Modul 4

Pengenalan ESP32 dan Pemograman ESP32 ke Saklar dan Serial Monitor dan OTA

3.1 Tujuan

Setelah mengikuti praktikum Modul 4, mahasiswa diharapkan dapat:

1. Mengetahui dan Memahami dasar-dasar ESP32 dan fitur-fiturnya.
2. Mengetahui dan menggunakan ESP32 untuk mengendalikan saklar.
3. Mengetahui dan mengimplementasikan komunikasi dengan Serial Monitor.
4. Mengetahui dan cara menggunakan OTA untuk pembaruan firmware.

3.2 Alat dan Bahan

1. ESP32 Development Board
2. Breadboard
3. Kabel jumper
4. Komputer dengan Arduino IDE terinstal
5. Koneksi internet (untuk OTA)
6. Saklar (push button)
7. Resistor (10kΩ untuk pull-down)
8. LED (opsional untuk indikator)
9. Breadboard dan kabel jumper

3.3 Dasar Teori Praktikum Modul 4

Pengenalan Konsep

- **Saklar**: Digunakan untuk menghidupkan atau mematikan sesuatu, dalam hal ini, kita akan menggunakan saklar untuk mengendalikan LED.
- **Serial Monitor**: Fitur dalam Arduino IDE yang memungkinkan pengguna untuk berkomunikasi dengan ESP32 melalui port serial.
- **OTA**: Memungkinkan Anda mengunggah kode ke ESP32 melalui jaringan Wi-Fi, tanpa perlu terhubung secara fisik ke komputer.

3.3.1 Saklar

Saklar adalah sebuah alat elektronik yang memiliki fungsi utama untuk menghubungkan dan memutuskan arus listrik. Selain fungsi utama tersebut, alat elektronik ini juga bisa digunakan untuk memindahkan arus listrik dari sebuah konduktor ke konduktor lainnya.

Alat elektronik ini pertama kali ditemukan dan dibuat oleh John Henry Holmes pada tahun 1884. Kemudian, seiring dengan berkembangnya teknologi dan bertambahnya zaman, semakin banyak perusahaan teknologi yang mengembangkan alat elektronik ini hingga sekarang.

3.3.2 Serial Monitor

Sesuai nama nya, Serial Monitor adalah sebuah feature yang dimiliki oleh software Arduino IDE untuk memonitor atau memantau data yang tersedia atau dikeluarkan oleh microcontroller via komunikasi serial. Dengan Serial Monitor kita bisa mengetahui apa yang dituliskan di port serial dan ditampilkan pada jendela Serial Monitor.

Komunikasi serial pada board UNO dilakukan pada port serial yang terletak pada pin 0 dan pin 1, atau melalui USB. Board UNO, Nano, ProMini, Lilypad memiliki hanya 1 buah serial yang terletak pada pin 0 dan 1 dan digunakan juga sebagai komunikasi melalui USB,Untuk mencoba Serial Monitor kali ini, wiring rangkaian yang digunakan hanya board Uno yang terkoneksi ke komputer via USB, tanpa komponen atau modul apapun.

3.3.3 OTA (Over-The-Air)

OTA, atau Over-The-Air, dalam konteks pemrograman Arduino, merujuk pada metode untuk mengunggah (upload) kode atau firmware ke perangkat Arduino tanpa perlu menyambungkannya secara fisik ke komputer. Ini sangat berguna untuk perangkat IoT (Internet of Things) yang mungkin sulit diakses secara langsung setelah dipasang.

Cara Kerja OTA :

1. Koneksi Jaringan: Perangkat Arduino, seperti ESP8266 atau ESP32, terhubung ke jaringan Wi-Fi.
2. Server Pembaruan: Firmware baru dihosting pada server atau dapat diunduh langsung dari komputer.
3. Proses Pembaruan: Dengan menggunakan pustaka tertentu di Arduino IDE, pengguna dapat mengunggah kode baru ke perangkat melalui jaringan.

ESP32 S3

ESP32-S3 adalah mikroprosesor sistem pada chip (SoC) yang dikembangkan oleh **Espressif Systems**. Ini adalah bagian dari keluarga **ESP32** yang terkenal dengan kemampuan Wi-Fi dan Bluetooth, yang biasa digunakan dalam proyek IoT (Internet of Things). ESP32-S3 menawarkan peningkatan kinerja dibandingkan model sebelumnya dan beberapa fitur canggih.

Kegunaan:

Proyek rumah pintar (smart home).

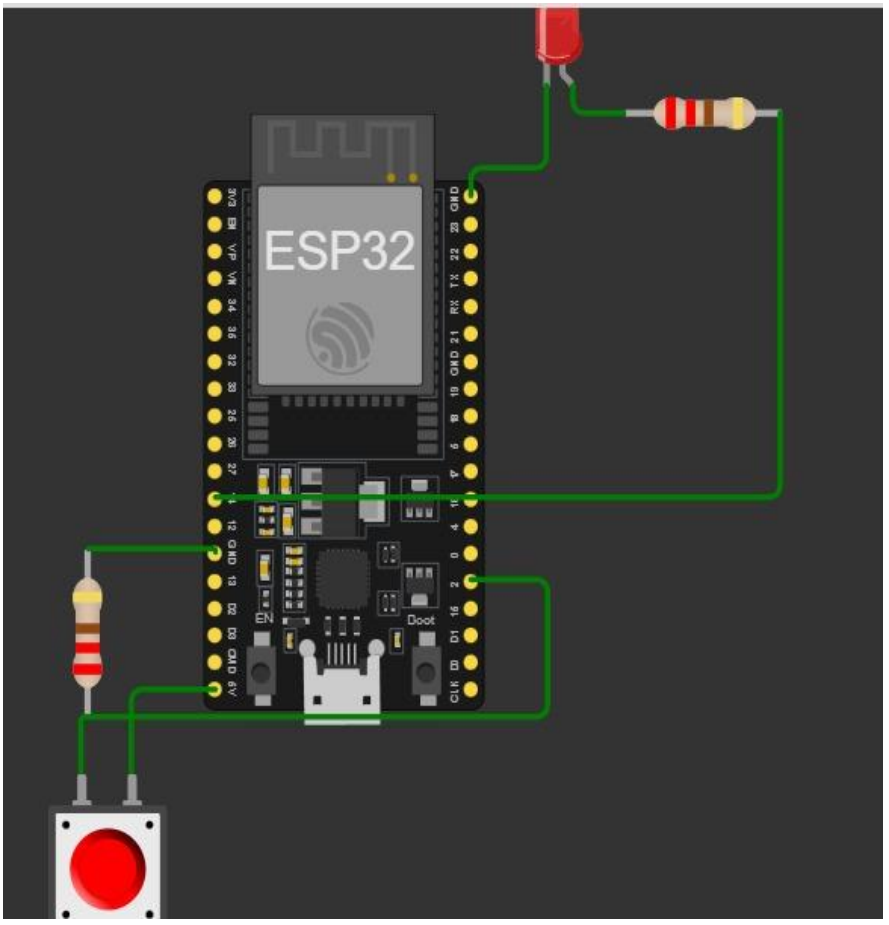
- Sistem otomatisasi industri.
- Perangkat wearable yang menggunakan pengenalan suara atau gambar.
- Sistem monitoring lingkungan atau sensor jaringan

3.4 Modul Praktikum

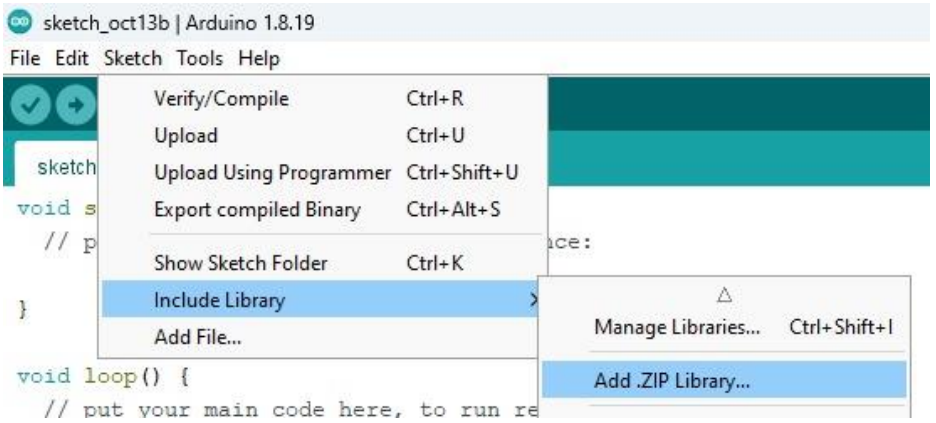
Mengendalikan LED dengan Komunikasi OTA

- **Arduino Ide**

1. Siapkan alat dan bahan
2. Rangkai ESP32 S3 menggunakan breakout Board beserta komponennya seperti dibawah ini



3. Buka Software Arduino IDE pada laptop yang telah terinstal.
4. Masuk pada menu *sketch> include library> add ZIP library* > lalu pilih file yang telah disediakan.



5. Selanjutkanya lakukan seperti dibawah

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

const char* host = "esp32";
const char* ssid = "whatursign";
const char* password = "gdaymate";

WebServer server(80);

/*
 * Login page
 */

const char* loginIndex =
 "<form name='loginForm'>"
    "<table width='20%' bgcolor='A09F9F' align='center'>"
        "<tr>"
            "<td colspan=2>"
                "<center><font size=4><b>ESP32 Login Page</b></font></center>"
                "<br>"
            "</td>"
            "<br>"
            "<br>"
```

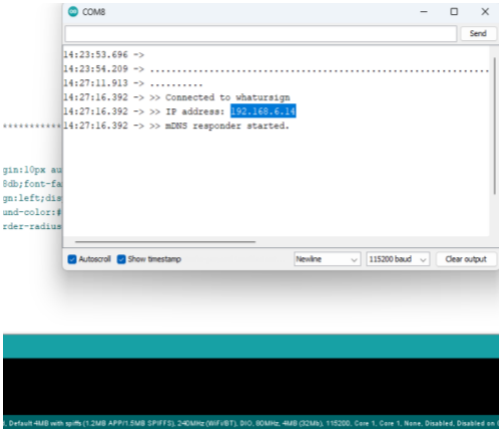
```
                    "<br>"
                    "<br>"
                "</tr>"
                "<tr>"
                     "<td>Username:</td>"
                     "<td><input type='text' size=25 name='userid'><br></td>"
                "</tr>"
                "<br>"
                "<br>"
                "<tr>"
                    "<td>Password:</td>"
                    "<td><input type='Password' size=25 name='pwd'><br></td>"
                    "<br>"
                    "<br>"
                "</tr>"
                "<tr>"
                    "<td><input type='submit' onclick='check(this.form)' value='Login'></td>"
                "</tr>"
            "</table>"
        "</form>"
        "<script>"
            "function check(form)"
            "{"
            "if(form.userid.value=='admin' && form.pwd.value=='admin')"
            "{"
            "window.open('/serverIndex')"



    "window.open('/serverIndex')"
    "}"
    "else"
    "{"
    " alert('Error Password or Username')/*displays error message*/"
    "}"
    "}"
"</script>";

/*
 * Server Index Page
 */

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'>progress: 0%</div>"
 "<script>"
  "$('form').submit(function(e){"
  "e.preventDefault();"
  "var form = $('#upload_form')[0];"
  "var data = new FormData(form);"
```

```
"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('progress: ' + Math.round(per*100) + '%');"
"}"
"}, false);"
"return xhr;"
"},"
"success:function(d, s) {"
"console.log('success!')"
"},"
"error: function (a, b, c) {"
"}"
"});"
"});"
"</script>";
```

```
void setup(void) {
  Serial.begin(115200);

  // Connect to WiFi network
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  /*use mdns for host name resolution*/
  if (!MDNS.begin(host)) { //http://esp32.local
    Serial.println("Error setting up MDNS responder!");
    while (1) {
      delay(1000);
```

```
    delay(1000);
  }
}
Serial.println("mDNS responder started");
/*return index page which is stored in serverIndex */
server.on("/", HTTP_GET, []() {
  server.sendHeader("Connection", "close");
  server.send(200, "text/html", loginIndex);
});
server.on("/serverIndex", HTTP_GET, []() {
  server.sendHeader("Connection", "close");
  server.send(200, "text/html", serverIndex);
});
/*handling uploading firmware file */
server.on("/update", HTTP_POST, []() {
  server.sendHeader("Connection", "close");
  server.send(200, "text/plain", (Update.hasError()) ? "FAIL" : "OK");
  ESP.restart();
}, []() {
  HTTPUpload& upload = server.upload();
  if (upload.status == UPLOAD_FILE_START) {
    Serial.printf("Update: %s\n", upload.filename.c_str());
    if (!Update.begin(UPDATE_SIZE_UNKNOWN)) { //start with max available size
      Update.printError(Serial);
    }
  } else if (upload.status == UPLOAD_FILE_WRITE) {



   } else if (upload.status == UPLOAD_FILE_WRITE) {
    /* flashing firmware to ESP*/
    if (Update.write(upload.buf, upload.currentSize) != upload.currentSize) {
      Update.printError(Serial);
    }
   } else if (upload.status == UPLOAD_FILE_END) {
    if (Update.end(true)) { //true to set the size to the current progress
      Serial.printf("Update Success: %u\nRebooting...\n", upload.totalSize);
    } else {
      Update.printError(Serial);
    }
   }
 });
 server.begin();
}

void loop(void) {
  server.handleClient();
  delay(1);
}
```

6. Sebelum menjalankan program ubah SSID dan Paswword sesuai dengan hospot yang terkoneksi dengan laptop kalian, lalu *verify* dan *upload* program.
7. Pastikan file sudah disimpan pada folder.
8. Lalu ke serial monitor, copy Ip yang terdapat pada serial monitor.



9. Lalu lakukan program kedua, seperti di bawah ini. Sebelum masuk ke program lepas ESP yang terhubung.

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

const char* host = "esp32";
const char* ssid = "whatursign";
const char* password = "gdaymate";

//variabls to blink without delay:
const int led = 2;
unsigned long previousMillis = 0;        // will store last
const long interval = 1000;           // interval at which
int ledState = LOW;             // ledState used to set the

WebServer server(80);

/*
 * Login page
 */

const char* loginIndex =
 "<form name='loginForm'>"
    "<table width='20%' bgcolor='A09F9F' align='center'>"
```

```
        "<table width='20%' bgcolor='A09F9F' align='center'>"
            "<tr>"
                "<td colspan=2>"
                    "<center><font size=4><b>ESP32 Login Page</b></font></center>"
                    "<br>"
                "</td>"
                "<br>"
                "<br>"
            "</tr>"
            "<td>Username:</td>"
            "<td><input type='text' size=25 name='userid'><br></td>"
            "</tr>"
            "<br>"
            "<br>"
            "<tr>"
                "<td>Password:</td>"
                "<td><input type='Password' size=25 name='pwd'><br></td>"
                "<br>"
                "<br>"
            "</tr>"
            "<tr>"
                "<td><input type='submit' onclick='check(this.form)' value='Login'></td>"
            "</tr>"
        "</table>"
    "</form>"
    "<script>"


    "function check(form)"
    "{"
    "if(form.userid.value=='admin' && form.pwd.value=='admin')"
    "{"
    "window.open('/serverIndex')"
    "}"
    "else"
    "{"
    " alert('Error Password or Username')/*displays error message*/"
    "}"
    "}"
"</script>";

/*
 * Server Index Page
 */

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'>progress: 0%</div>"
 "<script>"
  "$('form').submit(function(e){"
```

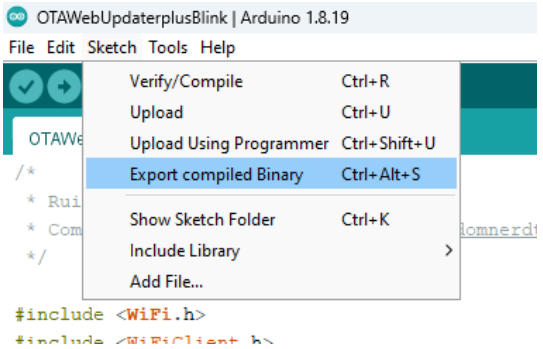
```
  "e.preventDefault();"
  "var form = $('#upload_form')[0];"
  "var data = new FormData(form);"
  " $.ajax({"
  "url: '/update',"
  "type: 'POST',"
  "data: data,"
  "contentType: false,"
  "processData:false,"
  "xhr: function() {"
  "var xhr = new window.XMLHttpRequest();"
  "xhr.upload.addEventListener('progress',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

```
 "</script>";

/*
 * setup function
 */
void setup(void) {
  pinMode(led, OUTPUT);

  Serial.begin(115200);

  // Connect to WiFi network
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  /*use mdns for host name resolution*/
```

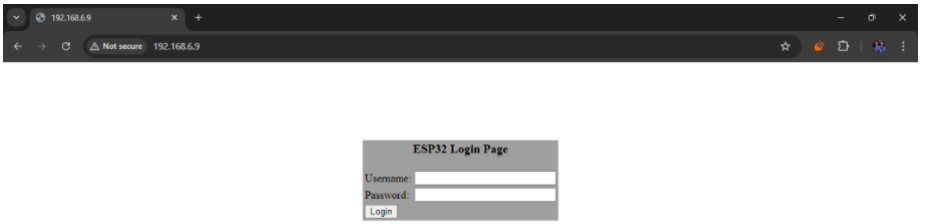
```
  /*use mdns for host name resolution*/
  if (!MDNS.begin(host)) { //http://esp32.local
    Serial.println("Error setting up MDNS responder!");
    while (1) {
      delay(1000);
    }
  }
  Serial.println("mDNS responder started");
  /*return index page which is stored in serverIndex */
  server.on("/", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", loginIndex);
  });
  server.on("/serverIndex", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", serverIndex);
  });
  /*handling uploading firmware file */
  server.on("/update", HTTP_POST, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/plain", (Update.hasError()) ? "FAIL" : "OK");
    ESP.restart();
  }, []() {
    HTTPUpload& upload = server.upload();
    if (upload.status == UPLOAD_FILE_START) {
      Serial.printf("Update: %s\n", upload.filename.c_str());
      if (!Update.begin(UPDATE_SIZE_UNKNOWN)) { //start with max available size
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_WRITE) {
      /* flashing firmware to ESP*/
      if (Update.write(upload.buf, upload.currentSize) != upload.currentSize) {
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_END) {
      if (Update.end(true)) { //true to set the size to the current progress
        Serial.printf("Update Success: %u\nRebooting...\n", upload.totalSize);
      } else {
        Update.printError(Serial);
      }
    }
  });
  server.begin();
}

void loop(void) {
  server.handleClient();
  delay(1);

  //loop to blink without delay
  unsigned long currentMillis = millis();
```

```
void loop(void) {
  server.handleClient();
  delay(1);

  //loop to blink without delay
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    ledState = not(ledState);

    // set the LED with the ledState of the variable:
    digitalWrite(led, ledState);
  }
}
```

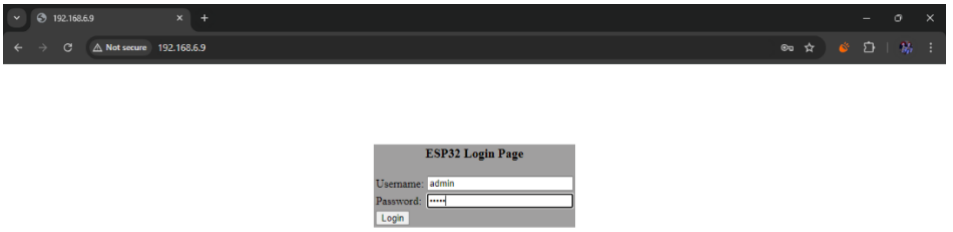
10. Selanjutnya lakukan verify, dan ke menu sketch> export compiled binary.



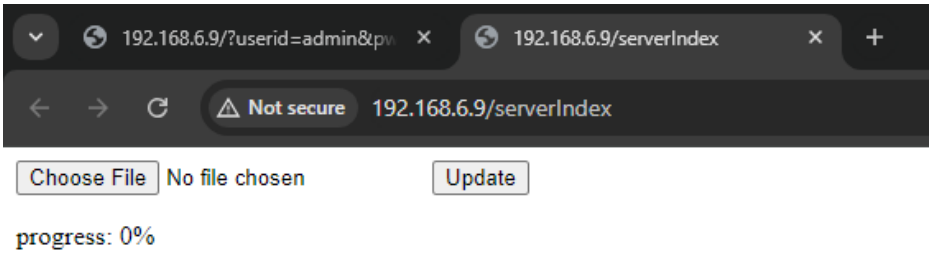
11. Buka chrome, lalu paste Ip yang telah kalian salin pada serial monitor, setelah itu akan muncul tampilan web ESP32.



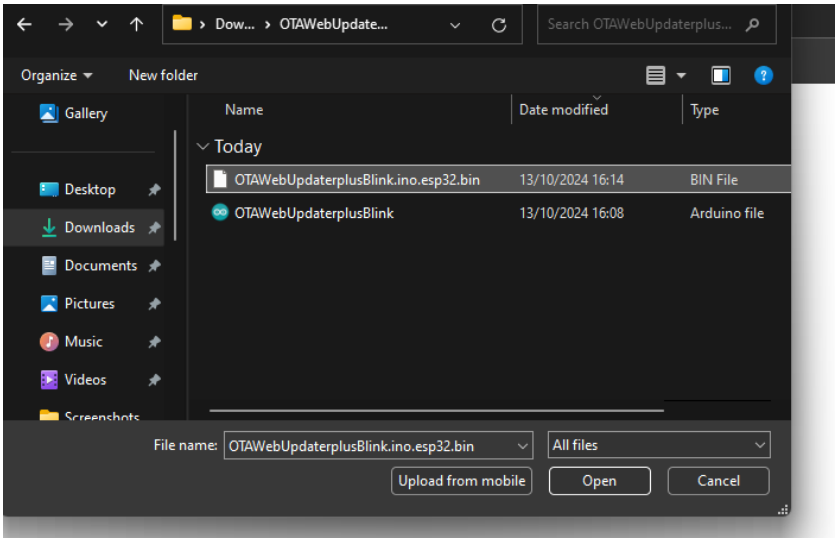
12. Masukan Username “admin” dan Password “admin”, dan klik Login.



13. Maka tampilan akan seperti di bawah ini , lalu pilih *choose file*.



14. Selanjutnya buka *file*> lalu cari file dengan *format .bin*> seperti gambar dibawah ini.



15. Kemudian hasil akan keluar pada rangkaian ESP32 S3, dimana LED akan menyala “on” dan “off”

16. Selanjutnya perhatikan langkah yang akan diberikan oleh Asisten.

SELAMAT MENGERJAKAN v

SELAMAT MENGERJAKAN ☺