Realizando atualização Over-The-Air no 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 work describes the implementation of an Over-The-Air (OTA) update mechanism on the ESP32 microcontroller. A circuit with a LED and a buzzer was assembled, and a web server was developed to allow firmware upload through a browser. The code was written in C++ using PlatformIO in Visual Studio Code. The activity was successfully completed, demonstrating the feasibility of remote firmware updates on the ESP32.*

Resumo. *Este trabalho descreve a implementação de um mecanismo de atualização Over-The-Air (OTA) no microcontrolador ESP32. Para isso, foi montado um circuito com LED e buzzer, além do desenvolvimento de um web server para upload de firmware via navegador. O código foi escrito em C++ com a extensão PlatformIO no Visual Studio Code. A atividade foi concluída com sucesso, demonstrando a viabilidade da atualização remota de firmware no ESP32.*

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 da atividade é realizar uma atualização OTA (Over-The-Air) no ESP32. Para isso, é preciso montar um circuito que mantém o LED aceso e apagado durante um segundo. Em seguida, é preciso desenvolver um código em C++ que forneça uma forma de atualizar o microcontrolador. Por isso, foi criado um web server para realizar upload do novo código que será gravado no ESP32.

O web server consiste em duas páginas HTML que podem ser abertas no navegador por meio do IP da rede WiFi capturado pelo código e impresso no terminal. A primeira tela apresenta um formulário de login que pode ser acessada por meio das credenciais de login ”admin” e senha ”admin”. Com isso, uma nova página com o formulário para inserção do binário será exibida para o usuário.

Tendo isso em vista, foi feita uma modificação no comportamento do LED para acender a luz constantemente e foi feito o build do novo arquivo binário. Então, o arquivo gerado foi submetido na página do web server.

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;
- 1 buzzer;
- 1 resistor;
- 2 cabos jumper macho-fêmea;
- 2 cabos jumper macho-macho.

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 na região de componentes da protoboard.
- Posicionar um pino do resistor na região de alimentação e a outro em paralelo com o pino negativo do LED.
- Ligar o GPIO13 do ESP32 em paralelo com o pino positivo do LED através de um jumper macho-fêmea.
- Posicionar o buzzer na região de componentes da protoboard.
- Ligar o pino positivo do buzzer em paralelo com o pino positivo do LED através de um jumper macho-macho.
- Ligar o pino negativo do buzzer em paralelo com a região de alimentação da protoboard através de um jumper macho-macho.

4. Implementação do código

O código responsável por permitir a atualização Over-The-Air no ESP32 foi desenvolvido com a linguagem C++ no editor de código Visual Studio Code juntamente com sua extensão PlatformIO.

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>

#define LED 13

const char* host = "esp32";
const char* ssid = "NOME_DA_REDE";
const char* password = "SENHA_DA_REDE";

int contador_ms = 0;

WebServer server(80);

const char* loginIndex =
  "<form name='loginForm'>"
```

```
    "<table width='20%' bgcolor='A09F9F' align='center'>"
      "<tr>"
        "<td colspan=2>"
          "<center><font size=4><b>ESP32 - identifique-se</b></font></center>"
          "<br>"
        "</td>"
        "<br><br>"
      "</tr>"
      "<tr>"
        "<td>Login:</td>"
        "<td><input type='text' size=25 name='userid'><br></td>"
      "</tr>"
      "<br><br>"
      "<tr>"
        "<td>Senha:</td>"
        "<td><input type='Password' size=25 name='pwd'><br></td>"
        "<br><br>"
      "</tr>"
      "<tr>"
          "<td><input type='submit' onclick='check(this.form)' value='Identificar'></td>"
      "</tr>"
    "</table>"
  "</form>"
  "<script>"
    "function check(form) {"
      "if(form.userid.value=='admin' && form.pwd.value=='admin'){"
        "window.open('/serverIndex')"
      "}"
      "else {"
        "alert('Login ou senha invalidos')"
      "}"
    "}"
"</script>";

const char* serverIndex =
  "<script src='https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script>"
  "<form method='POST' action='#' enctype='multipart/form-data' id='upload_form'>"
    "<input type='file' name='update'>"
    "<input type='submit' value='Update'>"
  "</form>"
  "<div id='prg'>Progresso: 0%</div>"
  "<script>"
    "$('form').submit(function(e) {"
```

```
      "e.preventDefault();"
      "var form = $('#upload_form')[0];"
      "var data = new FormData(form);"
      " $.ajax({"
        "url: '/update',"
        "type: 'POST',"
        "data: data,"
        "contentType: false,"
        "processData:false,"
        "xhr: function() {"
          "var xhr = new window.XMLHttpRequest();"
          "xhr.upload.addEventListener('progress', function(evt)
              {"
            "if (evt.lengthComputable) {"
              "var per = evt.loaded / evt.total;"
              "$('#prg').html('Progresso: ' + Math.round(per
                 *100) + '%');"
            "}"
          "}, false);"
          "return xhr;"
        "},"
        "success:function(d, s) {"
          "console.log('Sucesso!')"
        "},"
        "error: function (a, b, c) {}"
      "});"
    "});"
  "</script>";

void setup(void) {
  Serial.begin(9600);

  pinMode(LED, OUTPUT);

  WiFi.begin(ssid, password);
  Serial.println("");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.print("Conectado a rede wi-fi ");
  Serial.println(ssid);
  Serial.print("IP obtido: ");
  Serial.println(WiFi.localIP());

  if (!MDNS.begin(host)) {
```

```
109     Serial.println("Erro ao configurar mDNS. O ESP32 vai
            reiniciar em 1s...");
110     delay(1000);
111     ESP.restart();
112   }
113
114   Serial.println("mDNS configurado e inicializado;");
115
116   server.on("/", HTTP_GET, []() {
117     server.sendHeader("Connection", "close");
118     server.send(200, "text/html", loginIndex);
119   });
120
121   server.on("/serverIndex", HTTP_GET, []() {
122     server.sendHeader("Connection", "close");
123     server.send(200, "text/html", serverIndex);
124   });
125
126   server.on("/update", HTTP_POST, []() {
127     server.sendHeader("Connection", "close");
128     server.send(200, "text/plain", (Update.hasError()) ? "FAIL"
            : "OK");
129     ESP.restart();
130   }, []() {
131     HTTPUpload& upload = server.upload();
132
133     if (upload.status == UPLOAD_FILE_START) {
134       Serial.printf("Update: %s\n", upload.filename.c_str());
135       if (!Update.begin(UPDATE_SIZE_UNKNOWN)) {
136         Update.printError(Serial);
137       }
138     }
139     else if (upload.status == UPLOAD_FILE_WRITE) {
140       if (Update.write(upload.buf, upload.currentSize) != upload
              .currentSize) {
141         Update.printError(Serial);
142       }
143     }
144     else if (upload.status == UPLOAD_FILE_END) {
145       if (Update.end(true)) {
146         Serial.printf("Sucesso no update de firmware: %u\
                nReiniciando ESP32...\n", upload.totalSize);
147       }
148       else {
149         Update.printError(Serial);
150       }
151     }
152   });
153   server.begin();
```

```
}

void loop() {
  server.handleClient();
  delay(1);
  contador_ms++;
  if (contador_ms >= 1000) {
    digitalWrite(LED, HIGH);
    delay(1000);
    digitalWrite(LED, LOW);
    delay(1000);
    contador_ms = 0;
  }
}
```

5. Resultados

A montagem e implementação do código para realizar atualização OTA no microcontrolador ESP32 foi concluída de forma bem sucedida. A seguir, pode-se visualizar a forma do circuito no final da prática, assim como os testes realizados e uma demonstração do seu funcionamento.

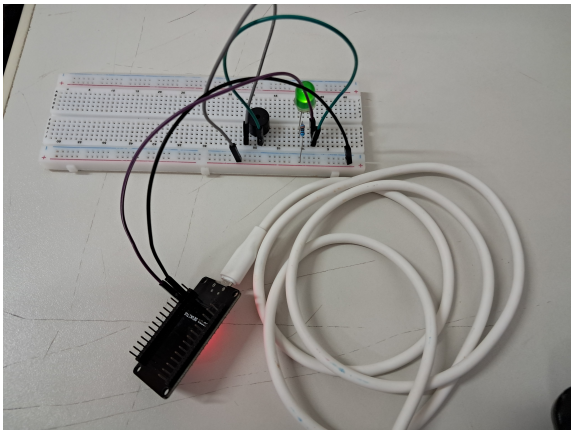

Figure 1. Circuito montado com um LED verde que tem seu comportamento alterado após a atualização OTA

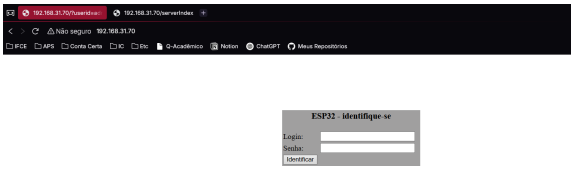

Figure 2. Tela de login para realizar atualização OTA no ESP32

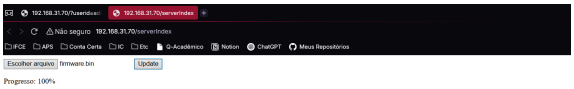

Figure 3. Tela de inserção do arquivo binário para atualização OTA no ESP32

O vídeo que demonstra o circuito em funcionamento está disponível em: Resultado da atividade prática 11

6. Conclusão

Após a finalização dessa atividade prática em laboratório, foi possível projetar o circuito e desenvolver o código que permite fazer atualização OTA no ESP32. Isso foi possível com o uso de um microcontrolador ESP32 e alguns componentes eletrônicos como uma protoboard, um LED verde, um buzzer, um resistor e alguns cabos jumper. Assim, o objetivo proposto foi atingido com sucesso.