 # OBJETIVO

O desafio proposto abrange os conceitos principais do desenvolvimento de embarcados/IoT, utilizando o framework ESP-IDF.

Deverá ser desenvolvido um projeto de hardware e software para o microcontrolador ESP32 (preferencialmente S3), com sensor DHT22 de temperatura e umidade, LED RBG compatível Neopíxel e botão.

O desenvolvimento pode ser feito usando a ferramenta Wokwi, ou com um microcontrolador físico [(1)].

 # HARDWARE

1) Microcontrolador ESP32 (preferencialmente S3);
2) Um LED RGB Compatível com Neopixel (pode usar o integrado ao ESP32-S3 ou usar um adicional);
3) Um sensor DHT22 de temperatura e umidade;
4) Um pushbutton de pull up.

 # SOFTWARE

O software deve ser desenvolvido usando prioritariamente as APIs do ESP-IDF. As APIs da plataforma Arduino devem ser utilizadas apenas para a operação das bibliotecas que não estiverem disponíveis para o ESP-IDF (ex: DHT22, etc).

O sistema deve operar com 4 tarefas (xTaskCreatePinnedToCore) distintas, operando divididas nos 2 cores de processamento do ESP32, conforme a seguir:

1) **(core 1)** Gerenciamento do LED RGB, com as seguintes funcionalidades:
    a) Possibilidade de ajustar as cores em 4 variações (Verde, Amarelo, Vermelho e Azul);
    b) Possibilidade de operar em 4 estados distintos de operação (Desligado, Ligado, Piscando em intervalos de 1 segundos, piscando em intervalos de 0,3 segundos).
2) **(core 1)** Gerenciamento do sensor de temperatura e umidade. A tarefa deve ler e atualizar os valores de temperatura e umidade;

3) **(core 0)** Gerenciamento do pushbutton, detectando:
    a) pressionamento rápido (abaixo de 1 seg). Quando o botão for pressionado de forma rápida, o estado de operação do LED RGB deve mudar de intervalo;
    b) pressionamento longo (acima de 3 seg). Quando o botão for pressionado por um longo período, o LED RGB deve mudar de cor.
4) **(core 0)** Relatório de estados. A tarefa deve exibir na console, a cada 3 segundos, os dados de temperatura, umidade e estado do LED RGB (cor e estado de operação).

 # DESAFIOS

1) Usando a tarefa de relatório de estados, enviar as informações de estado em formato JSON para um broker MQTT (temperatura, umidade, cor e estado de operação do LED RGB);
2) Receber comandos por MQTT para trocar a cor ou o estado de operação do LED RGB;
3) Fazer a operação do botão usando interrupções.

 # SUGESTÕES E DICAS

- ➔ Sugere-se usar como projeto base no Wokwi o ESP32-S3 com framework Arduino: https://wokwi.com/projects/new/esp32-s3;
- ➔ Na sua programação, leve em consideração que as tarefas estão operando em cores de processamento diferentes, o que pode gerar problemas de concorrência. O sistema não deve falhar/reiniciar nestas situações. Como você muito provavelmente estará executando o firmware no ambiente emulado do Wokwi, os problemas de concorrência não irão se manifestar. Porém, ao executar o software em dispositivos reais, esse é um problema frequente que deve ser considerado;
- ➔ Para o desafio de envio de mensagens MQTT, sugerimos usar o broker público SaaS HiveMQ: https://www.hivemq.com/demos/websocket-client/ (veja snippets de código para conexão ao broker abaixo);
- ➔ Para conectar o seu projeto Wokwi ao WiFi / Internet, será necessário baixar e executar o software de Gateway: https://github.com/wokwi/wokwigw/releases/ (veja snippets de código abaixo);
- ➔ Recomendamos usar as seguintes bibliotecas (https://docs.wokwi.com/pt-BR/guides/libraries):
    - ESP-IDF mqtt_client para comunicação MQTT: https://docs.espressif.com/projects/esp-idf/en/v5.2.1/esp32s3/api-reference/protocols/mqtt.html

 # NOTAS

(1) - Se for utilizado o microcontrolador físico, deve-se enviar além do código fonte:

- um arquivo esquemático em formato PDF;
- uma foto do sistema montado;
- um vídeo de demonstração da operação do software e do firmware.

 # REFERÊNCIAS E SNIPPETS DE CÓDIGO

WiFi

https://docs.espressif.com/projects/esp-idf/en/v5.2.1/esp32s3/api-guides/wifi.html
https://github.com/espressif/esp-idf/blob/master/examples/wifi/getting_started/station/main/station_example_main.c

```
C/C++
#define EXAMPLE_ESP_WIFI_SSID "Wokwi-GUEST"
#define EXAMPLE_ESP_WIFI_PASS ""
```

MQTT

https://docs.espressif.com/projects/esp-idf/en/v5.2.1/esp32s3/api-reference/protocols/mqtt.html
https://github.com/espressif/esp-idf/blob/v5.0/examples/protocols/mqtt/tcp/main/app_main.c

```
C/C++
#define CONFIG_BROKER_HOSTNAME "broker.hivemq.com"
#define CONFIG_BROKER_PORT 1883
const esp_mqtt_client_config_t mqtt_cfg = {
  .broker.address.hostname = CONFIG_BROKER_HOSTNAME,
  .broker.address.port = CONFIG_BROKER_PORT,
  .broker.address.transport = MQTT_TRANSPORT_OVER_TCP,
};
esp_mqtt_client_handle_t client;
```

 # ENTREGA

O desafio será considerado entregue quando:

a) Enviar o link do repositório Github ou Bitbucket, para o e-mail nicolau@power2go.com.br, com o título “Entrega do Desafio Embarcados/IoT”;
b) O desafio precisa ser entregue até a dia 03/06/2024 às 23:59;
c) No repositório, arquivo README.md, deve conter instruções claras de como compilar e testar o software;
d) Se o projeto for desenvolvido usando a platarforma Wokwi, devem ser publicados no repositório os arquivos:
    i) sketch.ino / main.c;
    ii) diagram.json;
    iii) library.txt (se forem utilizadas bibliotecas);
    iv) outros arquivos (.c, .cpp, .h) que forem adicionados ao projeto;

 # IMPORTANTE

Caso ainda não tenha realizado o preenchimento do formulário, clique no link e complete o seu perfil: https://forms.gle/1hnYkvbHjgY9PRm56