Utilizando o LittleFS do ESP32

Elias de Almeida Sombra Neto[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) - Campus Maracanaú
Av. Parque Central, 1315 - Distrito Industrial I, Maracanaú-CE, Brasil

elias.almeida09@aluno.ifce.edu.br

Abstract. *This report presents a practical activity using the ESP32 in the Microcontrollers course at IFCE. The LittleFS library was used to store Wi-Fi credentials and the RGB LED state in a JSON file, controlled by a button. ArduinoJSON simplified data handling. The setup used basic components and was successfully tested.*

Resumo. *Este relatório descreve uma atividade prática com ESP32 na disciplina de Microcontroladores do IFCE. Utilizou-se a biblioteca LittleFS para armazenar em JSON as credenciais de redes Wi-Fi e o estado de um LED RGB controlado por botão. A biblioteca ArduinoJSON facilitou a manipulação dos dados. O sistema foi montado com componentes simples e validado com sucesso.*

1. Introdução

Este relatório descreve uma atividade prática realizada no Laboratório de Eletroeletrônica e Sistema Embarcados (LAESE) durante a disciplina de Microcontroladores no Instituto Federal do Ceará (IFCE).

O objetivo dessa atividade é utilizar a biblioteca LittleFS do ESP32 para gerenciamento de arquivos salvos no microcontrolador. Para auxiliar na realização dessa prática foi importada a bilbioteca ArduinoJSON por meio do PlatformIO. Dentro do arquivo JSON, serão armazenadas as credenciais de redes para o ESP32 tentar se conectar, além das informações de cor acesa do LED RGB e o estado de cada uma das três cores do componente (vermelho, verde e azul). A mudança de estado de cada cor é controlada por meio de um botão conectado no circuito.

2. Materiais utilizados

Os materiais para a construção e acionamento do circuito incluem:

- 1 ESP32 30 pinos;
- 1 cabo micro USB;
- 1 protoboard;
- 1 LED RGB;
- 1 botão;
- 2 resistores;
- 5 cabos jumper macho-fêmea.

3. Montagem do circuito

Para realizar a montagem do circuito, é preciso seguir os seguintes passos:

- Ligar o ESP32 em uma fonte de alimentação com o cabo micro USB.
- Ligar o pino GND do ESP32 na região de alimentação da protoboard através de um jumper macho-fêmea.
- Posicionar o LED RGB na região de componentes da protoboard.
- Posicionar um pino do resistor na região de alimentação e a outra em paralelo com o pino negativo do LED RGB.
- Ligar o GPIO13 do ESP32 em paralelo com o pino correspondente a cor vermelha do LED RGB através de um jumper macho-fêmea.
- Ligar o GPIO12 do ESP32 em paralelo com o pino correspondente a cor verde do LED RGB através de um jumper macho-fêmea.
- Ligar o GPIO14 do ESP32 em paralelo com o pino correspondente a cor azul do LED RGB através de um jumper macho-fêmea.
- Posicionar o botão na região de componentes da protoboard.
- Posicionar um pino do resistor na região de alimentação e a outra em paralelo com um dos pinos do botão.
- Ligar o GPIO27 do ESP32 em paralelo com outro pino do botão através de um jumper macho-fêmea.

4. Implementação do código

O código responsável por acessar o arquivo JSON do ESP32, conectar-se na rede WiFi com as credenciais armazenadas e salvar as informações do estado atual do LED RGB foi desenvolvido com a linguagem C++ no editor de código Visual Studio Code juntamente com sua extensão PlatformIO.

```
#include <Arduino.h>
#include <WiFi.h>
#include <LittleFS.h>
#include <ArduinoJson.h>

#define RED_LED_PIN 13
#define GREEN_LED_PIN 12
#define BLUE_LED_PIN 14
#define BUTTON_PIN 27
#define FORMAT_LITTLEFS_IF_FAILED true

int ledCurrentlyOn = 0;
const char* jsonFilePath = "/config.json";
String ledColor = "nenhuma";

void saveJSON();
void connectWiFiFromJSON();
void addWiFiNetworkToJSON(const char*, const char*);
void printSavedConfig();

void saveJSON() {
```

```
  DynamicJsonDocument doc(1024);
  File file = LittleFS.open("/config.json", "r");
  if (file) {
    DeserializationError error = deserializeJson(doc, file);
    file.close();
    if (error) {
      Serial.println("Erro ao ler JSON existente, substituindo.");
      doc.clear();
    }
  }
  if (!doc.containsKey("wifi") || !doc["wifi"].is<JsonArray>()) {
    doc["wifi"] = JsonArray();
  }
  JsonObject led = doc.createNestedObject("led");
  led["color"] = ledColor;
  led["red"] = digitalRead(RED_LED_PIN);
  led["green"] = digitalRead(GREEN_LED_PIN);
  led["blue"] = digitalRead(BLUE_LED_PIN);
  file = LittleFS.open("/config.json", "w");
  if (!file) {
    Serial.println("Erro ao abrir arquivo para escrita");
    return;
  }
  serializeJsonPretty(doc, file);
  file.close();
  printSavedConfig();
}

void connectWiFiFromJSON() {
  File file = LittleFS.open(jsonFilePath, "r");
  if (!file) {
    Serial.println("Arquivo JSON nao encontrado. Criando um novo.");
    saveJSON();
    return;
  }
  DynamicJsonDocument doc(512);
  DeserializationError error = deserializeJson(doc, file);
  file.close();
  if (error) {
    Serial.println("Erro ao ler JSON");
    return;
  }
  JsonArray wifiList = doc["wifi"];
  for (JsonObject net : wifiList) {
    const char* ssid = net["ssid"];
    const char* password = net["password"];
```

```
68      Serial.printf("Tentando conectar em: %s\n", ssid);
69      WiFi.begin(ssid, password);
70      unsigned long startTime = millis();
71      while (WiFi.status() != WL_CONNECTED && (millis() -
           startTime) < 8000) {
72        if (WiFi.status() == WL_CONNECTED) break;
73        yield();
74      }
75      if (WiFi.status() == WL_CONNECTED) {
76        Serial.printf("Conectado em: %s\n", ssid);
77        return;
78      }
79      WiFi.disconnect(true);
80      Serial.printf("Falha ao conectar em: %s\n", ssid);
81    }
82    Serial.println("Nao foi possivel conectar a nenhuma rede.");
83  }
84
85  void addWiFiNetworkToJSON(const char* ssid, const char* password
       ) {
86    DynamicJsonDocument doc(1024);
87    File file = LittleFS.open("/config.json", "r");
88    if (file) {
89      DeserializationError error = deserializeJson(doc, file);
90      file.close();
91      if (error) {
92        Serial.println("Erro ao ler JSON existente. Criando novo."
             );
93        doc.clear();
94      }
95    } else {
96      Serial.println("Arquivo nao encontrado. Criando novo.");
97    }
98    if (!doc.containsKey("wifi")) {
99      doc.createNestedArray("wifi");
100   }
101   JsonArray wifiList = doc["wifi"];
102   for (JsonObject net : wifiList) {
103     if (net["ssid"] == ssid) {
104       Serial.println("Essa rede ja esta salva. Ignorando.");
105       return;
106     }
107   }
108   JsonObject newNet = wifiList.createNestedObject();
109   newNet["ssid"] = ssid;
110   newNet["password"] = password;
111   file = LittleFS.open("/config.json", "w");
112   if (!file) {
113     Serial.println("Erro ao abrir arquivo para escrita.");
```

```
114     return;
115   }
116   serializeJsonPretty(doc, file);
117   file.close();
118   Serial.println("Nova rede adicionada ao JSON com sucesso!");
119   printSavedConfig();
120 }
121
122 void printSavedConfig() {
123   File file = LittleFS.open("/config.json", "r");
124   if (!file) {
125     Serial.println("Arquivo de configuracao nao encontrado.");
126     return;
127   }
128   DynamicJsonDocument doc(1024);
129   DeserializationError error = deserializeJson(doc, file);
130   file.close();
131   if (error) {
132     Serial.print("Erro ao ler JSON: ");
133     Serial.println(error.c_str());
134     return;
135   }
136   Serial.println("=== Configuracao Salva no JSON ===");
137   if (doc.containsKey("wifi") && doc["wifi"].is<JsonArray>()) {
138     JsonArray wifiList = doc["wifi"].as<JsonArray>();
139     if (wifiList.size() > 0) {
140       Serial.println("Redes WiFi:");
141       for (JsonObject net : wifiList) {
142         const char* ssid = net["ssid"];
143         const char* password = net["password"];
144         Serial.printf("  - SSID: %s, Senha: %s\n", ssid ? ssid :
              "(vazio)", password ? password : "(vazio)");
145       }
146     } else {
147       Serial.println("Nenhuma rede WiFi salva.");
148     }
149   } else {
150     Serial.println("Campo 'wifi' ausente ou invalido.");
151   }
152   if (doc.containsKey("led") && doc["led"].is<JsonObject>()) {
153     JsonObject led = doc["led"];
154     Serial.println("Estado do LED:");
155     Serial.printf("  - Cor atual: %s\n", led["color"] | "nenhuma
          ");
156     Serial.printf("  - Vermelho: %s\n", led["red"] ? "ligado" :
          "desligado");
157     Serial.printf("  - Verde: %s\n", led["green"] ? "ligado" : "
          desligado");
158     Serial.printf("  - Azul: %s\n", led["blue"] ? "ligado" : "
```

```
        desligado");
159   } else {
160     Serial.println("Estado do LED nao definido.");
161   }
162   Serial.println("==================================");
163 }
164
165 void setup() {
166   Serial.begin(9600);
167   pinMode(RED_LED_PIN, OUTPUT);
168   pinMode(GREEN_LED_PIN, OUTPUT);
169   pinMode(BLUE_LED_PIN, OUTPUT);
170   pinMode(BUTTON_PIN, INPUT_PULLUP);
171   if (!LittleFS.begin(FORMAT_LITTLEFS_IF_FAILED)) {
172     Serial.println("Falha ao montar LittleFS");
173     return;
174   }
175   addWiFiNetworkToJSON("<WIFI>", "<PASSWORD>");
176   connectWiFiFromJSON();
177   printSavedConfig();
178 }
179
180 void loop() {
181   static bool buttonPressed = false;
182   int buttonState = digitalRead(BUTTON_PIN);
183   if (buttonState == LOW && !buttonPressed) {
184     buttonPressed = true;
185     if (ledCurrentlyOn == 0) {
186       digitalWrite(RED_LED_PIN, HIGH);
187       ledColor = "vermelho";
188       ledCurrentlyOn = RED_LED_PIN;
189     } else if (ledCurrentlyOn == RED_LED_PIN) {
190       digitalWrite(RED_LED_PIN, LOW);
191       digitalWrite(GREEN_LED_PIN, HIGH);
192       ledColor = "verde";
193       ledCurrentlyOn = GREEN_LED_PIN;
194     } else if (ledCurrentlyOn == GREEN_LED_PIN) {
195       digitalWrite(GREEN_LED_PIN, LOW);
196       digitalWrite(BLUE_LED_PIN, HIGH);
197       ledColor = "azul";
198       ledCurrentlyOn = BLUE_LED_PIN;
199     } else if (ledCurrentlyOn == BLUE_LED_PIN) {
200       digitalWrite(BLUE_LED_PIN, LOW);
201       ledColor = "nenhuma";
202       ledCurrentlyOn = 0;
203     }
204     saveJSON();
205     delay(250);
206   }
```

```
207   if (buttonState == HIGH) {
208     buttonPressed = false;
209   }
210 }
```

5. Resultados

A montagem e implementação do código salvar dados no arquivo JSON do ESP32 com a biblioteca LittleFS foi concluída de forma bem sucedida. A seguir, pode-se visualizar a forma do circuito no final da prática, assim como uma demonstração do seu funcionamento.

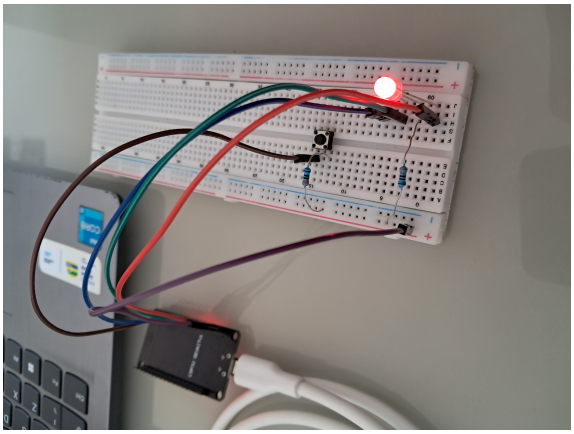

Figure 1. Circuito que muda o estado de um LED RGB por meio de um botão e salva as informações de estado dentro de arquivo JSON do ESP32

O vídeo que demonstra o circuito em funcionamento está disponível em: Resultado da atividade prática 07

6. Conclusão

Após a finalização dessa atividade prática em laboratório, foi possível projetar o circuito e desenvolver o código que muda o estado de um LED RGB com o pressionamento de um botão e armazena as informações do LED dentro de um arquivo JSON. Além disso, o ESP32 é capaz de captar as credenciais de redes WiFi salvas no arquivo e então se conectar em uma delas, Para isso, foi necessário utilizar a biblioteca ArduinoJSON disponibilizado pelo PlatformIO para manipular dados de arquivos JSON. Assim, o objetivo proposto foi atingido com sucesso.