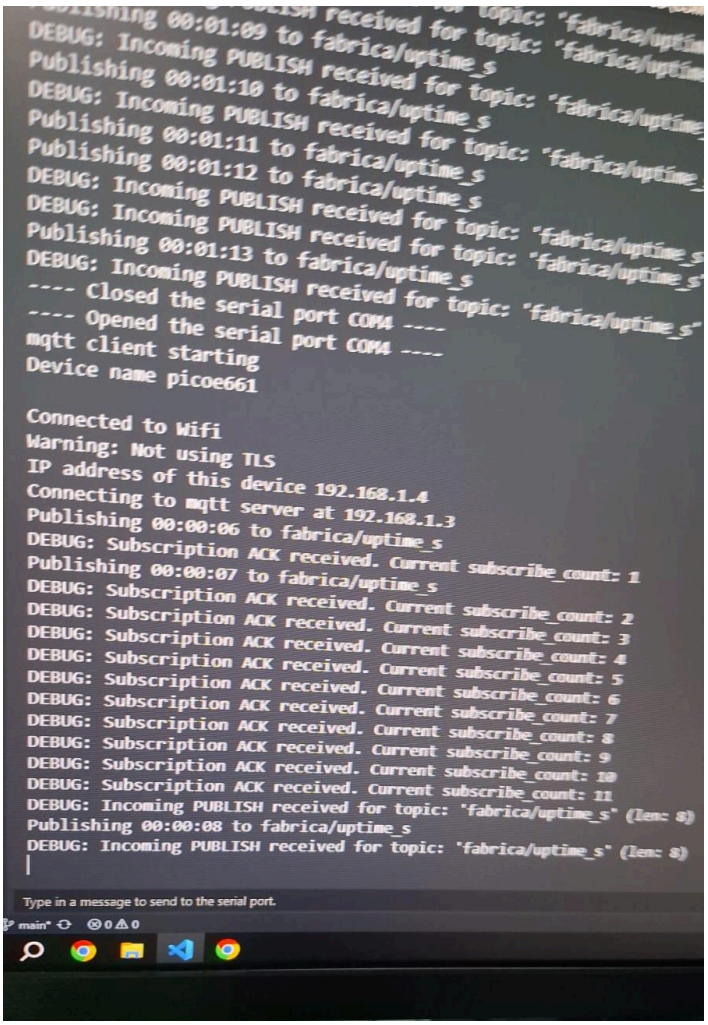
Imagens para melhor entendimento

Conexão do mqtt Explorer



Tela para controle criada no mqtt panel

Mensagens de depuração no Terminal



Mensagens no termux ao se conectar ao broker mosquitto

Links para acesso ao código e ao vídeo.

GitHub: https://github.com/LeonardoBonifacio/ControlaChaoDeFabricaMqtt
Youtube:https://youtu.be/8aXiQZ6gf78