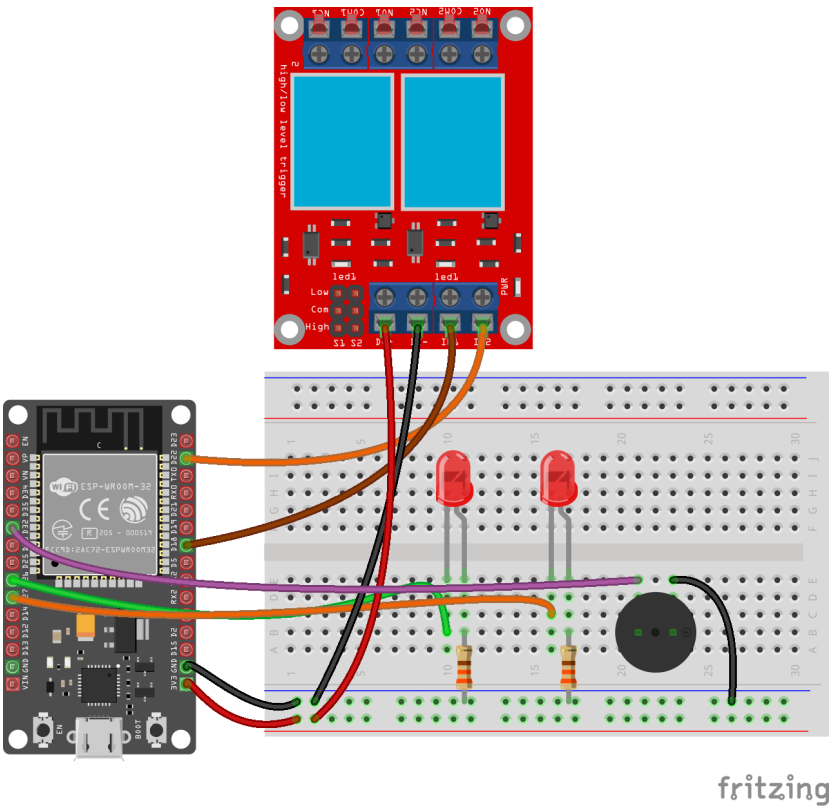
Membuat webserver dengan ESP-32

Webserver ESP 32

Web server di ESP32 memungkinkan ESP32 untuk terhubung ke internet dan melayani halaman web. ESP32 dapat bertindak sebagai router mini atau terhubung dengan jaringan yang ada. Pengguna dapat memprogram web server menggunakan **Arduino IDE** , termasuk library seperti **WebServer.h** dan **HTML** untuk membangun halaman web. Web server ESP32 memiliki berbagai kegunaan, seperti mengendalikan perangkat IoT, mencatat data, dan konfigurasi.

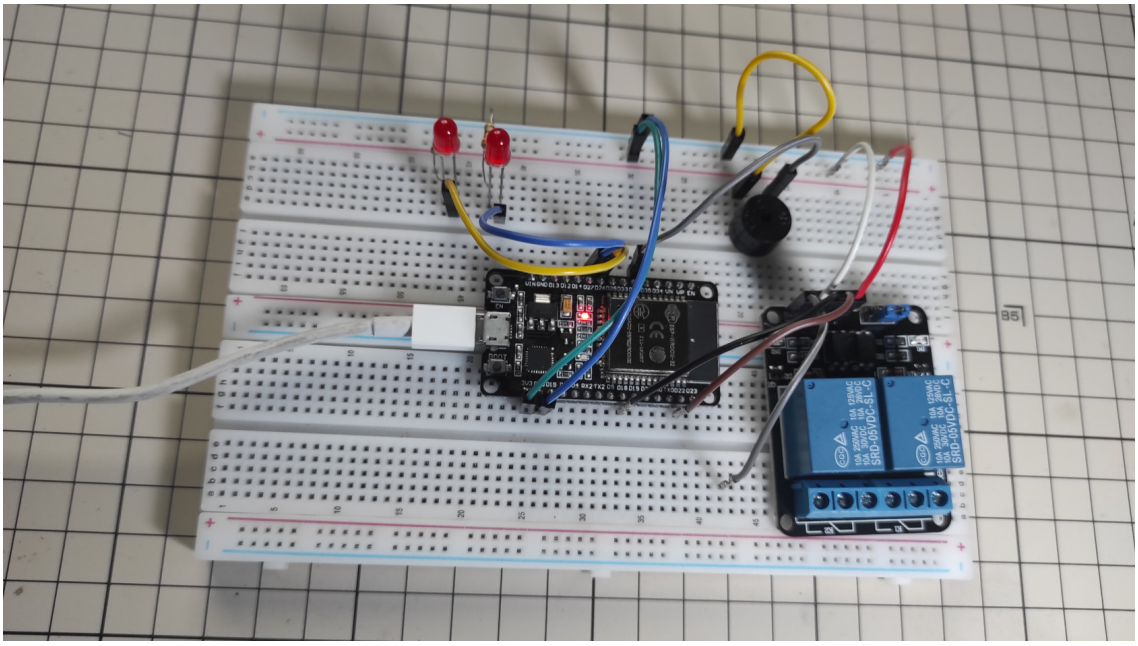
> EL 5057 | Sistem Penginderaan

Pada percobaan kali ini kita akan membuat website yang ditempatkan didalam webserver ESP32, website ini menggunakan html dan css sederhana untuk menampilkan button. Button ini digunakan untuk mengontrol LED, Buzzer dan Sebuah Relay dual channel. Skema rangkaiannya dapat dilihat dalam gamber berikut.



Daftar komponen yang harus kita siapkan adalah sebagai berikut.

No.	Daftar Komponen
1.	ESP 32 Development Board
2.	Relay dual channel
3.	Buzzer
4.	Breadboard
5.	Jumper wire
6.	LED
7.	Resistor 330 Ohm



Modifikasi sketch dari https://randomnerdtutotial.com.

```
#include <WiFi.h>

// Replace with your network credentials
const char* ssid = "iot";
const char* password = "";

// Set web server port number to 80
WiFiServer server(80);
```

```
// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String output26State = "off";
String output27State = "off";
String output32State = "off";
String output18State = "off";
String output22State = "off";

// Assign output variables to GPIO pins
const int output26 = 26;
const int output27 = 27;
const int output32 = 32;
const int output18 = 18;
const int output22 = 22;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(output26, OUTPUT);
  pinMode(output27, OUTPUT);
  pinMode(output32, OUTPUT);
  pinMode(output22, OUTPUT);
  pinMode(output18, OUTPUT);

  // Set outputs to LOW
  digitalWrite(output26, LOW);
  digitalWrite(output27, LOW);
  digitalWrite(output32, LOW);
  digitalWrite(output22, HIGH);
  digitalWrite(output18, HIGH);
```

```
  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients

  if (client) {                             // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    while (client.connected() && currentTime - previousTime <= timeoutTime) {  // loop while the client's
connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
```

```
client.println();

// turns the GPIOs on and off
if (header.indexOf("GET /26/on") >= 0) {
  Serial.println("GPIO 26 on");
  output26State = "on";
  digitalWrite(output26, HIGH);
} else if (header.indexOf("GET /26/off") >= 0) {
  Serial.println("GPIO 26 off");
  output26State = "off";
  digitalWrite(output26, LOW);
} else if (header.indexOf("GET /27/on") >= 0) {
  Serial.println("GPIO 27 on");
  output27State = "on";
  digitalWrite(output27, HIGH);
} else if (header.indexOf("GET /27/off") >= 0) {
  Serial.println("GPIO 27 off");
  output27State = "off";
  digitalWrite(output27, LOW);
} else if (header.indexOf("GET /32/off") >= 0) {
  Serial.println("GPIO 32 off");
  output32State = "off";
  digitalWrite(output32, LOW);
} else if (header.indexOf("GET /32/on") >= 0) {
  Serial.println("GPIO 32 on");
  output32State = "on";
  digitalWrite(output32, HIGH);
} else if (header.indexOf("GET /22/off") >= 0) {
  // Relay
  Serial.println("GPIO 22 off");
  output22State = "off";
  digitalWrite(output22, HIGH);
} else if (header.indexOf("GET /22/on") >= 0) {
  Serial.println("GPIO 22 on");
  output22State = "on";
  digitalWrite(output22, LOW);
} else if (header.indexOf("GET /18/off") >= 0) {
  Serial.println("GPIO 18 off");
  output18State = "off";
  digitalWrite(output18, HIGH);
```

```
            } else if (header.indexOf("GET /18/on") >= 0) {
              // Relay
              Serial.println("GPIO 18 on");
              output18State = "on";
              digitalWrite(output18, LOW);
            }

            // Konfigurasi CSS
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align:
center;}");
            client.println("h2 { margin-bottom: 0px; margin-top: 0px;}");
            client.println("img {height: 98px; width: 100px; margin-top: 35px;}");
            client.println("small { margin-top: 0px; }");
            client.println(".button { background-color: #4CAF50; border: none; border-radius: 5%; box-shadow: 3px
4px 6px rgba(0, 0, 0, 0.3); color: white; padding: 12px 40px;");
            client.println("text-decoration: none; font-size: 26px; margin: 2px; cursor: pointer;}");
            client.println(".button2 {background-color: #555555;}");
            client.println(".button3 {background-color: #555555;}</style></head>");


            // Judul Website
            client.println("<a href=\"https://imgbb.com/\"><img src=\"https://i.ibb.co/Pt5sbsb/logo-itb-512.png\"
alt=\"logo-itb-512\" border=\"0\"></a>");
            client.println("<body><h2>ESP32 Web Server</h2>");
            client.println("<small>SMART-X ITB</small>");

            // Display current state, and ON/OFF buttons for GPIO 26
            client.println("<p>GPIO 26 - State " + output26State + "</p>");
            // If the output26State is off, it displays the ON button
            if (output26State=="off") {
              client.println("<p><a href=\"/26/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/26/off\"><button class=\"button button2\">OFF</button></a></p>");
            }

            // Display current state, and ON/OFF buttons for GPIO 27
            client.println("<p>GPIO 27 - State " + output27State + "</p>");
```

```
    // If the output27State is off, it displays the ON button
    if (output27State=="off") {
      client.println("<p><a href=\"/27/on\"><button class=\"button\">ON</button></a></p>");
    } else {
      Serial.println("Tes Buzzer Apakah On");
      client.println("<p><a href=\"/27/off\"><button class=\"button button2\">OFF</button></a></p>");
    }

    // Display status buzzer
    client.println("<p>BUZZER : "+ output32State +"</p>");
    if (output32State=="off") {
      client.println("<p><a href=\"/32/on\"><button class=\"button\">ON</button></a></p>");
    } else {
      client.println("<p><a href=\"/32/off\"><button class=\"button button2\">OFF</button></a></p>");
    }
    // Display status Relay 1
    client.println("<p>RELAY 2 : "+ output22State +"</p>");
    if (output22State=="off") {
      client.println("<p><a href=\"/22/on\"><button class=\"button\">ON</button></a></p>");
    } else {
      client.println("<p><a href=\"/22/off\"><button class=\"button button2\">OFF</button></a></p>");
    }
    // Display status Relay 1
    client.println("<p>RELAY 1 : "+ output18State +"</p>");
    if (output18State=="off") {
      client.println("<p><a href=\"/18/on\"><button class=\"button\">ON</button></a></p>");
    } else {
      client.println("<p><a href=\"/18/off\"><button class=\"button button2\">OFF</button></a></p>");
    }

    client.println("</body></html>");

    // The HTTP response ends with another blank line
    client.println();
    // Break out of the while loop
    break;
  } else { // if you got a newline, then clear currentLine
    currentLine = "";
  }
} else if (c != '\r') {  // if you got anything else but a carriage return character,
```

```
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

Hasil percobaan kita secara lengkap dapat dilihat dalam video berikut ini.

https://www.youtube.com/embed/oOTVH4n_TnM

Revision #8
Created 29 March 2024 00:45:58 by Sandi Wibowo
Updated 29 March 2024 09:05:43 by Sandi Wibowo