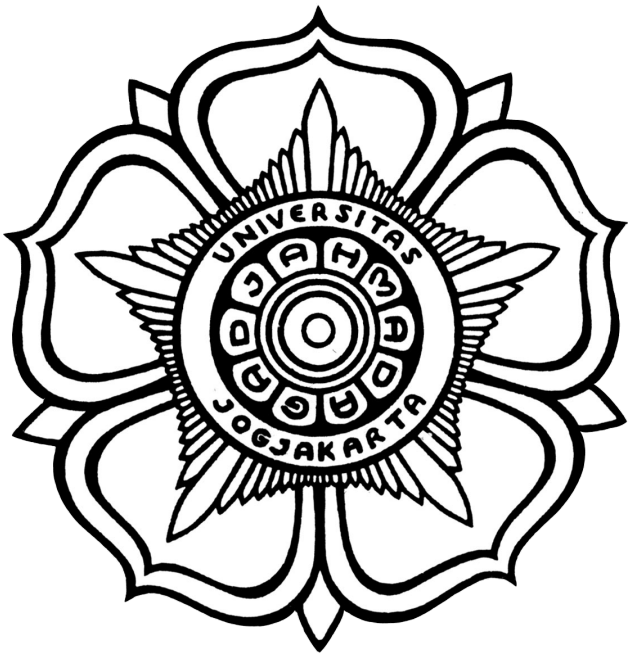
LAPORAN PRAKTIKUM

ELEKTRONIKA MESIN LISTRIK DAN TEKNIK KENDALI

"*ESP32 Physical Button Web Server Kendali LED + Oled Display*"

Dosen Pengampu: Irfan Bahiuddin, ST, M.Phil., Ph.D.



Disusun Oleh:

Kelompok 4

Chaesar Syaefuddin (19/441195/SV/16547)

Yeyen Karunia (19/441215/SV/16567)

Kelas: ARM 2

DEPARTEMEN TEKNIK MESIN

SEKOLAH VOKASI

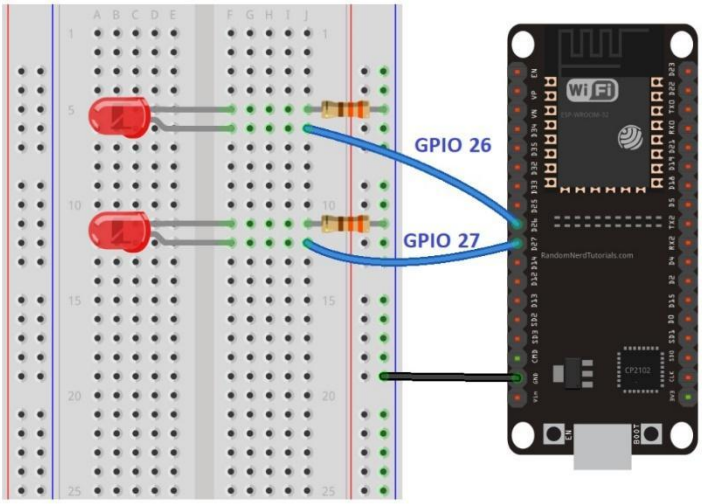
UNIVERSITAS GADJAH MADA

YOGYAKARTA

2022

BAB I
DESKRIPSI KASUS

Kasus pada tugas 1 ini adalah mengaktifkan dan menonaktifkan LED menggunakan Inputan button dan website yang kemudian ditampilkan pada modul OLED dan website itu sendiri.

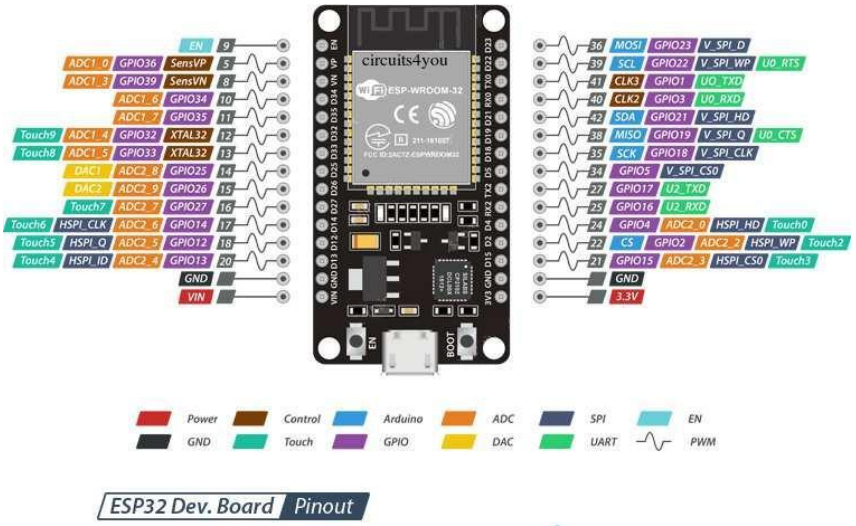


Gambar 1. Rancangan Rangkaian Elektronika

BAB II
KOMPONEN YANG DIGUNAKAN

2.1 Mikrokontroler ESP32

ESP 32 adalah mikrokontroler yang dikenalkan oleh Espressif System merupakan penerus dari mikrokontroler ESP8266. Pada mikrokontroler ini sudah tersedia modul WiFi dalam chip sehingga sangat mendukung untuk membuat sistem aplikasi Internet of Things. Terlihat pada gambar dibawah merupakan pin out dari ESP32. Pin tersebut dapat dijadikan input atau output untuk menyalakan LCD, lampu, bahkan untuk menggerakan motor DC. Pada mikrokontroler ini sudah tersedia modul wifi dan bluetooth sehingga sangat mendukung untuk membuat sistem aplikasi Internet of Things. Memiliki 18 ADC (Analog Digital Converter), 2 DAC, 16 PWM, 10 Sensor sentuh, 2 jalur antarmuka UART,pin antarmuka I2C, I2S, dan SPI.



Gambar 2. Mikrokontroller Esp32

ESP32 menggunakan prosesor dual core yang berjalan di instruksi Xtensa LX16 [3], ESP32 memiliki spesifikasi seperti yang ditampilkan pada tabel 1.

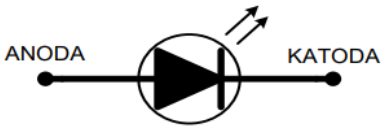
Tabel 1. Spesifikasi Esp32

Atribut	**Detail**
CPU	Tensilica Xtensa LX6 32bit Dual-Core di 160/240MHz
SRAM	520KB
FLASH	2MB (max. 64MB)
Tegangan	2.2V sampai 3.6V
Arus Kerja	Rata-rata 80mA
Dapat diprogram	Ya (C, C++, Python, Lua, dll)
Open Source	Ya
Konektivitas	
Wi-Fi	802.11 b/g/n
Bluetooth	4.2BR/EDR + BLE
UART	3
I/O	
GPIO	32

SPI	4
I2C	2
PWM	8
ADC	18 (12- bit)
DAC	2 (8-bit)

2.2 LED

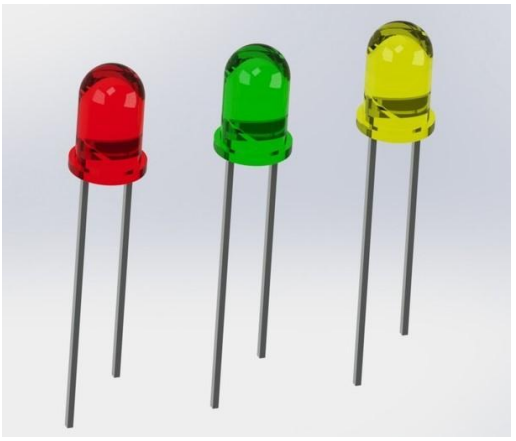
Light Emiting Diode (LED) adalah komponen yang dapat memancarkan cahaya. Struktur LED sama dengan dioda. Untuk mendapatkan pancaran cahaya pada semikonduktor, dopping yang dipakai adalah gallium, arsenic, dan phosporus. Jenis dopping yang berbeda akan menghasilkan warna cahaya yang berbeda. Bentuk LED bermacam-macam, ada yang bulat, persegi empat dan lonjong. Simbol LED terlihat pada gambar 3.



Gambar 3. Simbol LED

LED adalah dioda yang dapat memancarkan cahaya pada saat mendapat arus bias maju (forward bias). LED merupakan salah satu jenis dioda, sehingga hanya akan mengalirkan arus listrik satu arah saja. LED akan memancarkan cahaya apabila diberikan tegangan listrik dengan konfigurasi forward bias. Berbeda dengan dioda pada umumnya, kemampuan mengalirkan arus pada LED cukup rendah yaitu maksimal 20 mA. Apabila LED dialiri arus lebih besar dari 20 mA maka LED akan rusak, sehingga pada rangkaian LED dipasang sebuah resistor sebagai pembatas arus.

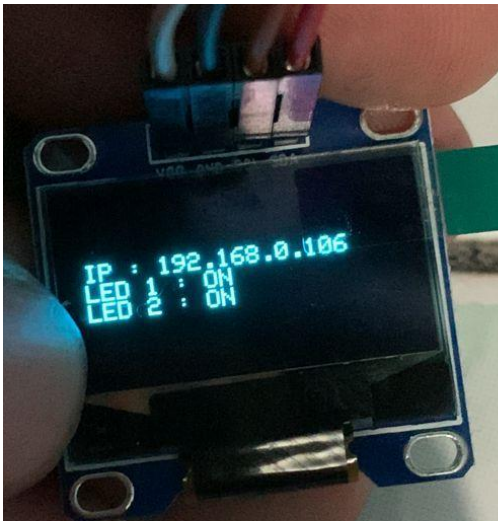
LED memiliki kaki 2 buah seperti dengan dioda yaitu kaki anoda dan kaki katoda. Pada gambar diatas kaki anoda memiliki ciri fisik lebih panjang dari kaki katoda pada saat masih baru, kemudian kaki katoda pada LED ditandai dengan bagian body yang dipapas rata. Pemasangan LED agar dapat menyala adalah dengan memberikan tegangan bias maju yaitu dengan memberikan tegangan positif ke kaki anoda dan tegangan negatif ke kaki katoda. Konsep pembatas arus pada dioda adalah dengan memasangkan resistor secara seri pada salah satu kaki LED.



Gambar 4. LED

2.3 OLED

Oled (*Organic- Light Emitting Diode*) adalah sebuah komponen semikonduktor yang solid seperti halnya komponen LED (*Light Emitting Diode*) yang dibuat dengan menyisipkan beberapa lembar lapisan tipis organik di antara dua konduktor. Jika dialiri arus maka OLED akan menyala. OLED tidak membutuhkan sumber cahaya lain seperti halnya LCD yang membutuhkan *backlight* sebagai sumber cahayanya.



Gambar 5. Oled

2.4 Button

Button merupakan saklar yang paling sering digunakan dikehidupan sehari hari Push button switch (saklar tombol tekan) adalah perangkat / saklar sederhana yang berfungsi untuk menghubungkan atau memutuskan aliran arus listrik dengan sistem kerja tekan unlock (tidak mengunci). Sistem kerja unlock disini berarti saklar akan bekerja sebagai device penghubung atau pemutus aliran arus listrik saat tombol ditekan, dan saat tombol tidak ditekan (dilepas), maka saklar akan kembali pada kondisi normal. Sebagai device penghubung atau pemutus, push button switch hanya memiliki 2 kondisi, yaitu On dan Off (1 dan 0). Istilah On dan Off ini menjadi sangat penting karena semua perangkat listrik yang

memerlukan sumber energi listrik pasti membutuhkan kondisi On dan Off. Button Dapat mendeteksi ketukan, hold down ketika pengguna menekan tombol, atau ketika pengguna melepas tombol. Ketika button mendeteksi salah satu dari hal tersebut, button akan menjalankan perintah.

Gambar 6. Button

BAB III
PEMBAHASAN

A. KODING

PR01_LED_Website_Oled

```
#include <WiFi.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

const char* ssid = "UGM-Hotspot";
const char* password = "";

// Set web server port number to 80
WiFiServer server(8080);

// Variable to store the HTTP request
String header;

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

// Assign output variables to GPIO pins
const int output26 = 26;
const int output27 = 27;
```

PR01_LED_Website_Oled

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


// Current time
unsigned long btnCurrentTime = millis();
// Previous time
unsigned long btnPreviousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long btnTimeoutTime = 2000;

const int buttonPin = 4;
const int buttonPin1 = 2;

int buttonState = 0;
int buttonState1 = 0;

int btnClick = 0;
int btnClick1 = 0;
```

PR01_LED_Website_Oled

```
String LED;
String LED1;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(output26, OUTPUT);
  pinMode(output27, OUTPUT);
  // Set outputs to LOW
  digitalWrite(output26, LOW);
  digitalWrite(output27, LOW);

  pinMode(buttonPin, INPUT);
  pinMode(buttonPin1, INPUT);

  // 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("");
```

PR01_LED_Website_Oled

```
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();


    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);


}

void loop() {
  WiFiClient client = server.available();   // Listen for incoming clients
  buttonState = digitalRead(buttonPin);
  buttonState1 = digitalRead(buttonPin1);
  if (buttonState == LOW && btnClick == 0) {
    btnClick = 1;
    output26State = output26State == "on" ? "off" : "on";
    Serial.println(buttonState + " ," + output26State);
    digitalWrite(output26, output26State == "on" ? HIGH : LOW);
```

PR01_LED_Website_Oled

```
    Serial.println(buttonState + " ," + output26State);
    digitalWrite(output26, output26State == "on" ? HIGH : LOW);
  }
  if (buttonState == HIGH) {
    delay(50);
    btnClick = 0;
  }

  if (buttonState1 == LOW && btnClick1 == 0) {
    btnClick1 = 1;
    output27State = output27State == "on" ? "off" : "on";
    Serial.println(buttonState1 + " ," + output27State);
    digitalWrite(output27, output27State == "on" ? HIGH : LOW);
  }
  if (buttonState1 == HIGH) {
    delay(50);
    btnClick1 = 0;
  }


  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
```

PR01_LED_Website_Oled

```
  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
      Serial.println("c" + c);
      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(header.indexOf("GET /26/on"));
            Serial.println("GPIO 26 on");
            output26State = "on";
            digitalWrite(output26, HIGH);
          } else if (header.indexOf("GET /26/off") >= 0) {
```

PR01_LED_Website_Oled

```
            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);
          }

          // Display the HTML web page
          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:,\">");
          // CSS to style the on/off buttons
          // Feel free to change the background-color and font-size attributes to fit your preferences
          client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
          client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");
          client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
          client.println(".button2 {background-color: #555555;}</style></head>");
```

PR01_LED_Website_Oled

```
// Web Page Heading
client.println("<body><h1>ESP32 Web Server</h1>");
client.println("<p><a href=\"/\"><button class=\"button\" style=\"background-color:#fff; color: #000;border: 1px solid #000!important;border-radius: 6px;\">Refresh</butto
// 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>");
  LED = "ON";

} else {
  client.println("<p><a href=\"/26/off\"><button class=\"button button2\">OFF</button></a></p>");
  LED = "OFF";
}

// 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>");
  LED1 = "ON";
} else {
  client.println("<p><a href=\"/27/off\"><button class=\"button button2\">OFF</button></a></p>");
  LED1 = "OFF";
}
client.println("</body></html>");
```

PR01_LED_Website_Oled

```
            }
            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("");

    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 20);
```

PR01_LED_Website_Oled

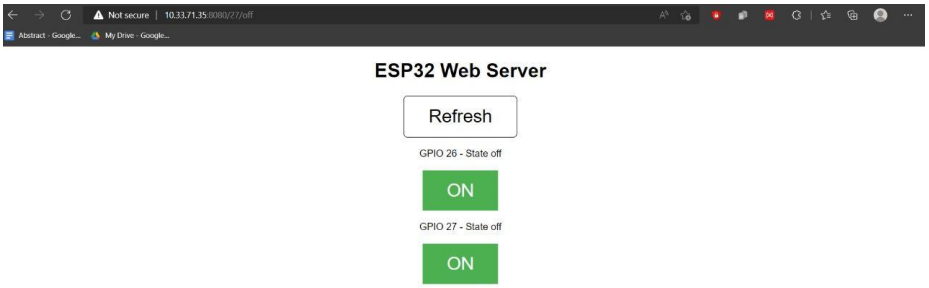
```
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");

    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 20);
    // Display static text
    display.print("IP : ");
    display.println(WiFi.localIP());
    display.print("LED 1 : ");
    display.println(LED);
    display.print("LED 2 : ");
    display.println(LED1);
    display.display();
  }
}
```

B. WEB SERVER

Web Server menjelaskan tentang suatu device/komputer/server yang menerima informasi, mengolah dan menyimpannya. Web server juga dapat menampilkan informasi ke user. Misal user mengetik alamat IP web server pada browser, kemudian web server menampilkan respon dengan menampilkan informasi berupa kode respon. Komunikasi antara web server dengan user menggunakan protokol HTTP (*Hypertext Transfer*

Protocol). ESP32 bisa difungsikan sebagai web server. Misal web server pada ESP32 mempunyai IP 192.167.123.4, user meminta koneksi ke web server dengan akses wifi yang dipancarkan oleh ESP32 atau wifi yang sudah ada. Kemudian ESP32 merespon dengan mengirim kode 200 yang berarti koneksi berhasil. Kode yang dikirim disertai dengan kode HTML, sehingga user akan mengerti respon dari web server.



BAB IV. DIAGRAM ALIR

BAB V. KESIMPULAN

Sistem ini dapat dioperasikan oleh dua inputan, yang pertama adalah dengan button dan yang kedua dengan website. Sistem ini bertujuan untuk mengendalikan aktuator yang kemudian status dari aktuatornya akan ditampilkan ke dalam website. Dalam sistem sederhana ini, kami menggunakan LED sebagai aktuator, namun pada pengembanganya aktuator bisa di ganti oleh motor listrik atau pun pompa dengan menambahkan relay pada sistem. Namun kekurangan sistem ini adalah belum adanya autorefresh sehingga User haru merefresh secara manual untuk melihat data terbaru.