LAMPIRAN

Source Code:

```cpp
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
#include <Arduino.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
#include <ESP32Servo.h>
#include "HX711.h"
#include <ArduinoJson.h>
#include <EEPROM.h>
#ifdef ESP32
#include <WiFi.h>
#include <HTTPClient.h>
#include <WiFiClientSecure.h>
#include <AsyncTCP.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClientSecure.h>
#include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

#define S0 14
#define S1 26
#define S2 13
#define S3 27
#define OUT 25
int port_buzzer = 4;

int port_tombol_merah = 34;
int port_tombol_kuning = 35;
int port_tombol_hijau = 32;

extern const char *default_nama_ssid = "isi nama wifi";
extern const char *default_password = "isi pass wifi";
extern const char *default_server = "isi default server";
extern const char *default_apikey = "isi apikey";
String nama_ssid;
String password;
String server_url;
String apikey;
AsyncWebServer server(80);
int reset_default = 0;

int treshold_server = 10;
int treshold_server_custom_r = 0;
int treshold_server_custom_g = 0;

const int pinDT_loadcell_1 = 5;
const int pinSCK_loadcell_1 = 18;
HX711 scale_loadcell_1;
const int pinDT_loadcell_2 = 19;
```

```
const int pinSCK_loadcell_2 = 23;
HX711 scale_loadcell_2;

float baca_loadcell_1() {
  return scale_loadcell_1.get_units(1);
}
float baca_loadcell_2() {
  return scale_loadcell_2.get_units(1);
}

Servo servoESP32_1;
int port_servoESP32_1 = 16;
Servo servoESP32_2;
int port_servoESP32_2 = 17;

int port_Infrared = 33;

void posisi_servoESP32_1(int posisi) {
  servoESP32_1.write(posisi);
}
void posisi_servoESP32_2(int posisi) {
  servoESP32_2.write(posisi);
}

struct RGBData {
  int red;
  int green;
  int blue;
};

const int maxDataEntries = 10;  // Maksimum entri data untuk setiap kode
const int maxKode = 5;          // Maksimum kode yang dapat disimpan

RGBData data[maxKode][maxDataEntries];  // Array untuk menyimpan data RGB
untuk setiap kode
int dataCount[maxKode] = { 0 };         // Array untuk menghitung jumlah
entri data untuk setiap kode

bool trainingMode = false;  // Mode training aktif secara default
int countDetection = 0;     // Menghitung berapa kali deteksi berturut-
turut telah terjadi
int lastDetectedCode = -1;  // Menyimpan kode terakhir yang terdeteksi

void bip() {
  digitalWrite(port_buzzer, HIGH);
  delay(100);
  digitalWrite(port_buzzer, LOW);
}

void bipp() {
  digitalWrite(port_buzzer, HIGH);
  delay(1000);
  digitalWrite(port_buzzer, LOW);
}

void lcd_i2c(String text = "", int kolom = 0, int baris = 0) {
  byte bar[8] = {
    B11111,
    B11111,
```

```
    B11111,
    B11111,
    B11111,
    B11111,
    B11111,
  };
  if (text == "") {
    lcd.init();  //jika error pakai lcd.init();
    lcd.backlight();
    lcd.createChar(0, bar);
    lcd.setCursor(0, 0);
    lcd.print("Loading..");
    for (int i = 0; i < 16; i++) {
      lcd.setCursor(i, 1);
      lcd.write(byte(0));
      delay(100);
    }
    delay(50);
    lcd.clear();
  } else {
    lcd.setCursor(kolom, baris);
    lcd.print(text + "                ");
  }
}

void simpan_data_ke_EEPROM(int kode) {
  int addr = kode * maxDataEntries * sizeof(RGBData);  // Hitung alamat
berdasarkan kode
  EEPROM.begin(512);

  // Simpan semua data untuk kode ini ke EEPROM
  for (int i = 0; i < maxDataEntries; i++) {
    EEPROM.put(addr, data[kode][i]);
    addr += sizeof(RGBData);
  }

  EEPROM.commit();
  EEPROM.end();

  Serial.println("Data untuk kode " + String(kode + 1) + " disimpan di
EEPROM.");
}

void clear_EEPROM() {
  EEPROM.begin(512);

  // Hapus semua data di EEPROM
  for (int addr = 0; addr < 512; addr++) {
    EEPROM.write(addr, 0);  // Tulis nilai 0 ke setiap alamat EEPROM
  }

  EEPROM.commit();
  EEPROM.end();

  // Reset semua dataCount menjadi 0 karena semua data telah dihapus
  for (int kode = 0; kode < maxKode; kode++) {
    dataCount[kode] = 0;
  }
```

```
  Serial.println("Semua data di EEPROM berhasil dihapus.");
}

void cek_dan_print_data() {
  // Loop through all data and print
  for (int kode = 0; kode < maxKode; kode++) {
    if (dataCount[kode] > 0) {
      Serial.println("-------------------------");
      Serial.println("Data untuk kode " + String(kode + 1) + ":");
      for (int i = 0; i < dataCount[kode]; i++) {
        Serial.print("Entry ");
        Serial.print(i + 1);
        Serial.print(": R=");
        Serial.print(data[kode][i].red);
        Serial.print(", G=");
        Serial.print(data[kode][i].green);
        Serial.print(", B=");
        Serial.println(data[kode][i].blue);
      }
    } else {
      Serial.println("Data untuk kode " + String(kode + 1) + " belum
tersedia.");
    }
  }
}

int sudah_cek = 0;
int kode_warna = 0;
int elsekode = 0;
void proses_cek_warna() {
  // Baca frekuensi warna
  int redFrequency, greenFrequency, blueFrequency;

  // Set sensor to read red frequency
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  redFrequency = pulseIn(OUT, LOW);

  // Set sensor to read green frequency
  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);
  greenFrequency = pulseIn(OUT, LOW);

  // Set sensor to read blue frequency
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  blueFrequency = pulseIn(OUT, LOW);

  // Print values
  Serial.print("R: ");
  Serial.print(redFrequency);
  Serial.print(" G: ");
  Serial.print(greenFrequency);
  Serial.print(" B: ");
  Serial.println(blueFrequency);

  bool foundMatch = false;

  // Loop through all stored data
```

```
  for (int kode = 0; kode < maxKode; kode++) {
    for (int i = 0; i < dataCount[kode]; i++) {
      // Threshold untuk perbandingan warna yang terdeteksi dengan data
yang tersimpan
      lcd_i2c("" + (String)redFrequency + "." + (String)greenFrequency +
"." + (String)blueFrequency + "...(" + (String)kode + ")", 0, 1);
      int threshold = treshold_server;  // Ubah sesuai dengan kebutuhan
atau toleransi warna
      if (abs(data[kode][i].red - redFrequency) < threshold &&
abs(data[kode][i].green - greenFrequency) < threshold &&
abs(data[kode][i].blue - blueFrequency) < threshold) {
        Serial.println("Kode terdeteksi: " + String(kode + 1));
        // Periksa jika kode yang sama terdeteksi berturut-turut
        if (kode == lastDetectedCode) {
          countDetection++;
          if (countDetection == 3) {
            int temp_kode = kode + 1;
            if (temp_kode == 1) {
              if (treshold_server_custom_r > 0) {
                if (redFrequency < treshold_server_custom_r) {
                  kode = 1;  // 2
                } else {
                  if (greenFrequency < treshold_server_custom_g) {
                    kode = 2;  // 3
                  }
                }
              }
            }
            Serial.println("TERDETEKSI AKURAT KODE : " + String(kode + 1));
            lcd_i2c("Berhasil : " + String(kode + 1));
            bipp();
            sudah_cek = 1;
            kode_warna = kode + 1;

            countDetection = 0;  // Reset hitungan deteksi berturut-turut
          }
        } else {
          elsekode = 0;
          lastDetectedCode = kode;
          countDetection = 1;  // Mulai hitungan deteksi berturut-turut
dari awal
        }

        foundMatch = true;
        break;
      }
    }
    if (foundMatch) break;
  }

  if (!foundMatch) {
    countDetection = 0;
    elsekode = elsekode + 1;
    if (elsekode > 5) {
      lcd_i2c("Berhasil : 3");
      bipp();
      sudah_cek = 1;
      kode_warna = 3;
      elsekode = 0;
```

```
      countDetection = 0;
    }
    Serial.println("Else...");
  }
}

void load_data_from_EEPROM() {
  EEPROM.begin(512);

  // Load data RGB dari EEPROM untuk setiap kode
  for (int kode = 0; kode < maxKode; kode++) {
    int addr = kode * maxDataEntries * sizeof(RGBData);  // Hitung alamat
berdasarkan kode

    for (int i = 0; i < maxDataEntries; i++) {
      RGBData tempData;
      EEPROM.get(addr, tempData);

      // Memeriksa apakah data yang dimuat valid (tidak semua nol)
      if (tempData.red != 0 || tempData.green != 0 || tempData.blue != 0) {
        data[kode][i] = tempData;
        dataCount[kode]++;
      }

      addr += sizeof(RGBData);
    }
  }

  EEPROM.end();
}

void hapus_data_di_EEPROM(int kode) {
  EEPROM.begin(512);

  // Menghapus entri data dari EEPROM untuk kode yang dipilih
  int addr = kode * maxDataEntries * sizeof(RGBData);  // Hitung alamat
berdasarkan kode

  for (int i = 0; i < maxDataEntries; i++) {
    RGBData emptyData = { 0, 0, 0 };  // Data kosong untuk dihapus
    EEPROM.put(addr, emptyData);
    addr += sizeof(RGBData);
  }

  EEPROM.commit();
  EEPROM.end();

  dataCount[kode] = 0;  // Set jumlah entri data untuk kode ini menjadi 0
  Serial.println("Berhasil menghapus semua entri data untuk kode " +
String(kode + 1));
}

void debug(String message) {
  Serial.println(message);
  //tampilkan jika menggunakan lcd
  lcd.clear();
  lcd_i2c(message);
}
```

```
void writeStringToEEPROM(int address, const String &str) {
  int len = str.length();
  EEPROM.write(address, len);
  for (int i = 0; i < len; i++) {
    EEPROM.write(address + 1 + i, str[i]);
  }
}

String readStringFromEEPROM(int address) {
  int len = EEPROM.read(address);
  char data[len + 1];
  for (int i = 0; i < len; i++) {
    data[i] = EEPROM.read(address + 1 + i);
  }
  data[len] = '\0';
  return String(data);
}

void saveCredentialsToEEPROM() {
  EEPROM.begin(512);
  writeStringToEEPROM(0, nama_ssid);
  writeStringToEEPROM(64, password);
  writeStringToEEPROM(128, server_url);
  writeStringToEEPROM(192, apikey);
  EEPROM.commit();
  debug("Konfigurasi yang disimpan ke EEPROM:");
  debug("nama_ssid: " + nama_ssid);
  debug("Password: " + password);
  debug("Server URL: " + server_url);
  debug("API Key: " + apikey);
}

void loadCredentialsFromEEPROM() {
  EEPROM.begin(512);
  nama_ssid = readStringFromEEPROM(0);
  password = readStringFromEEPROM(64);
  server_url = readStringFromEEPROM(128);
  apikey = readStringFromEEPROM(192);
  if (nama_ssid.length() == 0) {
    nama_ssid = default_nama_ssid;
    debug("SSID Default.");
  } else {
    debug("SSID EEPROM.");
  }
  if (password.length() == 0) password = default_password;
  if (server_url.length() == 0) server_url = default_server;
  if (apikey.length() == 0) apikey = default_apikey;

  Serial.println("SSID LENGTH : " + (String)nama_ssid.length());
  if (nama_ssid.length() > 250 || reset_default == 1) {
    debug("NOVALID:" + nama_ssid);
    delay(3000);
    debug("RESET DEFAULT...");
    nama_ssid = default_nama_ssid;
    password = default_password;
    server_url = default_server;
    apikey = default_apikey;
    saveCredentialsToEEPROM();
    delay(1000);
```

```
    debug("ESP RESTART...");
    delay(1000);
    ESP.restart();
  } else {
    debug("SSID :" + nama_ssid);
    delay(1000);
    debug("PASS :" + password);
    delay(1000);
    debug("URL :" + server_url);
    delay(1000);
    debug("API :" + apikey);
    delay(1000);
  }
}

void setupWiFi() {
  WiFi.begin(nama_ssid.c_str(), password.c_str());
  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
    delay(2000);
    debug("Connect Wi-Fi (" + (String)attempts + ")");
    attempts++;
  }
  if (WiFi.status() == WL_CONNECTED) {
    debug("Terhubung ke Wi-Fi");
    debug("ssid: " + String(WiFi.SSID()));
    debug("IP: " + WiFi.localIP().toString());
    delay(1000);
    debug("System Ready");
    proses_iot("");

  } else {
    //lcd.clear();
    debug("Gagal terhubung");
    delay(2000);
    debug("Beralih mode AP");
    delay(2000);

    debug("Gagal terhubung..");
    WiFi.softAP("wifi-ESP");
    debug("AP: Wifi-ESP");
    delay(5000);
    debug("IP:" + WiFi.softAPIP().toString());

    delay(2000);
    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
      String nama_ssidValue = (nama_ssid.length() > 0) ? nama_ssid :
default_nama_ssid;
      String passwordValue = (password.length() > 0) ? password :
default_password;
      String serverValue = (server_url.length() > 0) ? server_url :
default_server;
      String apiKeyValue = (apikey.length() > 0) ? apikey : default_apikey;
      String htmlContent = R"(
        <!DOCTYPE html>
        <html>
        <head>
          <title>ESP32 WiFi Configuration</title>
          <style>
```

```
            body {
              font-family: Arial, sans-serif;
              margin: 20px;
            }
            input[type="text"],
            input[type="password"] {
              width: 100%;
              padding: 10px;
              margin: 5px 0;
              display: inline-block;
              border: 1px solid #ccc;
              border-radius: 4px;
              box-sizing: border-box;
            }
            input[type="submit"]:hover {
              background-color: #45a049;
            }
            .container {
              padding: 20px;
              border-radius: 5px;
              background-color: #f2f2f2;
            }
          </style>
        </head>
        <body>
          <div style="max-width: 600px; margin: 20px auto; padding: 20px;
border: 1px solid #ccc; border-radius: 5px; background-color: #f9f9f9;">
            <div class="container">
              <h2>ESP WiFi Configuration</h2>
              <form action="/save" method="post">
                <label for="nama_ssid">WiFi SSID:</label>
                <input type="text" id="nama_ssid" name="nama_ssid"
value=")"
                          + nama_ssidValue + R"(" required><br>
                <label for="password">WiFi Password:</label>
                <input type="text" id="password" name="password" value=")"
                          + passwordValue + R"(" required><br>
                <label for="server">Server URL:</label>
                <input type="text" id="server" name="server" value=")"
                          + serverValue + R"(" required><br>
                <label for="apikey">API Key:</label>
                <input type="text" id="apikey" name="apikey" value=")"
                          + apiKeyValue + R"(" required><br>
                <input  style=" width: 100%;color: #fff; background-color:
green; padding: 10px 20px; text-decoration: none; border-radius: 4px;"
type="submit" value="SAVE CONFIGURATION">
              </form>
              <br>
              <br>
              Kembali Ke pengaturan Awal :
             <a href="/reset" style="color: #fff; background-color: red;
padding: 10px 20px; text-decoration: none; border-radius: 4px;">RESET
DEFAULT</a>
            </div>
          </div>
        </body>
        </html>
      )";
      request->send(200, "text/html", htmlContent);
```

```
    });
    server.on("/save", HTTP_POST, [](AsyncWebServerRequest *request) {
      if (request->args() > 0) {  // Pastikan ada argumen yang disampaikan
        for (uint8_t i = 0; i < request->args(); i++) {
          if (request->argName(i) == "nama_ssid") {
            nama_ssid = request->arg(i);
          } else if (request->argName(i) == "password") {
            password = request->arg(i);
          } else if (request->argName(i) == "server") {
            server_url = request->arg(i);
          } else if (request->argName(i) == "apikey") {
            apikey = request->arg(i);
          }
        }
        saveCredentialsTo*EEPROM*();  // Simpan konfigurasi ke *EEPROM*
        request->send(200, "text/html", R"(
          <div style="max-width: 600px; margin: 20px auto; padding: 20px;
border: 1px solid #ccc; border-radius: 5px; background-color: #f9f9f9;">
            <h2 style="color: #4CAF50;">Konfigurasi Berhasil Disimpan</h2>
            <p><br>Klik tombol dibawah ini untuk restart esp <br><br><br><a
href="/restart" style="color: #fff; background-color: #4CAF50; padding:
10px 20px; text-decoration: none; border-radius: 4px;">RESTART ESP</a></p>
          </div>
        </body>
        )");
      } else {
        request->send(400, "text/html", "Bad Request: Tidak ada data yang
disampaikan.");
      }
    });
    server.on("/reset", HTTP_GET, [](AsyncWebServerRequest *request) {
      nama_ssid = default_nama_ssid;
      password = default_password;
      server_url = default_server;
      apikey = default_apikey;
      saveCredentialsTo*EEPROM*();
      request->send(200, "text/html", R"(
          <div style="max-width: 600px; margin: 20px auto; padding: 20px;
border: 1px solid #ccc; border-radius: 5px; background-color: #f9f9f9;">
            <h2 style="color: RED;">Konfigurasi Berhasil Di Reset</h2>
            <p><br>Klik tombol dibawah ini untuk restart esp <br><br><br><a
href="/restart" style="color: #fff; background-color: red; padding: 10px
20px; text-decoration: none; border-radius: 4px;">RESTART ESP</a></p>
          </div>
        )");
    });
    server.on("/restart", HTTP_GET, [](AsyncWebServerRequest *request) {
      request->send(200, "text/html", R"(
        <head>
  <meta http-equiv="refresh" content="5;url=/">
</head>
<body>
  <div style="max-width: 600px; margin: 20px auto; padding: 20px; border:
1px solid #ccc; border-radius: 5px; background-color: #f9f9f9;">
    <p><br>ESP Restart... <br><br></p>
  </div>
</body>
        )");
```

```
      delay(1000);    // Tambahkan jeda sebelum merestart
      ESP.restart();  // Restart ESP
      request->redirect("/");
    });
  }
  server.begin();
}

int looping_iot = 0;
int out_1 = 0;
int out_2 = 0;
int out_3 = 0;
int out_4 = 0;
int out_5 = 0;
int out_6 = 0;
int out_7 = 0;
int out_8 = 0;
int out_9 = 0;
int out_10 = 0;

void proses_iot(String nilai) {
  if (WiFi.status() != WL_CONNECTED) return;

  WiFiClientSecure client;  // Use WiFiClientSecure for HTTPS
  client.setInsecure();  // Optional, skip certificate validation for
testing purposes

  HTTPClient http;
  String url = server_url + apikey + nilai;  // Menggunakan server_url
  url.replace(" ", "%20");
  Serial.println("Request URL: " + url);
  http.begin(client, url);
  int httpResponseCode = http.GET();
  String payload = http.getString();  // Get the response payload
  Serial.println(httpResponseCode);  // Print HTTP response code
  Serial.println(payload);  // Print response payload
  if (httpResponseCode == HTTP_CODE_OK) {
    const size_t capacity = JSON_OBJECT_SIZE(1024);
    DynamicJsonDocument jsonDoc(capacity);
    String jsonResponse = http.getString();
    DeserializationError error = deserializeJson(jsonDoc, jsonResponse);
    if (error) {
      Serial.println("Error parsing JSON: " + String(error.c_str()));
      return;
    }
    for (int i = 1; i <= 10; i++) {
      String out = jsonDoc["out_" + String(i)].as<String>();
      Serial.println("out_" + String(i) + ": " + out);
    }
    out_1 = jsonDoc["out_1"].as<int>();
    out_2 = jsonDoc["out_2"].as<int>();
    out_3 = jsonDoc["out_3"].as<int>();
    out_4 = jsonDoc["out_4"].as<int>();
    out_5 = jsonDoc["out_5"].as<int>();
    out_6 = jsonDoc["out_6"].as<int>();
    out_7 = jsonDoc["out_7"].as<int>();
    out_8 = jsonDoc["out_8"].as<int>();
    out_9 = jsonDoc["out_9"].as<int>();
    out_10 = jsonDoc["out_10"].as<int>();
```

```
    treshold_server = out_1;
    treshold_server_custom_r = out_2;
    treshold_server_custom_g = out_3;
  } else {
    Serial.println("Error Code: " + String(httpResponseCode));
  }
  http.end();
}

void servo_warna_standby() {
  posisi_servoESP32_1(150);
  delay(1000);
}

void servo_warna_cek() {
  posisi_servoESP32_1(90);
  delay(1000);
}

void servo_warna_kirim() {
  posisi_servoESP32_1(0);
  delay(1000);
}

void servo_pemisah_tengah() {
  posisi_servoESP32_2(70);
  delay(1000);
}

void servo_pemisah_1() {
  posisi_servoESP32_2(40);
  delay(1000);
}

void servo_pemisah_2() {
  posisi_servoESP32_2(100);
  delay(1000);
}

void servo_pemisah_3() {
  posisi_servoESP32_2(130);
  delay(1000);
}

void setup() {
  Serial.begin(9600);
  lcd_i2c();

  pinMode(port_tombol_merah, INPUT);
  pinMode(port_tombol_kuning, INPUT);
  pinMode(port_tombol_hijau, INPUT);

  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);

  pinMode(OUT, INPUT);
```

```
  pinMode(port_buzzer, OUTPUT);
  bip();
  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);
  load_data_from_EEPROM();
  int tombol_merah = digitalRead(port_tombol_merah);
  Serial.print(" MERAH : " + (String)tombol_merah);

  int tombol_kuning = digitalRead(port_tombol_kuning);
  Serial.print(" KUNING : " + (String)tombol_kuning);

  int tombol_hijau = digitalRead(port_tombol_hijau);
  Serial.println(" HIJAU : " + (String)tombol_hijau);

  if (tombol_merah == 1 && tombol_kuning == 1 & tombol_hijau == 1) {
    lcd.clear();
    lcd_i2c("Kosongkan EEPROM");
    bipp();
    clear_EEPROM();
    delay(5000);
    lcd_i2c("Berhsil Hapus..");
    lcd_i2c("RESTART...", 0, 1);
    bipp();
    ESP.restart();
  } else if (tombol_merah == 1 && tombol_kuning == 1 & tombol_hijau == 0) {
    lcd.clear();
    lcd_i2c("Check EEPROM");
    bipp();
    cek_dan_print_data();
    delay(5000);
    lcd_i2c("Serial Monitor..", 0, 1);
  }

  EEPROM.begin(512);
  //BERI NILAI 1 JIKA MAU DIRESET (PERTAMA UPLOAD WAJIB RESET)
  reset_default = 0;
  loadCredentialsFromEEPROM();
  setupWiFi();

  servoESP32_1.attach(port_servoESP32_1);
  servoESP32_2.attach(port_servoESP32_2);

  pinMode(port_Infrared, INPUT_PULLUP);

  servo_warna_standby();
  servo_pemisah_tengah();

  scale_loadcell_1.begin(pinDT_loadcell_1, pinSCK_loadcell_1);
  scale_loadcell_1.set_scale(2280.f);
  scale_loadcell_1.tare();
  scale_loadcell_2.begin(pinDT_loadcell_2, pinSCK_loadcell_2);
  scale_loadcell_2.set_scale(2280.f);
  scale_loadcell_2.tare();
}

void test_servo_1() {
  servo_warna_standby();
  servo_warna_cek();
  servo_warna_kirim();
```

```
  servo_warna_standby();
}

void test_servo_2() {
  servo_pemisah_1();
  servo_pemisah_2();
  servo_pemisah_3();
}

int status = 0;
int view_lcd = 0;
int kode_training = 0;
void loop() {
  if (status == 0) {
    int tombol_merah = digitalRead(port_tombol_merah);
    Serial.print(" MERAH : " + (String)tombol_merah);

    int tombol_kuning = digitalRead(port_tombol_kuning);
    Serial.print(" KUNING : " + (String)tombol_kuning);

    int tombol_hijau = digitalRead(port_tombol_hijau);
    Serial.println(" HIJAU : " + (String)tombol_hijau);

    if (tombol_merah == 1) {
      status = 100;
      kode_training = 1;
      lcd.clear();
      bip();
      delay(1000);
      lcd_i2c("EEPROM (1)");

      //pendaftaran
    } else if (tombol_kuning == 1) {
      status = 100;
      kode_training = 2;
      lcd.clear();
      bip();
      delay(1000);
      lcd_i2c("EEPROM (2)");
      //pendaftaran
    } else if (tombol_hijau == 1) {
      status = 100;
      kode_training = 3;
      lcd.clear();
      bip();
      delay(1000);
      lcd_i2c("EEPROM (3)");
      //pendaftaran
    }

    int Infrared = digitalRead(port_Infrared);

    if (Infrared == 0) {
      bip();
      lcd_i2c("Detect..");
      delay(1000);
      status = 1;
    }
```

```
    if (view_lcd > 10) {
      view_lcd = 0;
      float loadcell_1 = baca_loadcell_1();
      // Serial.print(" Berat1: " + (String)loadcell_1 + " gram");
      if (loadcell_1 < 1) {
        loadcell_1 = 0;
      }

      float loadcell_2 = baca_loadcell_2();
      if (loadcell_2 < 1) {
        loadcell_2 = 0;
      }
      // Serial.println(" Berat2: " + (String)loadcell_2 + " gram");

      lcd_i2c("Berat 1: " + (String)loadcell_1 + " g");
      lcd_i2c("Berat 2: " + (String)loadcell_2 + " g", 0, 1);
    } else {
      view_lcd = view_lcd + 1;
    }

  } else if (status == 1) {
    lcd.clear();
    lcd_i2c("Scanning...");
    servo_warna_cek();
    status = 2;
  } else if (status == 2) {
    int redFrequency, greenFrequency, blueFrequency;  // Declare variables
here

    if (trainingMode == false) {
      proses_cek_warna();  // Memulai proses cek warna

      if (sudah_cek == 1) {
        status = 0;
        sudah_cek = 0;
        if (kode_warna == 1) {
          lcd_i2c("KODE (1)");
          servo_pemisah_1();
        } else if (kode_warna == 2) {
          lcd_i2c("KODE (1)");
          servo_pemisah_2();
        } else {
          lcd_i2c("KODE (1)");
          servo_pemisah_3();
        }
        servo_warna_kirim();
        //kirim server

        float loadcell_1 = baca_loadcell_1();
        Serial.print(" Berat1: " + (String)loadcell_1 + " gram");
        if (loadcell_1 < 1) {
          loadcell_1 = 0;
        }

        float loadcell_2 = baca_loadcell_2();
        if (loadcell_2 < 1) {
          loadcell_2 = 0;
        }
        Serial.println(" Berat2: " + (String)loadcell_2 + " gram");
```

```
        lcd.clear();
        lcd_i2c("Kirim Server..");

        proses_iot("&merah=" + (String)loadcell_1 + "&orange=" +
(String)loadcell_2);
        servo_warna_standby();
        servo_pemisah_tengah();
        kode_warna = 0;

        bip();
      }
    }

    if (Serial.available() > 0) {
      String input = Serial.readStringUntil('\n');

      if (input.equals("cek")) {
        cek_dan_print_data();
      } else if (input.equals("restart")) {
        Serial.println("ESP akan di-restart...");
        delay(1000);
        ESP.restart();
      } else if (input.equals("run")) {
        trainingMode = false;  // Matikan mode training
        Serial.println("Mode training dinonaktifkan. Mulai proses cek
warna...");
        delay(1000);
      } else if (input.equals("train")) {
        trainingMode = true;  // Matikan mode training
        Serial.println("Mode training diaktifkan...");
        delay(1000);
      } else if (input.equals("clear")) {
        clear_EEPROM();
      } else if (input.startsWith("delete ")) {
        // Menghapus entri data berdasarkan input "delete [kode]"
        input.remove(0, 7);  // Menghapus "delete "
        int kode = input.substring(0, 1).toInt();

        if (kode >= 1 && kode <= maxKode) {
          hapus_data_di_EEPROM(kode - 1);
          load_data_from_EEPROM();  // Reload data setelah penghapusan
        } else {
          Serial.println("Perintah delete tidak valid.");
        }
      } else {
        // Jika input adalah kode (1 sampai maxKode)
        int kode = input.toInt();
        if (kode >= 1 && kode <= maxKode) {
          kode--;  // Ubah dari 1-indexed ke 0-indexed

          for (int j = 0; j < 5; j++) {                          // Ambil
5 data RGB
            Serial.println("Mengambil sampel " + String(j + 1));  // Output
serial untuk setiap pengambilan sampel
            delay(1000);

            // Set sensor to read red frequency
            digitalWrite(S2, LOW);
            digitalWrite(S3, LOW);
```

```
            redFrequency = pulseIn(OUT, LOW);

            // Set sensor to read green frequency
            digitalWrite(S2, HIGH);
            digitalWrite(S3, HIGH);
            greenFrequency = pulseIn(OUT, LOW);

            // Set sensor to read blue frequency
            digitalWrite(S2, LOW);
            digitalWrite(S3, HIGH);
            blueFrequency = pulseIn(OUT, LOW);

            // Simpan data RGB ke array untuk kode yang dipilih
            RGBData newData = { redFrequency, greenFrequency, blueFrequency
};
            data[kode][dataCount[kode]] = newData;
            dataCount[kode]++;

            Serial.print("R: ");
            Serial.print(redFrequency);
            Serial.print(", G: ");
            Serial.print(greenFrequency);
            Serial.print(", B: ");
            Serial.println(blueFrequency);
            bip();
            delay(1000);  // Tunggu 1 detik sebelum mengambil data
berikutnya
          }

          // Simpan data ke EEPROM setelah selesai mengambil 5 data
          simpan_data_ke_EEPROM(kode);

          // Print informasi bahwa data telah disimpan
          Serial.println("Data RGB berhasil disimpan untuk kode " + input);
          bipp();
        } else {
          Serial.println("Perintah tidak valid.");
        }
      }
    }
    delay(1000);
  } else if (status == 100) {
    int redFrequency, greenFrequency, blueFrequency;

    trainingMode = true;
    if (kode_training >= 1 && kode_training <= maxKode) {
      kode_training--;  // Ubah dari 1-indexed ke 0-indexed

      for (int j = 0; j < 5; j++) {                          // Ambil 5
data RGB
        Serial.println("Mengambil sampel " + String(j + 1));  // Output
serial untuk setiap pengambilan sampel
        delay(1000);

        // Set sensor to read red frequency
        digitalWrite(S2, LOW);
        digitalWrite(S3, LOW);
        redFrequency = pulseIn(OUT, LOW);
```

```
        // Set sensor to read green frequency
        digitalWrite(S2, HIGH);
        digitalWrite(S3, HIGH);
        greenFrequency = pulseIn(OUT, LOW);

        // Set sensor to read blue frequency
        digitalWrite(S2, LOW);
        digitalWrite(S3, HIGH);
        blueFrequency = pulseIn(OUT, LOW);

        // Simpan data RGB ke array untuk kode yang dipilih
        RGBData newData = { redFrequency, greenFrequency, blueFrequency };
        data[kode_training][dataCount[kode_training]] = newData;
        dataCount[kode_training]++;

        Serial.print("R: ");
        Serial.print(redFrequency);
        Serial.print(", G: ");
        Serial.print(greenFrequency);
        Serial.print(", B: ");
        Serial.println(blueFrequency);
        bip();
        delay(1000);  // Tunggu 1 detik sebelum mengambil data berikutnya
      }

      // Simpan data ke EEPROM setelah selesai mengambil 5 data
      simpan_data_ke_EEPROM(kode_training);

      bipp();
      trainingMode = false;
      status = 0;
    }
  }
}

```
```
```