**Controlador Remoto de led RGB**

**Disciplina:** Projeto Interdisciplinar I

**Docente:** Sidinei de Andrade

**Alunos:**

30316600 - Elvis Leite Diniz

29073642 - Gabriel Francisco de Abreu

30221463 - Leonardo Serafim Pinton

Salto

2022

**Introdução**

Por meio do projeto proposto foi possível estudarmos a fundo como funciona um microcontrolador. Foi possível tocar e até mesmo programar o mesmo, no caso utilizamos uma placa ESP32 para criar um controlador de led RGB controlado remotamente via WIFI usando o protocolo MQTT que faz a troca de mensagens através de tópicos.

Para que o fluxo das mensagens funcione corretamente, se faz necessário a utilização de uma rede com conexão a internet, ou de uma rede interna. De ambas as maneiras, é necessário fazer a configuração da rede tanto no código do ESP32, quanto no aplicativo MQTT Dashboard.

**Descrição do Projeto**

Nosso projeto se trata de algo simples, elaboramos um controlador de led RGB usando um microcontrolador ESP32 como base de tudo, a partir dele foi possível programa-lo para que seja possível controlar remotamente o led RGB presente no mesmo.

Utilizamos a popular linguagem de programação chamada "Linguagem C", que é utilizada em diversos projetos por conta de sua flexibilidade; sendo possível criar aplicativos de todos os tipos, drivers e outros controladores de dispositivos e até mesmo sistemas operacionais.

Toda programação foi feita para funcionar no sistema operacional FreeRTOS que é nada menos que um sistema operacional em tempo real de código aberto e neutro na nuvem que oferece um kernel rápido, confiável e responsivo. O mesmo é nativo da placa ESP32.



Foto 01: Overview do projeto

**Pesquisa Bibliográfica**

O ESP32 é uma família de microcontroladores de baixo custo e baixo consumo de energia. É também um sistema em um chip que integra um microcontrolador, Wi-Fi e Bluetooth. A série ESP32 apresenta versões dual-core e single-core do microprocessador Tensilica Xtensa LX6, incluindo antenas balun de RF integradas, amplificadores de potência, receptores amplificados de baixo ruído, filtros, gerenciamento de energia do módulo. O ESP32 foi criado e desenvolvido pela Espressif Sistemas, uma empresa chinesa com sede em Xangai, e fabricado pela TSMC usando seu processo de fabricação de 40nm. É o sucessor do microcontrolador ESP8266.

O sistema utilizado na placa ESP32 é o FreeRTOS, que é nada menos que um sistema operacional em tempo real (RTOS) pequeno, portátil, preventivo e de código aberto para microcontroladores. Foi adaptado para mais de 40 arquiteturas diferentes. Criado em 2003 por Richard Barry e a equipe do FreeRTOS, é hoje um dos sistemas operacionais de tempo real mais utilizados no mercado. Desde 29 de novembro de 2017, o FreeRTOS está disponível gratuitamente sob a licença MIT. Versões anteriores estão disponíveis sob uma licença GPL modificada e podem ser usadas sem pagamento de royalties. A licença não obriga os desenvolvedores a liberar seu código de software de aplicativo, mas exige que eles mantenham o kernel de código aberto FreeRTOS. A High Integrity Systems também oferece licenças comerciais com suporte provisório (OpenRTOS).

Para ser feita a troca de mensagens entre a placa ESP32 e o aplicativo, utilizamos o aplicativo MQTT Dashboard que serviu de ponte para o protocolo MQTT que é um protocolo de mensagens leve para sensores e pequenos dispositivos móveis otimizados para redes TCP/IP. O esquema de troca de mensagens é baseado no modelo Publisher-Subscriber, que é extremamente simples e leve. O princípio arquitetônico é minimizar a largura de banda da rede e o uso de recursos do dispositivo, garantindo a confiabilidade e um certo grau de garantia de entrega. Esses princípios tornam o protocolo ideal para comunicações "machine-to-machine" (M2M) emergentes e aplicativos "Internet of Things", um mundo de dispositivos conectados e aplicativos móveis onde a largura de banda e a energia da bateria são uma preocupação. Suas versões atuais são 5.0 e 3.1.1 (padrão ISO), ambos padrões OASIS. MQTT-SN é uma variante do protocolo para redes que não são baseadas em TCP/IP, como ZigBee. Hoje, o MQTT é amplamente utilizado em indústrias como automotiva, manufatura, telecomunicações, petróleo e gás.

**Lista de materiais**

* **Hardware:**

- 1 ESP32 DevikitC v1

- 1 LED RGB

- 3 Resistores 220 Ohms 5%

- 1 Celular com SO Android

- 1 Protoboard

* **Software:**

- FreeRTOS (sistema nativo do ESP32)

- Eclipse com plugins da Espressif (IDE)

- MQTT Dash (app de celular para monitoramento e comandos MQTT)

**Programação comentada:**

* **Definição da configuração da rede e broker**

#define CONFIG\_WIFI\_SSID "terahdeth"

#define CONFIG\_WIFI\_PASSWORD "iuashdiuhsad"

#define CONFIG\_BROKER\_URL "mqtt://mqtt.eclipseprojects.io:1883"

* **Declaração das variáveis**

int rele1 = 0;

int rele2 = 0;

int rele3 = 0;

int vermelho = 0;

int verde = 0;

int azul = 0;

int amarelo = 0;

int roxo = 0;

int azulclaro = 0;

int branco = 0;

int multicor = 0;

int tempoPolicia = 500;

int tempoBalada = 200;

int vermelhoComando = 0;

int verdeComando = 0;

int azulComando = 0;

* **Subscrição dos tópicos mqtt do projeto**

case MQTT\_EVENT\_CONNECTED:

ESP\_LOGI(TAG, "MQTT\_EVENT\_CONNECTED");

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/vermelho", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/verde", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/azul", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/amarelo", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/roxo", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/azulclaro", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/branco", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/case", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/tempoPolicia", 1);

msg\_id = esp\_mqtt\_client\_subscribe(client, "ceunsp/tempoBalada", 1);

* **Atribuição dos valores dos tópicos mqtt para variáveis internas do código**

case MQTT\_EVENT\_DATA:

//ESP\_LOGI(TAG, "MQTT\_EVENT\_DATA");

sprintf(topic\_name, "%.\*s", event->topic\_len, event->topic);

sprintf(strData, "%.\*s", event->data\_len, event->data);

if (strcmp(topic\_name, "ceunsp/vermelho") == 0) {

vermelho = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/verde") == 0) {

verde = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/azul") == 0) {

azul = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/amarelo") == 0) {

amarelo = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/roxo") == 0) {

roxo = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/azulclaro") == 0) {

azulclaro = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/branco") == 0) {

branco = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/case") == 0) {

multicor = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/tempoPolicia") == 0) {

tempoPolicia = atoi(strData);

}

if (strcmp(topic\_name, "ceunsp/tempoBalada") == 0) {

tempoBalada = atoi(strData);

}

break;

* **Pré configuração e parametrização das GPIOs dentro da função switch\_io**

//Configuração das GPIOs

gpio\_pad\_select\_gpio(red);

gpio\_set\_pull\_mode(red, GPIO\_PULLDOWN\_ONLY);

gpio\_set\_direction(red,GPIO\_MODE\_OUTPUT);

gpio\_pad\_select\_gpio(green);

gpio\_set\_pull\_mode(green, GPIO\_PULLDOWN\_ONLY);

gpio\_set\_direction(green,GPIO\_MODE\_OUTPUT);

gpio\_pad\_select\_gpio(blue);

gpio\_set\_pull\_mode(blue, GPIO\_PULLDOWN\_ONLY);

gpio\_set\_direction(blue,GPIO\_MODE\_OUTPUT);

* **Acionamento dos GPIOs de acordo com o valor das variáveis de comando**

while(1) { //Loop Infinito

if(vermelhoComando){

gpio\_set\_level(red, 0);

}else{

gpio\_set\_level(red, 1);

}

if(verdeComando){

gpio\_set\_level(green, 0);

}else{

gpio\_set\_level(green, 1);

}

if(azulComando){

gpio\_set\_level(blue, 0);

}else{

gpio\_set\_level(blue, 1);

}

* **switch case para acionamento dos pinos do LED RGB**

switch(multicor){

case 0:

vermelhoComando = 0;

verdeComando = 0;

azulComando = 0;

break;

case 1:

esp\_mqtt\_client\_publish(client, "ceunsp/verde", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azul", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/amarelo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/roxo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azulclaro", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/branco", "0", 0, 0, 0);

vermelhoComando = 1;

verdeComando = 0;

azulComando = 0;

break;

case 2:

esp\_mqtt\_client\_publish(client, "ceunsp/vermelho", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azul", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/amarelo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/roxo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azulclaro", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/branco", "0", 0, 0, 0);

vermelhoComando = 0;

verdeComando = 1;

azulComando = 0;

break;

case 3:

esp\_mqtt\_client\_publish(client, "ceunsp/vermelho", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/verde", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/amarelo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/roxo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azulclaro", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/branco", "0", 0, 0, 0);

vermelhoComando = 0;

verdeComando = 0;

azulComando = 1;

break;

case 4:

esp\_mqtt\_client\_publish(client, "ceunsp/vermelho", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/verde", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azul", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/roxo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azulclaro", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/branco", "0", 0, 0, 0);

vermelhoComando = 1;

verdeComando = 1;

azulComando = 0;

break;

case 5:

esp\_mqtt\_client\_publish(client, "ceunsp/vermelho", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/verde", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azul", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/amarelo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azulclaro", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/branco", "0", 0, 0, 0);

vermelhoComando = 1;

verdeComando = 0;

azulComando = 1;

break;

case 6:

esp\_mqtt\_client\_publish(client, "ceunsp/vermelho", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/verde", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azul", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/amarelo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/roxo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/branco", "0", 0, 0, 0);

vermelhoComando = 0;

verdeComando = 1;

azulComando = 1;

break;

case 7:

esp\_mqtt\_client\_publish(client, "ceunsp/vermelho", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/verde", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azul", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/amarelo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/roxo", "0", 0, 0, 0);

esp\_mqtt\_client\_publish(client, "ceunsp/azulclaro", "0", 0, 0, 0);

vermelhoComando = 1;

verdeComando = 1;

azulComando = 1;

break;

case 8:

while(multicor == 8){

gpio\_set\_level(red, 0);

vTaskDelay(tempoPolicia/ portTICK\_PERIOD\_MS);

gpio\_set\_level(red, 1);

gpio\_set\_level(blue, 0);

vTaskDelay(tempoPolicia/ portTICK\_PERIOD\_MS);

gpio\_set\_level(blue, 1);

}

break;

case 9:

while(multicor == 9){

gpio\_set\_level(red, 0);

vTaskDelay(tempoBalada/ portTICK\_PERIOD\_MS);

gpio\_set\_level(blue, 0);

vTaskDelay(tempoBalada/ portTICK\_PERIOD\_MS);

gpio\_set\_level(red, 1);

vTaskDelay(tempoBalada/ portTICK\_PERIOD\_MS);

gpio\_set\_level(green, 0);

vTaskDelay(tempoBalada/ portTICK\_PERIOD\_MS);

gpio\_set\_level(blue, 1);

vTaskDelay(tempoBalada/ portTICK\_PERIOD\_MS);

gpio\_set\_level(red, 0);

gpio\_set\_level(blue, 0);

vTaskDelay(tempoBalada/ portTICK\_PERIOD\_MS);

gpio\_set\_level(blue, 1);

gpio\_set\_level(green, 1);

}

break;

default :

printf ("Valor invalido!\n");

}

vTaskDelay(200/ portTICK\_PERIOD\_MS);

}

vTaskDelay(1000/ portTICK\_PERIOD\_MS);

}

* **Função principal: Responsável pela chamada das funções/tasks do código**

void app\_main()

{

if( esp\_flash\_encryption\_enabled() )

{

printf("\r\nStatus: Encryption Enabled\n");

}

ESP\_LOGI(TAG, "[APP] Startup..");

ESP\_LOGI(TAG, "[APP] Free memory: %d bytes", esp\_get\_free\_heap\_size());

ESP\_LOGI(TAG, "[APP] IDF version: %s", esp\_get\_idf\_version());

esp\_log\_level\_set("\*", ESP\_LOG\_INFO);

esp\_log\_level\_set("MQTT\_CLIENT", ESP\_LOG\_VERBOSE);

esp\_log\_level\_set("TRANSPORT\_TCP", ESP\_LOG\_VERBOSE);

esp\_log\_level\_set("TRANSPORT\_SSL", ESP\_LOG\_VERBOSE);

esp\_log\_level\_set("TRANSPORT", ESP\_LOG\_VERBOSE);

esp\_log\_level\_set("OUTBOX", ESP\_LOG\_VERBOSE);

ESP\_ERROR\_CHECK(nvs\_flash\_init());

wifi\_init();

mqtt\_app\_start();

xTaskCreatePinnedToCore(publish\_task,"publish\_task",2048,NULL,1,NULL,0);

xTaskCreatePinnedToCore(switch\_io,"liga\_desliga",2048,NULL,2,NULL,1);

}

**Conclusão:**

Visando o aprendizado sobre Arduino foi realizado um estudo aprofundado sobre Iot, placas de Arduino e automatização. Por meio deste estudo foi possível ver como projetos em Arduino podem ajudar em tarefas do dia-dia e fazer com que possamos automatizar muitas delas, fazendo assim uma grande economia de tempo. Com o projeto de controle remoto de RGB podemos aprender a utilizar o protocolo MQTT, escrever códigos na linguagem C e fazer toda a montagem da placa ESP32.

**Referências:**

**Espressif**. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 10 de nov. de 2022.

**FreeRTOS**. Disponível em: <https://www.freertos.org>. Acesso em: 10 de nov. de 2022.

**Eclipse**. Disponível em: <https://www.eclipse.org/downloads/>. Acesso em: 10 de nov. de 2022.

**Mosquito MQTT Eclipse foundation**. Disponível em: <https://mosquitto.org>. Acesso em: 10 de nov. de 2022.

**MQTT Dash**. Disponível em: <https://embarcados.com.br/mqtt-dash/>. Acesso em: 10 de nov. de 2022.