LAPORAN PRAKTIKUM

EMBEDDED INTELLIGENCE SYSTEM

Dosen Pengampu :

Susilawati, M.Si.

Disusun Oleh :

Nadiyah Nur R
2210631170142

KELAS 4E

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

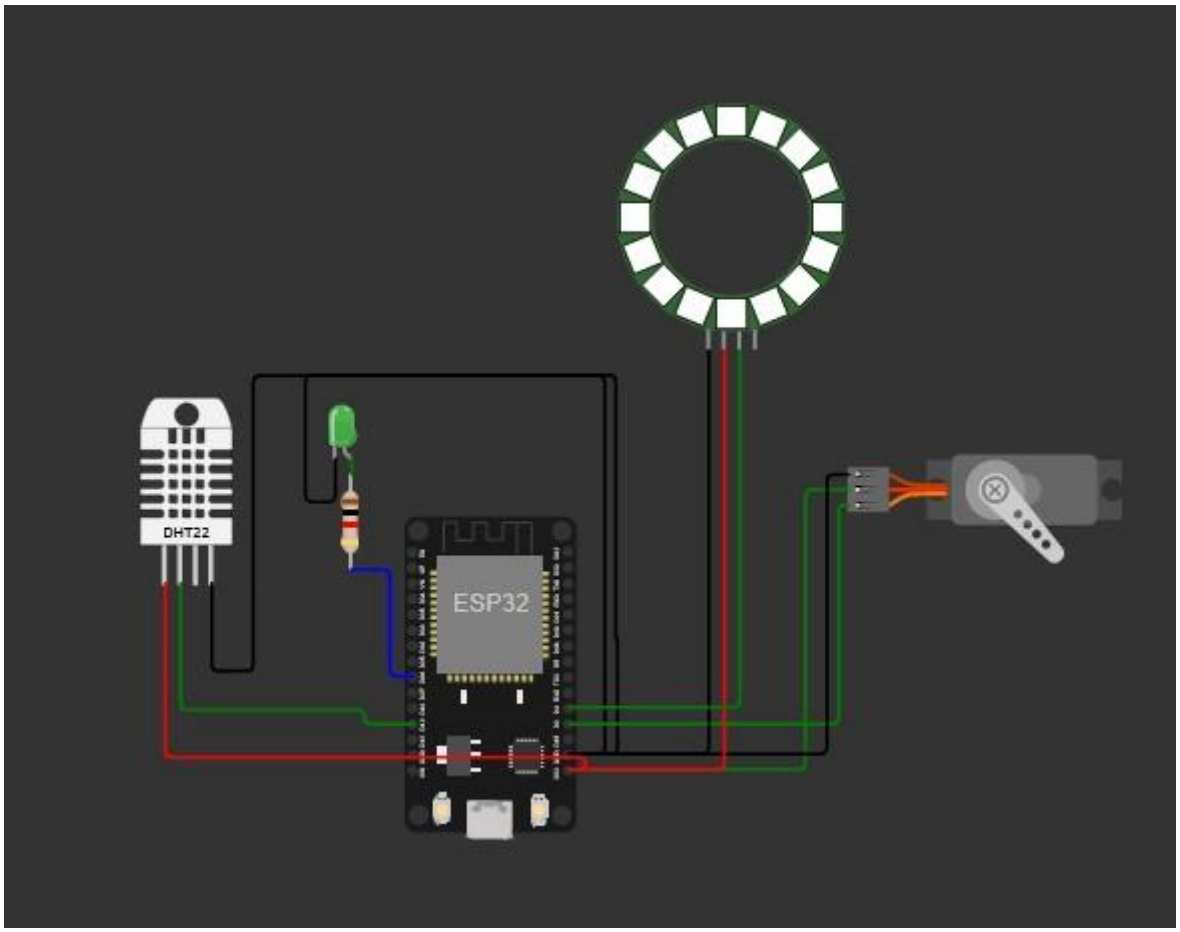
UNIVERSITAS SINGAPERBANGSA KARAWANG

2024

SOAL PRAKTIKUM

1. Setelah belajar ESP8266, project iot apa yang terpikirkan oleh mu ? Anda bisa memikirkan ide project apa saja yang menarik untuk diimplementasikan menggunakan ESP8266. Beberapa contoh ide bisa berupa smart home automation, monitoring suhu dan kelembaban, atau sistem pengecekan stok barang secara real-time.
2. Jelaskan secara detail seperti sensor apa saja yang akan digunakan dan output apa saja yang akan ditampilkan.
3. Kirimkan dengan format PDF source code dan morkup project nya.

Project ESP32 dengan Sensor dan Kontrol MQTT



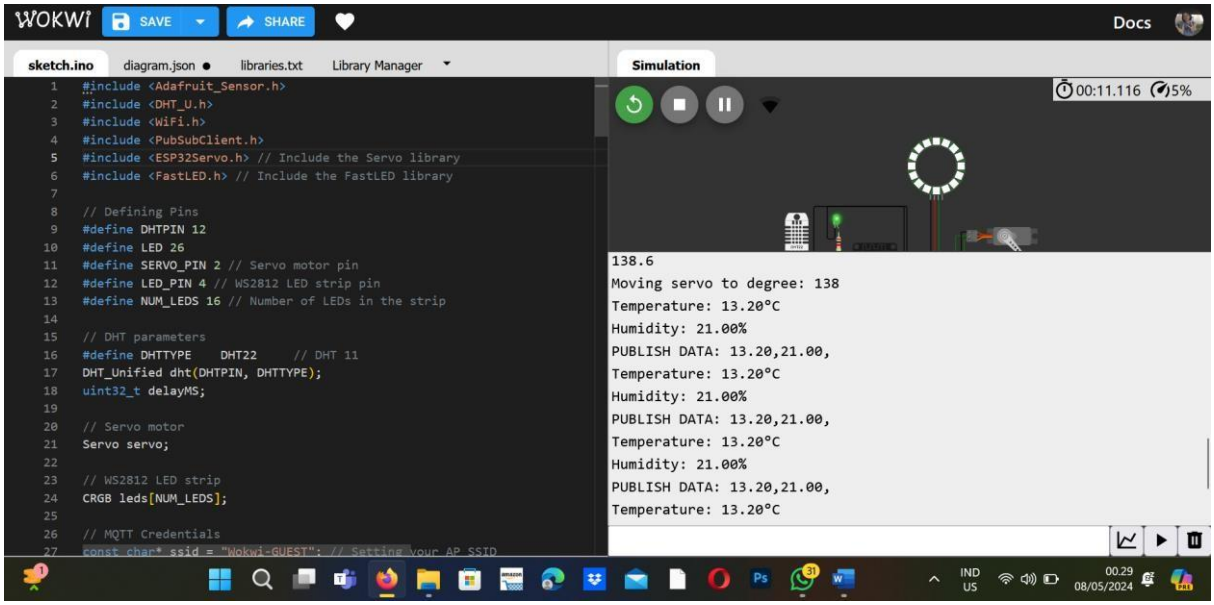
Komponen:

1. **ESP32:** Modul mikrokontroler yang digunakan sebagai inti sistem. ESP32 memiliki kemampuan WiFi, Bluetooth, dan pemrosesan yang kuat untuk menjalankan aplikasi IoT.
2. **LED:** Dioda pemancar cahaya yang digunakan untuk indikasi visual. Dalam proyek ini, LED dapat digunakan untuk menunjukkan status sistem, seperti terhubung/tidak terhubung ke WiFi atau MQTT, atau untuk memberikan umpan balik visual untuk tindakan kontrol.
3. **Resistor:** Komponen elektronik yang digunakan untuk membatasi aliran arus dan melindungi komponen lain. Dalam proyek ini, resistor dapat digunakan untuk membatasi arus ke LED dan memastikan operasi yang stabil.
4. **DHT22:** Sensor suhu dan kelembapan yang digunakan untuk mengukur dan melaporkan kondisi lingkungan. Sensor ini menyediakan data suhu dan kelembapan yang dapat diakses oleh ESP32 melalui protokol komunikasi I2C.
5. **Servo:** Motor servo yang digunakan untuk mengontrol gerakan rotasi dengan presisi. Dalam proyek ini, servo dapat digunakan untuk menggerakkan objek fisik, seperti robot atau model, atau untuk mengatur sudut elemen seperti lampu atau kamera.

6. **NeoPixel Compatible LED Ring (WS2812):** Cincin LED yang terdiri dari LED addressable individual yang dapat dikontrol secara terpisah. LED ini memungkinkan kontrol warna dan animasi yang kompleks untuk efek pencahayaan yang dinamis.

Output:

- **Data Suhu dan Kelembapan:** Nilai suhu dan kelembapan yang diukur oleh sensor DHT22 akan dikirim melalui MQTT ke server MQTT. Data ini dapat divisualisasikan pada dashboard atau aplikasi IoT untuk memantau kondisi lingkungan.
- **Status LED:** Status LED (ON/OFF) akan dikendalikan berdasarkan pesan yang diterima dari server MQTT. Pesan ini dapat berasal dari aplikasi pengguna atau sistem otomasi lainnya.
- **Posisi Servo:** Posisi servo motor akan disesuaikan berdasarkan nilai sudut yang diterima dari server MQTT. Nilai ini memungkinkan kontrol presisi gerakan rotasi servo.
- **Warna LED WS2812:** Warna cincin LED WS2812 akan dikendalikan berdasarkan nilai RGB yang diterima dari server MQTT. Nilai RGB memungkinkan pemilihan warna yang tepat untuk efek pencahayaan yang diinginkan.



Source Code :

```
#include <Adafruit_Sensor.h>
#include <DHT_U.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <ESP32Servo.h> // Include the Servo library
#include <FastLED.h> // Include the FastLED library

// Defining Pins
#define DHTPIN 12
```

```
#define LED 26
#define SERVO_PIN 2 // Servo motor pin
#define LED_PIN 4 // WS2812 LED strip pin
#define NUM_LEDS 16 // Number of LEDs in the strip

// DHT parameters
#define DHTTYPE    DHT22     // DHT 11
DHT_Unified dht(DHTPIN, DHTTYPE);
uint32_t delayMS;

// Servo motor
Servo servo;

// WS2812 LED strip
CRGB leds[NUM_LEDS];

// MQTT Credentials
const char* ssid = "Wokwi-GUEST"; // Setting your AP SSID
const char* password = ""; // Setting your AP PSK
const char* mqttServer = "broker.hivemq.com";
// const char* mqttUserName = "bqzbdodo";
// const char* mqttPwd = "5oU2W_QN2WD8";
const char* clientID = "ujaisldaaasdfgh;laslksdja1"; // Client ID
username+0001
const char* topic = "Tempdata"; // Publish topic

// Parameters for using non-blocking delay
unsigned long previousMillis = 0;
const long interval = 1000;
String msgStr = "";      // MQTT message buffer
float temp, hum;

// Setting up WiFi and MQTT client
WiFiClient  espClient;
PubSubClient client(espClient);

void setup_wifi() {
  delay(10);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
```

```
}

void reconnect() {
  while (!client.connected()) {
    if (client.connect(clientID)) {
      Serial.println("MQTT connected");
      client.subscribe("lights");
      client.subscribe("servo"); // Subscribe to servo topic
      client.subscribe("lights/neopixel"); // Subscribe to neopixel topic
      Serial.println("Topic Subscribed");
    }
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);  // wait 5sec and retry
    }
  }
}

// Subscribe callback
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived in topic: ");
  Serial.println(topic);
  Serial.print("Message: ");
  String data = "";
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
    data += (char)payload[i];
  }
  Serial.println();
  Serial.print("Message size: ");
  Serial.println(length);
  Serial.println();
  Serial.println("-----------------------");
  Serial.println(data);

  if (String(topic) == "lights") {
    if (data == "ON") {
      Serial.println("LED");
      digitalWrite(LED, HIGH);
    }
    else {
      digitalWrite(LED, LOW);
    }
  }
  else if (String(topic) == "servo") {
```

```
    int degree = data.toInt(); // Convert the received data to an integer
    Serial.print("Moving servo to degree: ");
    Serial.println(degree);
    servo.write(degree); // Move the servo to the specified degree
  }
  else if (String(topic) == "lights/neopixel") {
    int red, green, blue;
    sscanf(data.c_str(), "%d,%d,%d", &red, &green, &blue); // Parse the
received data into RGB values
    Serial.print("Setting NeoPixel color to (R,G,B): ");
    Serial.print(red);
    Serial.print(",");
    Serial.print(green);
    Serial.print(",");
    Serial.println(blue);
    fill_solid(leds, NUM_LEDS, CRGB(red, green, blue)); // Set all LEDs in the
strip to the specified color
    FastLED.show(); // Update the LED strip with the new color
     fill_solid(leds, NUM_LEDS, CRGB(red, green, blue));
     FastLED.show();
  }
}

void setup() {
  Serial.begin(115200);
  // Initialize device.
  dht.begin();
  // Get temperature sensor details.
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  dht.humidity().getSensor(&sensor);
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);

  // Setup servo
  servo.attach(SERVO_PIN, 500, 2400);
  servo.write(0);

  // Setup WS2812 LED strip
  FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);

  setup_wifi();
  client.setServer(mqttServer, 1883); // Setting MQTT server
  client.setCallback(callback); // Define function which will be called when a
message is received.
}
```

```
void loop() {
  if (!client.connected()) { // If client is not connected
    reconnect(); // Try to reconnect
  }
  client.loop();
  unsigned long currentMillis = millis(); // Read current time
  if (currentMillis - previousMillis >= interval) { // If current time - last
time > 5 sec
    previousMillis = currentMillis;
    // Read temperature and humidity
    sensors_event_t event;
    dht.temperature().getEvent(&event);
    if (isnan(event.temperature)) {
      Serial.println(F("Error reading temperature!"));
    }
    else {
      Serial.print(F("Temperature: "));
      temp = event.temperature;
      Serial.print(temp);
      Serial.println(F("°C"));
    }
    // Get humidity event and print its value
    dht.humidity().getEvent(&event);
    if (isnan(event.relative_humidity)) {
      Serial.println(F("Error reading humidity!"));
    }
    else {
      Serial.print(F("Humidity: "));
      hum = event.relative_humidity;
      Serial.print(hum);
      Serial.println(F("%"));
    }
    msgStr = String(temp) + "," + String(hum) + ",";
    byte arrSize = msgStr.length() + 1;
    char msg[arrSize];
    Serial.print("PUBLISH DATA: ");
    Serial.println(msgStr);
    msgStr.toCharArray(msg, arrSize);
    client.publish(topic, msg);
    msgStr = "";
    delay(1);
  }
}
```

Penjelasan :

Pengaturan WiFi dan MQTT (Fungsi `setup_wifi` dan `reconnect`):

- Fungsi `setup_wifi`:
  - Menyambungkan ESP32 ke jaringan WiFi yang ditentukan (`ssid` dan `password`).
  - Mencetak alamat IP perangkat setelah koneksi berhasil.
- Fungsi `reconnect`:
  - Mencoba untuk menyambungkan kembali ke broker MQTT (`mqttServer`) jika koneksi terputus.
  - Berlangganan ke tiga topik setelah koneksi berhasil:
    - `"lights"`: Untuk menerima perintah untuk mengontrol LED tunggal (ON/OFF).
    - `"servo"`: Untuk menerima perintah untuk mengontrol motor servo (sudut dalam derajat).
    - `"lights/neopixel"`: Untuk menerima perintah untuk mengontrol strip LED WS2812 (nilai RGB).

Callback Berlangganan (Fungsi `callback`):

- Dipanggil setiap kali pesan diterima pada topik yang berlangganan.
- Mencetak nama topik yang diterima, data pesan, dan ukuran pesan untuk keperluan debugging.
- Menerjemahkan data pesan berdasarkan topik yang berlangganan:
  - `"lights"`:
    - Memeriksa apakah pesan tersebut "ON". Jika ya, menyalakan LED tunggal (digitalWrite(LED, HIGH)). Jika tidak, mematikannya (digitalWrite(LED, LOW)).
  - `"servo"`:
    - Mengubah pesan yang diterima (string) menjadi integer yang mewakili sudut yang diinginkan untuk motor servo.
    - Mengatur posisi servo menggunakan perintah `servo.write(degree)`.
  - `"lights/neopixel"`:
    - Mengekstrak nilai warna merah, hijau, dan biru (RGB) dari pesan yang diterima menggunakan `sscanf`.
    - Mengatur warna semua LED di strip WS2812 ke nilai RGB yang diuraikan menggunakan `fill_solid` dan memperbarui strip LED dengan `FastLED.show`.

Fungsi `setup`:

- Meminisi komunikasi serial untuk mencetak pesan debug ke komputer Anda.
- Meminisi sensor DHT22 menggunakan `dht.begin()`.
- Mengambil detail sensor menggunakan `dht.temperature().getSensor(&sensor)` dan `dht.humidity().getSensor(&sensor)`.

- Mengatur pin LED tunggal (`LED`) sebagai output dan mematikannya pada awalnya (`digitalWrite(LED, LOW)`).
- Meminisi motor servo dengan memasangnya ke pin yang ditentukan (`SERVO_PIN`) dan menyetel posisi awal (`servo.write(0)`) menggunakan fungsi `servo.attach`.
- Meminisi pustaka strip LED WS2812 (`FastLED`) dengan pin LED (`LED_PIN`), jenis LED (`WS2812`), dan urutan warna (`GRB`).
- Memanggil `setup_wifi` untuk terhubung ke WiFi.
- Mengatur alamat server MQTT (`mqttServer`) dan port (1883) menggunakan `client.setServer`.
- Mendefinisikan fungsi callback (`callback`) yang akan dipanggil untuk pesan yang masuk.

Fungsi `loop`:

- Loop program utama yang berjalan terus menerus.
- Memeriksa apakah koneksi ke broker MQTT terputus dan mencoba untuk menyambungkan kembali jika perlu menggunakan `reconnect`.
- Memanggil `client.loop()` untuk menangani tugas komunikasi MQTT.
- Membaca waktu saat ini menggunakan `millis`.
- Memeriksa apakah sudah waktunya untuk membaca data sensor berdasarkan interval yang ditentukan oleh `interval` (1 detik dalam kasus ini).
    - Membaca suhu dan kelembaban dari sensor DHT22 menggunakan `dht.temperature().getEvent` dan `dht.humidity().getEvent`.
    - Jika pembacaan berhasil, cetak nilainya ke monitor serial.
    - Mempersiapkan string pesan (`msgStr`) yang berisi suhu, kelembaban, dan tanda koma pemisah.
    - Mengubah string pesan menjadi array karakter (`msg`) untuk penerbitan MQTT.
    - Mencetak pesan yang akan diterbitkan untuk debugging.
    - Menerbitkan string pesan (`msg`) ke topik yang ditentukan (`topic`) menggunakan `client.publish`.
    - Mengatur ulang string pesan (`msgStr`) untuk iterasi berikutnya.
    - Memasukkan penundaan kecil (`delay(1)`) untuk menghindari membebani bus komunikasi.