BAB II
PELAKSANAAN UJIAN KOMPETENSI

2.1. Topik Yang Dipelajari

Perancangan dan Perekayasaan Industrial IoT, pelatihan komprehensif ini dirancang untuk membekali penulis dengan pengetahuan teoritis dan keterampilan praktis dalam merekayasa, mengembangkan, dan mengimplementasikan solusi Industrial Internet of Things (IIoT). Materi akan mencakup standar kompetensi yang relevan dari SKKNI IoT. Yang terdiri dari:

a. Fundamental dari Sistem Tertanam (*Embedded System*) dan IoT.

Dalam era transformasi digital industri, pemahaman mengenai sistem tertanam dan IoT menjadi fondasi utama dalam pengembangan solusi industri berbasis teknologi cerdas. Pada hal ini akan diberikan gambaran menyeluruh, dari konsep teoretis hingga implementasi praktis dalam bentuk proyek prototipe IIoT. Beberapa komponen penting yang diperlukan dalam pembuatan proyek prototipe IIoT, seperti :

1. Lampu LED komponen elekronika yang memancarkan cahaya

2. ESP32 sebagai mikrokontroler utama,

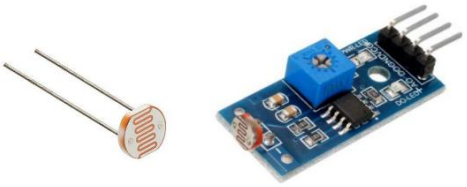
3. Push button ntuk menghubungkan atau memutuskan aliran arus listrik

4. Potensiometer jenis resistor yang nilai resistansinya dapat diatur sesuai kebutuhan rangkaian elektronika atau pengguna

5. Sensor DHT11 untuk pengukuran suhu dan kelembapan,

6. HC-SR04 untuk pengukuran jarak,

7. LDR untuk pendeteksian intensitas cahaya.



8. Modul Relay bekerja dengan memutus dan menyambung aliran listrik dalam rangkaian, berfungsi sebagai sakelar otomatis.

9. RS-485 Modbus Temperature Humidity Sensor. Sensor yang mengukur suhu dan kelembaban udara serta mengirimkan data melalui antarmuka RS-485 menggunakan protokol Modbus

10. Display LCD 16x2 I2C, media tampilan yang paling mudah untuk diamati karena menghasilkan tampilan karakter yang baik dan cukup banyak.

11. Adaptor power supply adalah komponen elektronik yang mengubah arus listrik AC dari sumber listrik menjadi DC.

Semua komponen ini disusun pada breadboard dan dihubungkan dengan kabel jumper, memungkinkan perakitan sistem tanpa perlu penyolderan. Selain itu, juga penggunaan relay, LCD 16x2 I2C, dan komunikasi RS-485, yang memperkaya kemampuan sistem untuk beroperasi dalam lingkungan industri nyata.

Embedded system adalah komputer kecil yang hanya menjalankan satu atau beberapa fungsi tertentu secara spesifik. ESP32, yang disebutkan adalah contoh nyata dari sistem ini. Lebih jauh, menguraikan konsep IoT yakni ekosistem perangkat yang saling terhubung melalui internet untuk mengumpulkan, memproses, dan bertindak berdasarkan data. Penerapan IoT di sektor industri disebut sebagai IIoT, yang bertujuan untuk meningkatkan efisiensi, melakukan pemantauan jarak jauh, dan otomatisasi berbasis data.

Tak kalah penting, aspek Keselamatan dan Kesehatan Kerja (K3) dalam proses perakitan perangkat IoT. Hal ini menjadi krusial mengingat proyek IoT yang melibatkan arus listrik, komponen tajam, dan soldering berisiko jika tidak ditangani secara aman. panduan praktis dalam aspek K3 seperti penggunaan pakaian kerja, gelang anti-statis, dan stand solder, yang mendukung proses perekayasaan perangkat secara profesional dan aman.

Melalui pemahaman teoretis yang kuat dan persiapan teknis yang matang, pengembangan perangkat IIoT tidak hanya memberikan inovasi pada sistem otomasi industri, tetapi juga memperkuat kapabilitas manusia dalam menciptakan solusi teknologi yang adaptif, hemat energi, dan aman. Hal ini sejalan dengan arah transformasi industri menuju efisiensi tinggi dan sistem kerja yang berbasis data.

b. Instalasi software Industrial IoT

Software dan Tools yang diperlukan :

- Arduino IDE

  Digunakan untuk membuat, meng-upload, dan debugging program ke mikrokontroler (seperti ESP32).

  Instalasi board ESP32: melalui Preferences dan Board Manager.

  Dapat ditambahkan library melalui menu Sketch > Manage Library.

- Wokwi IoT Simulator

  Simulator online yang kompatibel dengan Arduino IDE. Digunakan untuk membuat simulasi proyek tanpa perangkat fisik.

- Node.js dan Node-RED

  Node.js: runtime untuk menjalankan aplikasi berbasis JavaScript.

  Node-RED: visual programming untuk IoT

- ModScan dan Modbus Poll

  Untuk menguji dan memonitor komunikasi Modbus (protokol industri). Dapat menjalankan simulasi komunikasi RS-485.

- Python

  Digunakan sebagai bahasa pemrograman tambahan dalam berbagai automasi dan komunikasi perangkat IoT. Disarankan menginstal versi 3.x, dan pastikan 'Add to PATH' dicentang saat instalasi.

- MQTT X Client

  Alat untuk komunikasi IoT berbasis protokol MQTT (ringan dan real-time). Memungkinkan publish/subscribe antar perangkat.

- XAMPP

  Web server lokal (Apache) dan sistem basis data (MySQL/phpMyAdmin) untuk menyimpan atau menampilkan data IoT secara lokal.

- Microcontroller ESP32 untuk Device IoT

  ESP32 adalah mikrokontroler buatan Espressif Systems yang dilengkapi dengan modul WiFi dan Bluetooth bawaan. Mikrokontroler ini merupakan penerus dari ESP8266 dan lebih canggih untuk aplikasi IoT. ESP32 tersedia dalam berbagai form factor

Tabel 2.1 Jenis Dan Fungsi Pin Pada ESP32

Jenis Pin	Fungsi	Contoh Perangkat
GPIO	Input/Output umum	DHT11, HC-SR04
ADC	Membaca sinyal analog	Sensor kelembaban tanah, MQ-2
PWM	Kontrol sinyal lebar pulsa	LED, Motor Servo
UART	Komunikasi serial (RX/TX)	RS-485 ke UART, SIM800 GSM
I2C	Komunikasi serial sinkron 2-pin (SCL/SDA)	LCD 16x2, RTC DS3231
SPI	Komunikasi serial full-duplex 4-pin	Lora SX1278, SD Card Module

Perbedaan Sinyal Analog dan Digital

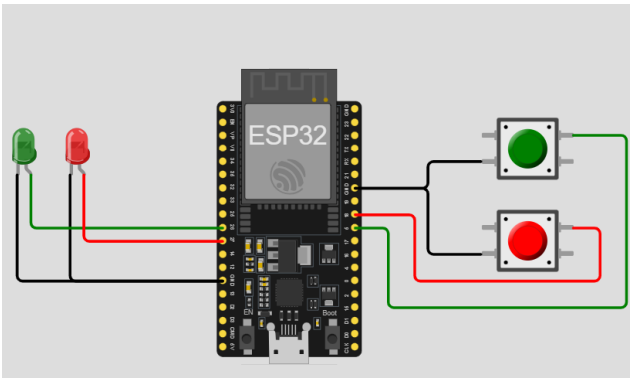
- Sinyal analog: kontinu dan bisa bernilai apa saja dalam rentang tertentu
- Sinyal digital: diskrit, hanya 0 atau 1

Protokol Komunikasi Serial, ESP32 mendukung tiga protokol komunikasi utama:

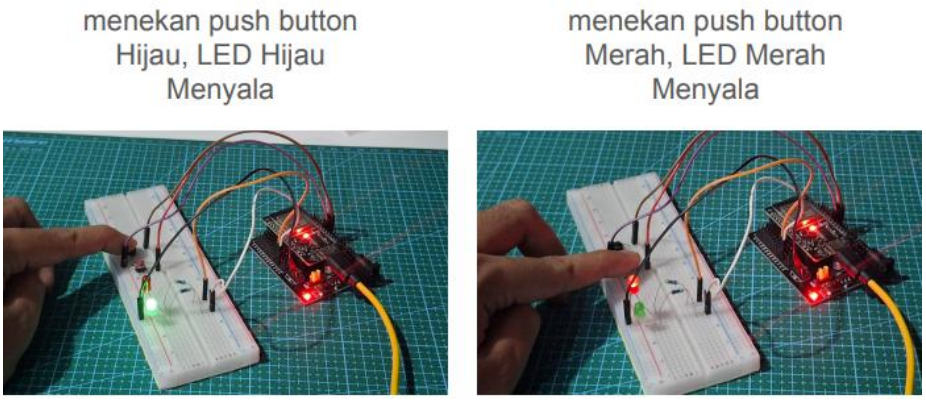
1. UART: Komunikasi serial asinkron (2 pin: RX dan TX)
2. I2C: Komunikasi serial sinkron half-duplex (2 pin: SCL dan SDA)
3. SPI: Komunikasi serial sinkron full-duplex (4 pin: SCLK, MISO, MOSI, CS)

c. Digital Input Push Button dan Digital Output LED

Membuat program pada ESP32 untuk membaca sinyal digital dari Push Button, serta memerintahkan sinyal digital untuk menyalakan Lampu LED



Gambar 2.1 Simulasi "Push Button dan LED" menggunakan WOKWI



Gambar 2.2 Rangkaian Push Button dan LED

Pin yang digunakan:

- D27 → LED Merah
- D14 → LED Hijau
- D18 → Push Button Merah
- D5 → Push Button Hijau
- GND untuk ground
- Resistor 480Ω untuk membatasi arus ke LED

```
// Mendefinisikan pin yang digunakan untuk LED dan tombol
#define led_merah 27     // LED merah terhubung ke pin 27
#define led_hijau 14     // LED hijau terhubung ke pin 14
#define bt_merah 18      // Tombol merah terhubung ke pin 18
#define bt_hijau 5       // Tombol hijau terhubung ke pin 5

void setup() {
  Serial.begin(9600);             // Memulai komunikasi serial dengan baudrate 9600
  pinMode(led_merah, OUTPUT);     // Mengatur pin LED merah sebagai output
  pinMode(led_hijau, OUTPUT);     // Mengatur pin LED hijau sebagai output
  pinMode(bt_merah, INPUT_PULLUP); // Mengatur pin tombol merah sebagai input dengan resistor pull-up internal
  pinMode(bt_hijau, INPUT_PULLUP); // Mengatur pin tombol hijau sebagai input dengan resistor pull-up internal
}

void loop() {
  // Membaca status tombol. Karena menggunakan INPUT_PULLUP, tombol akan terbaca LOW saat ditekan.
  int ledState_merah = !digitalRead(bt_merah);  // Jika tombol merah ditekan (LOW), maka hasilnya HIGH (LED menyala)
  int ledState_hijau = !digitalRead(bt_hijau);  // Jika tombol hijau ditekan (LOW), maka hasilnya HIGH (LED menyala)

  // Mengatur status LED sesuai tombol yang ditekan
  digitalWrite(led_merah, ledState_merah);   // Nyalakan atau matikan LED merah
  digitalWrite(led_hijau, ledState_hijau);   // Nyalakan atau matikan LED hijau

  delay(10); // Delay kecil untuk menghindari bouncing tombol
}
```

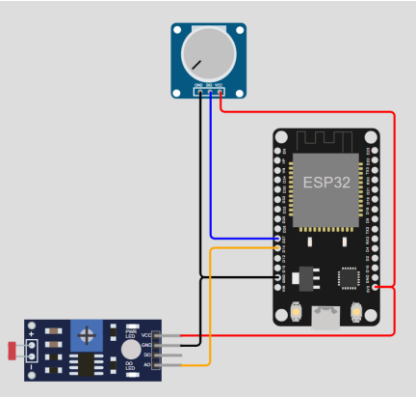
Gambar 2.3 Kode Program Push Button dan LED

Program ini mengontrol dua LED berdasarkan dua tombol:

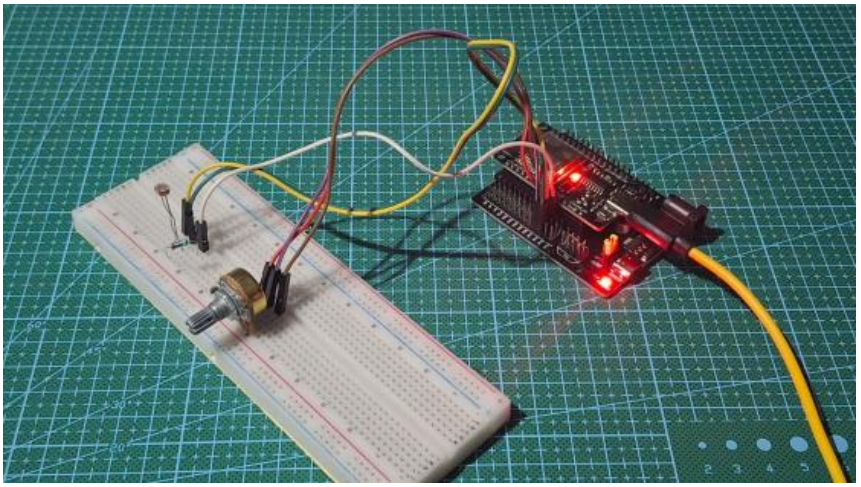
- Jika **tombol merah** ditekan, **LED merah** menyala.
- Jika **tombol hijau** ditekan, **LED hijau** menyala.

d. Potensiometer dan Sensor LDR — Testing ADC (Analog-to-Digital Converter)

Membuat program pada ESP32 untuk membaca sinyal analog dari Potensiometer dan Light Dependent Resistor (LDR)



Gambar 2.4 Skematik Potensiometer dan LDR



Gambar 2.5 Rangkaian Potensiometer dan LDR

2_Potensiometer_dan_LDR

```
// Mendefinisikan pin analog yang digunakan untuk membaca nilai dari potensiometer dan sensor LDR
#define potensio_pin 27   // Potensiometer terhubung ke pin analog 27
#define ldr_pin 14        // Sensor LDR terhubung ke pin analog 14

void setup() {
  Serial.begin(9600);               // Memulai komunikasi serial dengan baudrate 9600
  pinMode(potensio_pin, INPUT);     // Mengatur pin potensiometer sebagai input
  pinMode(ldr_pin, INPUT);          // Mengatur pin LDR sebagai input
}

void loop() {
  // Membaca nilai analog dari potensiometer dan LDR
  int potensioValue = analogRead(potensio_pin); // Membaca nilai tegangan dari potensiometer (0-4095 untuk ESP32)
  int ldrValue = analogRead(ldr_pin);           // Membaca nilai cahaya dari sensor LDR (0-4095 untuk ESP32)

  // Menampilkan nilai hasil pembacaan ke Serial Monitor
  Serial.print("potensio Value: ");
  Serial.println(potensioValue);
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);

  delay(500); // Delay 500 ms sebelum membaca ulang
}
```

Gambar 2.6 Kode Program Potensiometer dan LDR

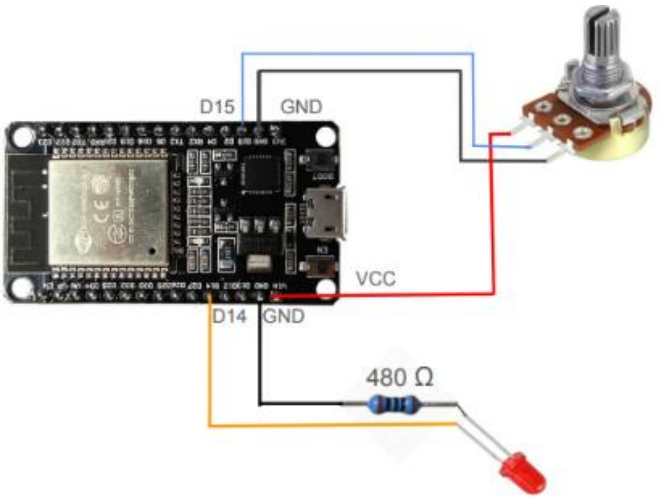
Program ini membaca dan menampilkan nilai analog dari:

- Potensiometer – memberikan nilai berdasarkan posisi putaran (biasanya 0 sampai 4095 di ESP32).
- Sensor LDR (Light Dependent Resistor) – memberikan nilai tergantung intensitas cahaya (semakin terang, nilai biasanya makin kecil atau besar tergantung rangkaian).

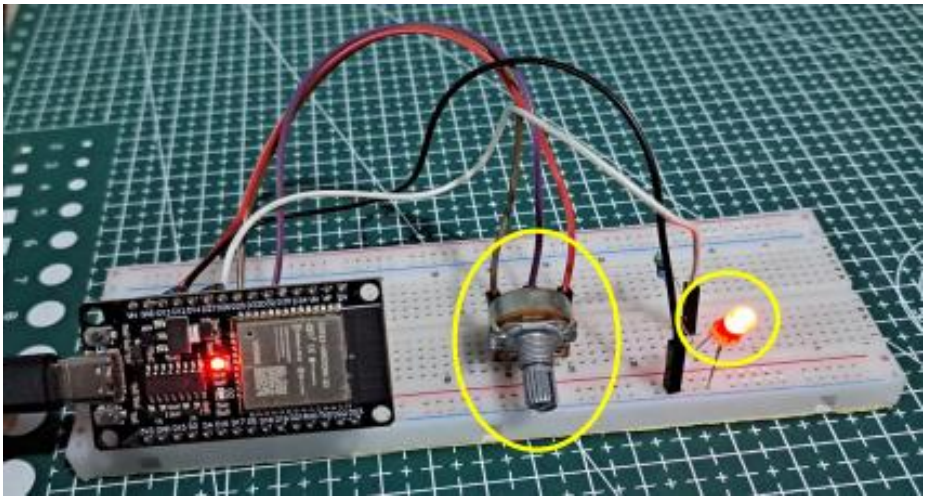
- Nilai-nilai ini ditampilkan ke Serial Monitor setiap 500 milidetik, sehingga bisa memantau perubahan nilai secara real-time.

e. Pulse Width Modulation (PWM) dan Analog Output

PWM (Pulse Width Modulation) adalah teknik yang sering digunakan untuk mengontrol tingkat daya yang diberikan ke suatu perangkat, seperti LED, Motor, Kipas, Servo, dan perangkat lain. PWM bekerja dengan cara mengatur lebar pulsa sinyal digital untuk mensimulasikan sinyal analog. Pada ESP32, PWM dapat digunakan untuk mengontrol intensitas cahaya LED. Lampu otomatis dengan Sensor LDR. Peserta belajar membuat Sistem Cerdas Sederhana yang mengkombinasikan ESP32 – Sensor LDR – Potensiometer – Relay/Lampu



Gambar 2.7 Wiring PWM Analog Output



Gambar 2.8 Rangkaian PWM Analog Output

```
int ledPin = 14;              // Pin LED yang akan dikontrol dengan PWM
int potensiometer = 15;       // Pin input potensiometer
int value = 0;                // Variabel untuk menyimpan nilai analog dari potensiometer
int pwmValue = 0;             // Variabel untuk menyimpan nilai PWM hasil pemetaan

void setup() {
  Serial.begin(115200);       // Memulai komunikasi serial dengan baudrate 115200
  pinMode(ledPin, OUTPUT);    // Mengatur pin LED sebagai output
}

void loop() {
  value = analogRead(potensiometer);   // Membaca nilai analog dari potensiometer (0 - 4095 untuk ESP32)
  Serial.print("Analog Value: ");
  Serial.println(value);

  int pwmValue = map(value, 0, 4095, 0, 255); // Memetakan nilai 12-bit ke skala 8-bit untuk PWM (0-255)
  Serial.print("PWM Value: ");
  Serial.println(pwmValue);

  analogWrite(ledPin, pwmValue); // Mengatur tingkat kecerahan LED berdasarkan nilai PWM
}
```

Gambar 2.9 Kode Program PWM Analog Output

Fungsi Program:

- Baca nilai potensiometer (0–4095).
- Ubah nilai tersebut menjadi skala PWM (0–255).
- Atur kecerahan LED dengan PWM berdasarkan nilai potensiometer.
- Tampilkan nilai di Serial Monitor.

f. Mengukur Jarak/Ketinggian dengan Sensor Ultrasonic HC-SR04

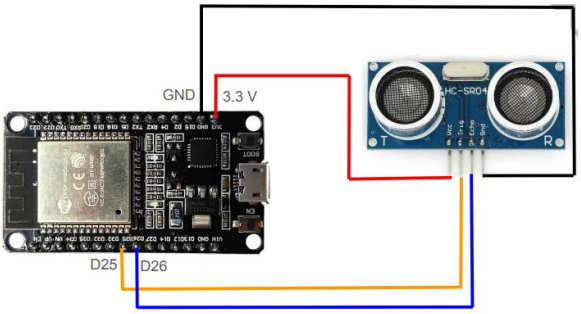
Sensor ultrasonik HC-SR04 adalah perangkat yang digunakan untuk mengukur jarak dengan cara memanfaatkan gelombang ultrasonik. Sensor ini terdiri dari dua bagian utama:

1. Transmitter: Mengeluarkan gelombang ultrasonik dengan frekuensi sekitar 40 kHz.
2. Receiver: Menerima gelombang ultrasonik yang dipantulkan kembali dari objek yang berada di depannya.

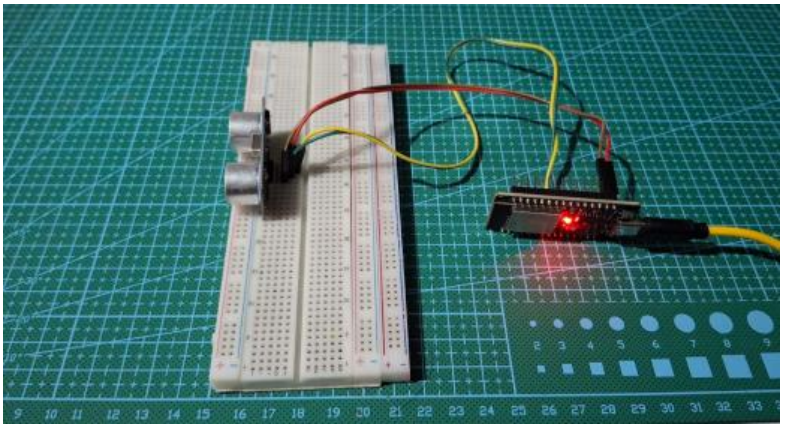
Cara kerja sensor ini adalah sebagai berikut:

1. Pengiriman Gelombang: Transmitter mengirimkan pulsa ultrasonik.
2. Pantulan: Gelombang tersebut mengenai objek dan dipantulkan kembali ke sensor.
3. Penerimaan: Receiver menangkap gelombang yang dipantulkan.
4. Pengukuran Waktu: Sensor mengukur waktu yang dibutuhkan gelombang untuk kembali.
5. Perhitungan Jarak: Berdasarkan waktu perjalanan gelombang, sensor menghitung jarak antara sensor dan objek.

Membuat program yang menampilkan jarak objek yang terdeteksi sensor ultrasonik dalam sentimeter dan inchi, setiap 1 detik.



Gambar 2.10 Wiring Mengukur Jarak dengan Sensor Ultrasonik



Gambar 2.11 Rangkaian Mengukur Jarak dengan Sensor Ultrasonik

```
// Mendefinisikan pin untuk sensor ultrasonik
#define PIN_TRIG 25  // Pin trigger untuk mengirim sinyal ultrasonik
#define PIN_ECHO 26  // Pin echo untuk menerima pantulan sinyal

void setup() {
  Serial.begin(115200);         // Memulai komunikasi serial pada baudrate 115200
  pinMode(PIN_TRIG, OUTPUT);    // Mengatur pin TRIG sebagai output
  pinMode(PIN_ECHO, INPUT);     // Mengatur pin ECHO sebagai input
}

void loop() {
  // Mengirim sinyal trigger selama 10 mikrodetik
  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  // Membaca durasi pantulan sinyal ultrasonik dalam mikrodetik
  int duration = pulseIn(PIN_ECHO, HIGH);

  // Menghitung jarak berdasarkan durasi (dalam cm dan inci)
  Serial.print("Jarak dalam CM: ");
  Serial.println(duration / 58);      // Rumus jarak dalam cm (kecepatan suara dibagi 2)

  Serial.print("Jarak dalam Inche: ");
  Serial.println(duration / 148);     // Rumus jarak dalam inci

  delay(1000);  // Delay 1 detik sebelum pembacaan berikutnya
}
```

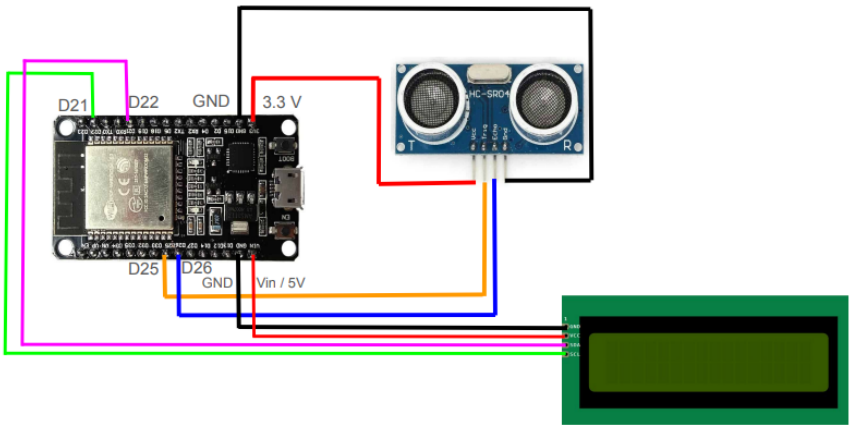
Gambar 2.12 Kode Porgram Rangkaian Mengukur Jarak dengan Sensor Ultrasonik

Penjelasan :

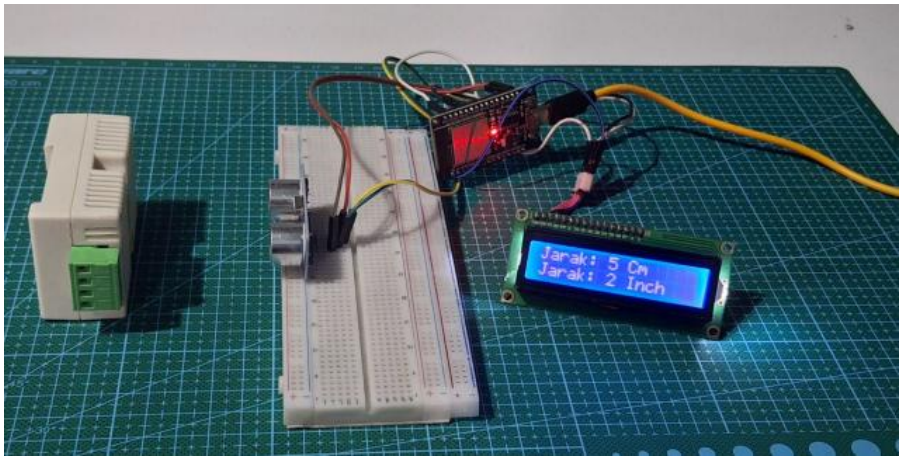
- Sensor HC-SR04 bekerja dengan mengirim gelombang ultrasonik, lalu menghitung waktu yang dibutuhkan sampai gelombang dipantulkan kembali oleh objek.
- Waktu yang didapat dari pulseIn() adalah waktu pulang-pergi, jadi perlu dibagi dua saat menghitung jarak. Faktor 58 dan 148 sudah memperhitungkan ini.

g. Monitoring Jarak dengan sensor HC-SR04 pada LCD 16x2

Mengukur jarak objek menggunakan sensor ultrasonik HC-SR04 dan menampilkannya dalam sentimeter (cm) dan inci (inch) di layar LCD I2C 16x2.



Gambar 2.13 Wiring Monitoring Jarak dengan Ultrasonic HC-SR04 pada LCD 16x2 I2C



Gambar 2.14 Rangkaian Monitoring Jarak dengan Ultrasonic HC-SR04 pada LCD 16x2 I2C

5_Mengukur_Jarak_dengan_Sensor_Ultrasonic

```
#include <LiquidCrystal_I2C.h>  // Library untuk LCD I2C

// Definisi pin HC-SR04
#define PIN_TRIG 25  // Pin trigger sensor ultrasonik
#define PIN_ECHO 26  // Pin echo sensor ultrasonik

// Konfigurasi LCD I2C
#define I2C_ADDR    0x27      // Alamat I2C LCD (umumnya 0x27 atau 0x3F)
#define LCD_COLUMNS 16        // Jumlah kolom LCD
#define LCD_LINES   2         // Jumlah baris LCD

LiquidCrystal_I2C lcd(I2C_ADDR, LCD_COLUMNS, LCD_LINES); // Inisialisasi objek LCD

void setup() {
  Serial.begin(115200);           // Mulai komunikasi serial untuk debugging
  pinMode(PIN_TRIG, OUTPUT);      // Set pin trigger sebagai output
  pinMode(PIN_ECHO, INPUT);       // Set pin echo sebagai input
  lcd.init();                     // Inisialisasi LCD
  lcd.backlight();                // Menyalakan backlight LCD
}

void loop() {
  lcd.clear();                    // Bersihkan layar sebelum menampilkan data baru

  // Mengirimkan sinyal trigger 10 mikrodetik
  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  // Membaca durasi pantulan sinyal ultrasonik
  int duration = pulseIn(PIN_ECHO, HIGH);

  // Menampilkan hasil pengukuran ke LCD
  lcd.setCursor(0, 0);  // Posisi baris pertama, kolom pertama
  lcd.print("Jarak: " + String(duration / 58) + " Cm");

  lcd.setCursor(0, 1);  // Posisi baris kedua, kolom pertama
  lcd.print("Jarak: " + String(duration / 148) + " Inch");

  delay(1000);          // Tunggu 1 detik sebelum mengulang
}
```

Gambar 2.15 Kode Program Monitoring Jarak dengan Ultrasonic HC-SR04 pada LCD 16x2 I2C

Fungsi: Mengirim sinyal **ultrasonik** dari sensor.

- Pin TRIG diatur ke HIGH selama 10 mikrodetik.
- Ini memberi perintah pada sensor untuk mengeluarkan pulsa suara ultrasonik.
- Setelah itu TRIG dimatikan (LOW) kembali.
- **Durasi 10 mikrodetik** adalah standar untuk mengaktifkan HC-SR04.

- **lcd.setCursor(0, 0);**

  - Mengatur **kursor LCD** pada **kolom ke-0, baris ke-0** (pojok kiri atas).
  - Artinya, teks yang ditampilkan di baris pertama LCD akan mulai dari karakter pertama.

- **lcd.print("Jarak: " + String(duration / 58) + " Cm");**

  - Menampilkan teks dan hasil perhitungan **jarak dalam cm**.
  - duration / 58 mengubah durasi pantulan (dalam mikrodetik) menjadi satuan **centimeter**.
  - String(...) mengkonversi angka menjadi teks agar bisa digabung dengan "Jarak: " dan " Cm".

- **lcd.setCursor(0, 1);**

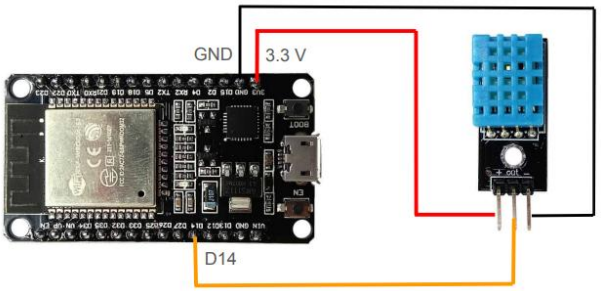
  - Mengatur **kursor pada baris kedua** LCD (kolom pertama, baris ke-1).

- **lcd.print("Jarak: " + String(duration / 148) + " Inch");**

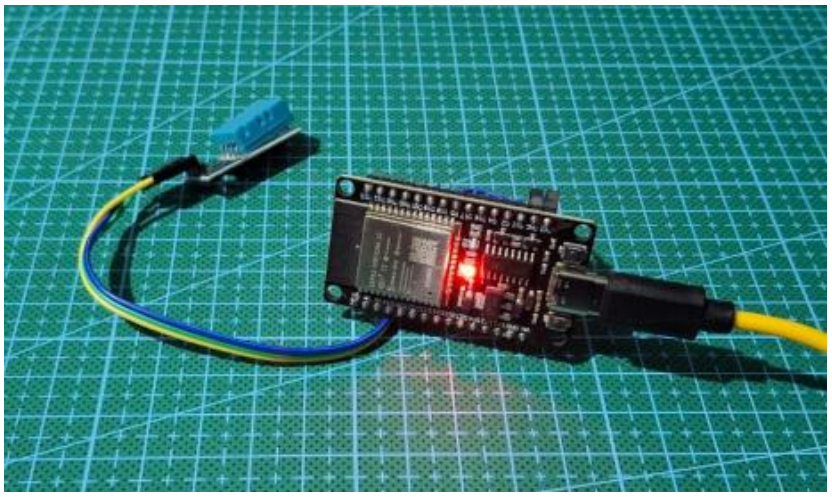
  - Menampilkan jarak dalam **inchi**, menggunakan rumus duration / 148.
  - Ini berdasarkan kecepatan suara untuk satuan inchi.

h. Monitoring Suhu dan Kelembaban menggunakan Sensor DHTII

Mengenal Sensor Suhu dan Kelembaban DHT11, peserta belajar bagaimana cara berinteraksi dengan sensor tersebut



Gambar 2.16 Wiring Monitoring Suhu dan Kelembapan dengan Sensor DHT11



Gambar 2.17 Rangkaian Sensor Suhu dan Kelembapan DHT11

```
#include "DHT.h"  // Library untuk sensor DHT (mendukung DHT11, DHT22, dll)

// Mendefinisikan pin dan tipe sensor
#define DHTPIN 14         // Pin digital tempat sensor DHT22 terhubung
#define DHTTYPE DHT22     // Tipe sensor: DHT11 atau DHT22

// Membuat objek dht dengan pin dan tipe yang sudah didefinisikan
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);            // Memulai komunikasi serial dengan kecepatan 9600 bps
  Serial.println(F("DHTxx test!")); // Menampilkan pesan awal di Serial Monitor
  dht.begin();                   // Menginisialisasi sensor DHT
}

void loop() {
  // Membaca kelembaban dari sensor
  float h = dht.readHumidity();

  // Membaca suhu dari sensor (default: dalam Celcius)
  float t = dht.readTemperature();

  // Mengecek apakah pembacaan gagal (nilai tidak valid = NaN)
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!")); // Tampilkan pesan error
    return;  // Keluar dari loop untuk mencoba membaca lagi nanti
  }

  // Menampilkan kelembaban dan suhu ke Serial Monitor
  Serial.print(F("Humidity: "));
  Serial.print(h);               // Tampilkan nilai kelembaban
  Serial.print(F("%  Temperature: "));
  Serial.println(t);             // Tampilkan nilai suhu

  delay(2000); // Tunggu 2 detik sebelum pembacaan ulang (disarankan untuk DHT)
}
```

Gambar 2.18 Kode Program Sensor Suhu dan Kelembapa DHT11

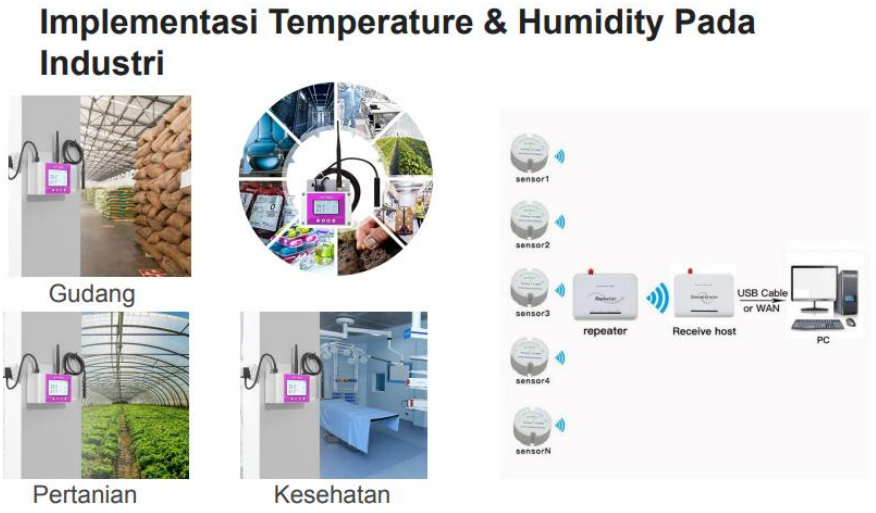
Fungsi Program Menggunakan sensor DHT11 untuk mengukur:

- Suhu dalam Celcius
- Kelembaban dalam persen (%)
- Menampilkan hasil ke Serial Monitor setiap 2 detik.
- Jika sensor gagal dibaca (misalnya tidak terhubung), akan muncul pesan kesalahan.

i. Sistem Monitoring Suhu dan Kelembaban Local Web-Base WiFi Client dan Data Logger.

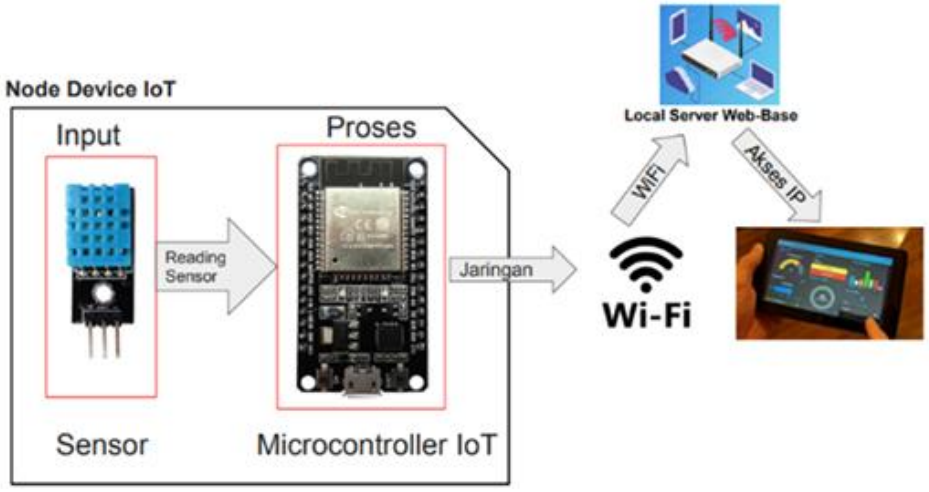
Peserta membuat web server lokal pada ESP32 yang dapat

menampilkan nilai suhu dan kelembaban dari sensor DHT11 begitu juga untuk pengumpulan datanya (data logging), contoh penerapan sistem monitorung suhu dan kelembapan :



Gambar 2.19 Contoh Implementasi Temperature & Humidity pada Industri

Topologi yang digunakan :



Gambar 2.20 Topologi Local Web Base Data Logger

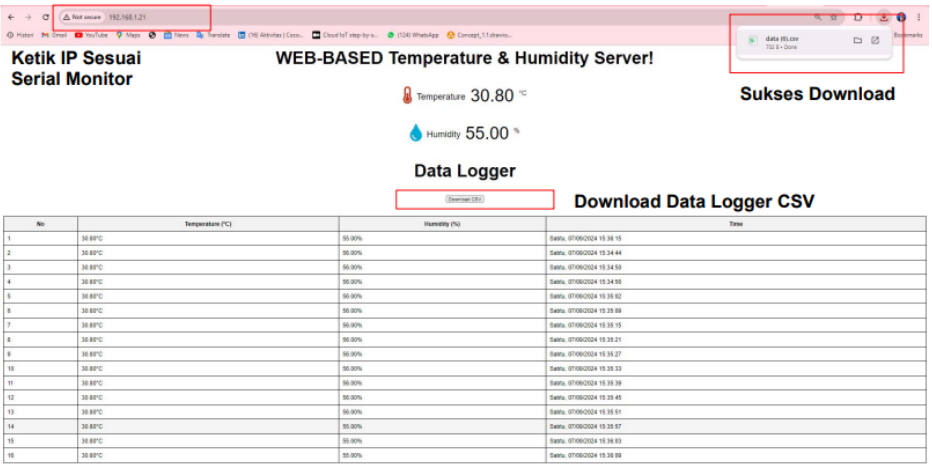
Kode program :

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

const char *ssid = "BDIYOGYAKARTA";
const char *password = "BDIBDIYK";

WebServer server(80);
DHT dht(14, DHT11);

void setup(void) {

  Serial.begin(115200);
  dht.begin();

  WiFi.mode(WIFI_STA);
  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());

  // DNS untuk domain pengganti IP
  if (MDNS.begin("esp32")) {
    Serial.println("MDNS responder started");
  }
  server.on("/", handleRoot);

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
  delay(2);
}
```

Gambar 2.21 Kode Program Sistem Monitoring Suhu dan Kelembaban Local Web-Base WiFi Client dan Data Logger

```
float readDHTTemperature() {
  // Sensor readings may also be up to 2 seconds
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return -1;
  } else {
    Serial.println(t);
    return t;
  }
}

float readDHTHumidity() {
  // Sensor readings may also be up to 2 seconds
  float h = dht.readHumidity();
  if (isnan(h)) {
    Serial.println("Failed to read from DHT sensor!");
    return -1;
  } else {
    Serial.println(h);
    return h;
  }
}

void handleRoot() {
  String html_page = R"rawliteral(
<html>
  <head>
    <meta http-equiv='refresh' content='4'/>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <link rel='stylesheet' href='https://use.fontawesome.com/releases/v5.7.2/css/all.css'
          integrity='sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr'
          crossorigin='anonymous'>
    <title>ESP32 DHT Server</title>
    <style>
      html { font-family: Arial; display: inline-block; margin: 0px auto; text-align: center;}
      h2 { font-size: 3.0rem; }
      p { font-size: 3.0rem; }
      .units { font-size: 1.2rem; }
      .dht-labels { font-size: 1.5rem; vertical-align:middle; padding-bottom: 15px;}
    </style>
  </head>
  <body>
    <h2>Local Web-Base ESP32 Temperature & Humidity!</h2>
    <p>


     <p>
       <i class='fas fa-tint' style='color:#00add6;'></i>
       <span class='dht-labels'>Humidity</span>
       <span>%HUMIDITY%</span>
       <sup class='units'>&percnt;</sup>
     </p>
  </body>
</html>
)rawliteral";

  html_page.replace("%TEMPERATURE%", String(readDHTTemperature(), 2));
  html_page.replace("%HUMIDITY%", String(readDHTHumidity(), 2));

  server.send(200, "text/html", html_page);
}
```

Gambar 2.22 Lanjutan Kode Program Sistem Monitoring Suhu dan Kelembaban Local Web-Base WiFi Client dan Data Logger



Gambar 2.23 Tampilan Hasil Sistem Monitoring Suhu dan Kelembaban Local Web-Base WiFi Client dan Data Logger

j. Pengenalan protocol MQTT dan HTTP

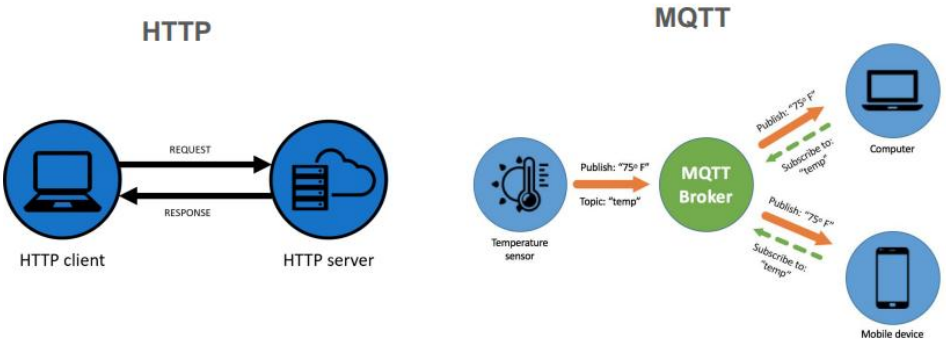
Message Queuing Telemetry Transport (MQTT) adalah protokol komunikasi ringan yang digunakan untuk mentransfer data antar perangkat melalui jaringan TCP/IP. MQTT seringnya digunakan untuk komunikasi machine-to-machine (M2M) dan perangkat IoT, MQTT menjadi solusi pada perangkat IoT, yang mana mengirim dan menerima data melalui jaringan dengan sumber daya dan bandwidth terbatas. Pihak – pihak dalam MQTT

1. Publisher, pihak yang mengirim/publish data
2. Subscriber, pihak yang meminta/menerima data
3. Broker, pihak yang menjadi mediator antara publisher dan subscriber

Jenis data yang dikirim protokol ini adalah jenis protokol data-agnostic, artinya bisa mengirimkan data apapun seperti data angka, text ataupun JSON.

Tabel 2.2 HTTP vs MQTT

Aspek	HTTP	MQTT
Protokol dan Tujuan	Protokol berbasis request - respons untuk transfer data web	Protokol messaging ringan untuk komunikasi M2M dan IoT.
Overhead dan Efisiensi	Overhead besar dengan header yang panjang.	Overhead kecil, header minimal, efisiensi bandwidth tinggi.
Keandalan dan QoS	Tidak ada mekanisme QoS bawaan, data tidak selalu terjamin sampai.	Menyediakan tiga tingkat QoS (0, 1, 2) untuk keandalan pesan.
Konektivitas dan Latensi	Memerlukan koneksi TCP untuk setiap permintaan-respons; latensi lebih tinggi.	Memelihara koneksi TCP terbuka; latensi lebih rendah.
Penggunaan dan Aplikasi	Aplikasi web, RESTful APIs, interaksi pengguna berbasis permintaan-respons.	Aplikasi IoT, sensor, komunikasi real-time, pengiriman pesan asinkron.



Gambar 2.24 Topologi Perbandingan HTTP dan MQTT

k. ESP32 Publish dan Subscribe dengan Protocol MQTT

Mengimplementasikan MQTT pada ESP32 untuk mengirim data (Publish) serta menerima data (subscribe)

l. MODBUS PROTOCOL (Industrial protocol)

Modbus adalah protokol komunikasi yang digunakan untuk menghubungkan perangkat elektronik Industri. Modbus dirilis pada 1979 oleh Modicon untuk diimplementasikan ke dalam Programmable Logic Controller (PLC). Jenis

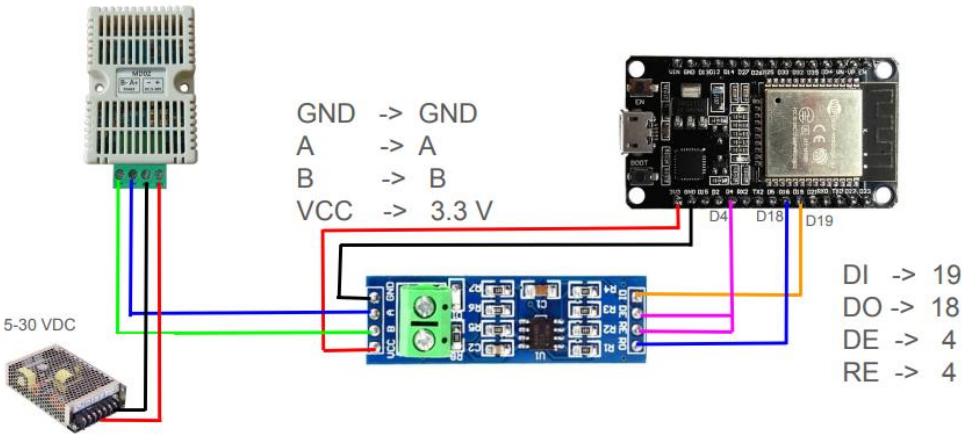
1. Modbus RTU, berjalan diatas Port Serial RS-232 dan RS-485
2. Modbus ASCII, menggunakan aturan ASCII (American Standard Code for Information Interchange)
3. Modbus TCP, berjalan diatas Stack TCP/IP dan Port Ethernet

Karakteristik Modbus

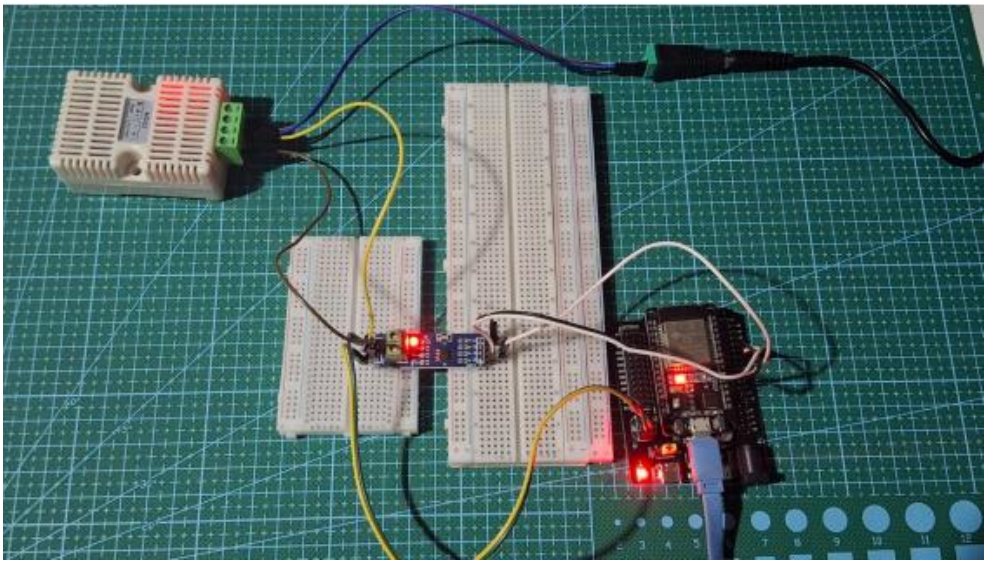
- Menggunakan sistem polling (master meminta, slave merespons)
- Error checking pada setiap frame data
- Hanya data yang diminta yang dikirimkan

m. Modbus Master untuk membaca Sensor XY-MD02 dengan ESP32

ESP32 bawaannya tidak memiliki Pin RS-485 namun, dengan bantuan converter MAX485, ESP32 bisa terhubung ke device dengan Port RS-485 dan berkomunikasi dengan Protokol Modbus RTU. Peserta akan mencoba untuk membaca data dari sensor SHT20 / XY-MD02 menggunakan ESP32.



Gambar 2.25 Wiring Membaca Sensor Modbus Rtu Rs-485 Dengan Esp32



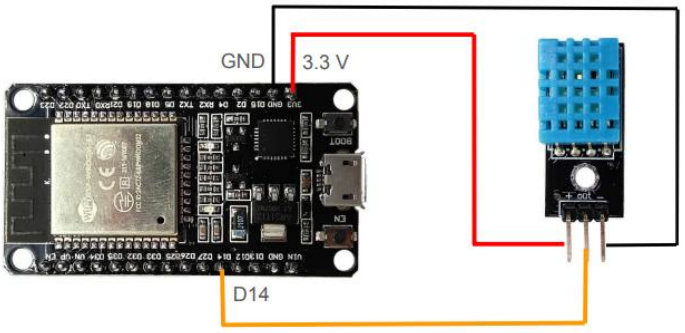
Gambar 2.26 Rangkaian Membaca Sensor Modbus Rtu Rs-485 Dengan Esp32

```
#include <ModbusMaster.h> // Library untuk komunikasi Modbus RTU

// Definisi pin-pinnya
#define MODBUS_EN_PIN  4     // Pin untuk mengontrol arah komunikasi (DE & RE pada RS485)
#define MODBUS_RO_PIN 18     // Pin RX (RO dari modul RS485 ke ESP)
#define MODBUS_DI_PIN 19     // Pin TX (DI dari ESP ke modul RS485)
#define MODBUS_SERIAL_BAUD 9600          // Baudrate komunikasi serial
#define MODBUS_PARITY SERIAL_8N1         // Format parity: 8 data bit, No parity, 1 stop bit

// Konfigurasi Modbus
#define MODBUS_SLAVE_ID 1         // Alamat ID slave yang dituju
#define MODBUS_ADDRESS 0x0001     // Alamat register pertama yang ingin dibaca
#define MODBUS_QUANTITY 2         // Jumlah register yang dibaca (2 buah)

ModbusMaster modbus; // Objek ModbusMaster untuk komunikasi

// Fungsi yang dipanggil sebelum transmisi data Modbus (set RS485 ke mode kirim)
void modbusPreTransmission()
{
  delay(500); // Delay untuk memberi waktu sebelum kirim (terlalu lama, bisa dioptimasi)
  digitalWrite(MODBUS_EN_PIN, HIGH); // Aktifkan mode TX (DE & RE = HIGH)
}

// Fungsi yang dipanggil setelah transmisi selesai (set RS485 ke mode terima)
void modbusPostTransmission()
{
  digitalWrite(MODBUS_EN_PIN, LOW); // Aktifkan mode RX (DE & RE = LOW)
  delay(500); // Delay untuk memberi waktu sebelum menerima data (terlalu lama)
}

void setup()
{
  Serial.begin(115200); // Inisialisasi serial monitor untuk debug

  // Atur pin kontrol RS485 sebagai output
  pinMode(MODBUS_EN_PIN, OUTPUT);
  digitalWrite(MODBUS_EN_PIN, LOW); // Mulai dalam mode terima

  // Inisialisasi Serial2 untuk komunikasi Modbus (ESP32 memiliki hardware serial tambahan)
  Serial2.begin(MODBUS_SERIAL_BAUD, MODBUS_PARITY, MODBUS_RO_PIN, MODBUS_DI_PIN);
  Serial2.setTimeout(1000); // Timeout untuk pembacaan serial

  // Hubungkan objek Modbus ke serial yang digunakan dan set ID slave
  modbus.begin(MODBUS_SLAVE_ID, Serial2);



  // Tetapkan fungsi untuk pre dan post transmission
  modbus.preTransmission(modbusPreTransmission);
  modbus.postTransmission(modbusPostTransmission);
}

void loop()
{
  int pooling;           // Variabel untuk menyimpan status hasil pembacaan Modbus
  int hasil[2];          // Array untuk menyimpan 2 nilai register yang diterima
  float temperature;     // Variabel suhu
  float humidity;        // Variabel kelembapan

  // Kirim permintaan baca register input dari alamat MODBUS_ADDRESS sebanyak MODBUS_QUANTITY
  pooling = modbus.readInputRegisters(MODBUS_ADDRESS, MODBUS_QUANTITY);

  // Jika pembacaan berhasil (pooling == ku8MBSuccess)
  if (pooling == modbus.ku8MBSuccess) {
    Serial.println("Success, Data diterima: ");

    // Ambil data dari response buffer
    hasil[0] = modbus.getResponseBuffer(0x00); // Data pertama (misal suhu)
    hasil[1] = modbus.getResponseBuffer(0x01); // Data kedua (misal kelembapan)

    // Konversi data dari integer ke float dengan skala 0.1
    temperature = hasil[0] / 10.f;
    humidity = hasil[1] / 10.f;

    // Tampilkan hasil ke Serial Monitor
    Serial.println("Temperature: " + String(temperature));
    Serial.println("Humidity: " + String(humidity));
    Serial.println();
  }
  else {
    // Jika gagal membaca data
    Serial.println("GAGAL membaca data");
  }

  delay(1000); // Delay 1 detik sebelum pembacaan berikutnya
}
```

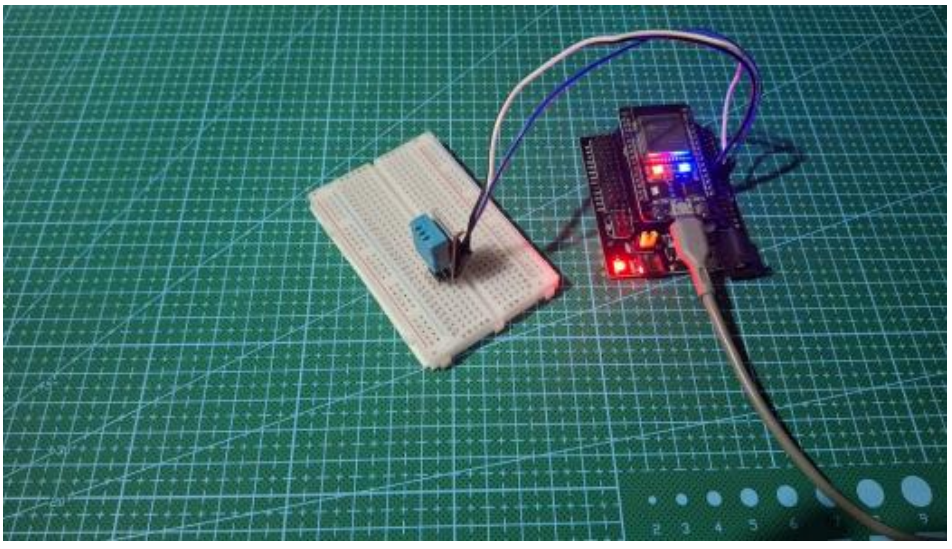
Gambar 2.27 Kode Program Modbus Master Untuk Membaca Sensor XY-MD02 Dengan ESP32

n. Membuat Sensor Modbus TCP/IP menggunakan ESP32

Selain membaca sensor modbus, kita juga bisa membuat sensor modbus sendiri. kali ini kita akan membuat sebuah Sensor Suhu dan Kelembaban menggunakan protokol Modbus TCP/IP. dengan adanya fitur WiFi pada ESP32, bisa memanfaatkannya sebagai media transmisi modbus TCP/IP.



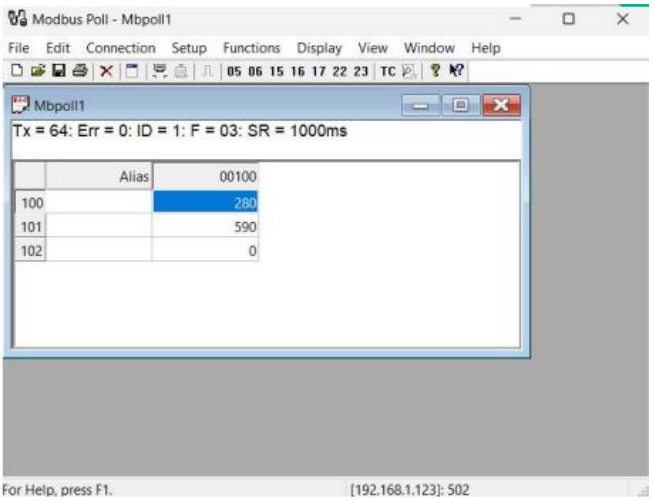
Gambar 2.28 Wiring Membuat Sensor TCP/IP Menggunakan ESP32



Gambar 2.29 Rangkaian Membuat Sensor TCP/IP Menggunakan ESP32

```
#include <WiFi.h>                   // Library WiFi untuk ESP32
#include <ModbusIP_ESP8266.h>       // Library Modbus TCP (juga kompatibel dengan ESP32)
#include "DHT.h"                    // Library sensor DHT

// Setup pin
#define LED_PIN 2                   // Pin LED internal ESP32
#define DHTPIN 14                   // Pin data sensor DHT11
#define DHTTYPE DHT11               // Tipe sensor DHT

// Alamat register Modbus (holding register)
#define TEMPERATURE_ADDRESS 100     // Alamat untuk suhu
#define HUMIDITY_ADDRESS 101        // Alamat untuk kelembapan
#define LED_ADDRESS 102             // Alamat untuk kontrol LED

// Data WiFi
const char* ssid = "BDIYOGYAKARTA";
const char* pass = "BDIBDIYK";

// Alamat IP statis
IPAddress local_IP(192, 168, 1, 123);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);

// Inisialisasi sensor DHT dan objek Modbus
DHT dht(DHTPIN, DHTTYPE);
ModbusIP mb;  // Objek utama untuk server Modbus TCP

void setup() {
  Serial.begin(115200);
  Serial.println(F("MODBUS TCP OVER WIFI ESP32"));

  // Konfigurasi IP statis dan koneksi WiFi
  WiFi.config(local_IP, gateway, subnet);
  WiFi.begin(ssid, pass);



  // Tunggu sampai koneksi WiFi berhasil
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi Terhubung!");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  // Inisialisasi pin dan sensor
  pinMode(LED_PIN, OUTPUT);
  dht.begin();

  // Mulai server Modbus TCP
  mb.server();

  // Tambahkan register Modbus untuk suhu, kelembapan, dan LED
  mb.addHreg(TEMPERATURE_ADDRESS);  // Register 100
  mb.addHreg(HUMIDITY_ADDRESS);     // Register 101
  mb.addHreg(LED_ADDRESS);          // Register 102
}

void loop() {
  // Baca suhu dan kelembapan dari sensor DHT
  float humidity = dht.readHumidity();          // Kelembapan dalam %
  float temperature = dht.readTemperature();    // Suhu dalam °C

  // Simpan nilai ke register Modbus setelah dikalikan 10 (agar tetap integer)
  mb.Hreg(TEMPERATURE_ADDRESS, temperature * 10); // Misal: 25.4°C → 254
  mb.Hreg(HUMIDITY_ADDRESS, humidity * 10);       // Misal: 60.5% → 605

  // Proses permintaan dari client Modbus TCP
  mb.task();

  // Kontrol LED dari register Modbus (1 = ON, 0 = OFF)
  digitalWrite(LED_PIN, mb.Hreg(LED_ADDRESS));

  delay(50); // Delay kecil agar CPU tidak 100% terus-menerus
}
```

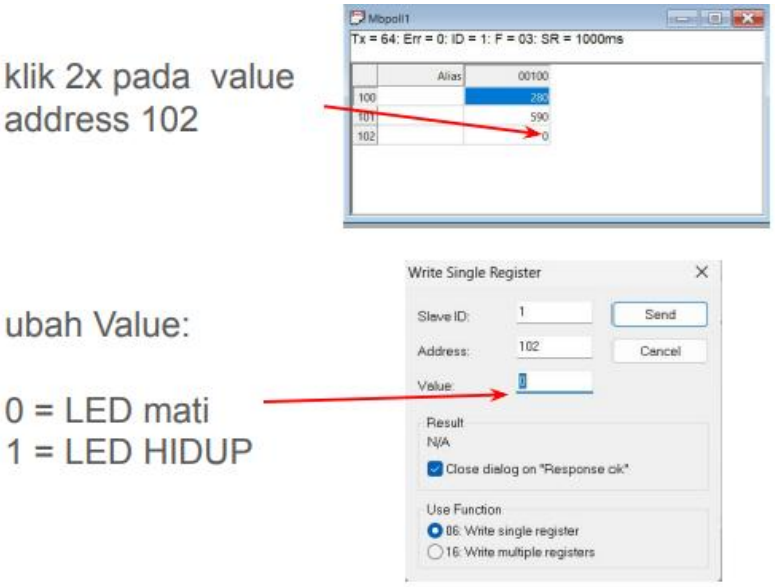
Gambar 2. 30 Kode Program Modbus TCP Over WiFi Dengan ESP32



Gambar 2. 31 Hasil Pada Modbus Poll Test

Data Pembacaan Pada hasil pengujian pada Modbus Poll Test :

- 100 untuk suhu
- 101 untuk kelembapan
- 102 untuk LED biru pada esp32



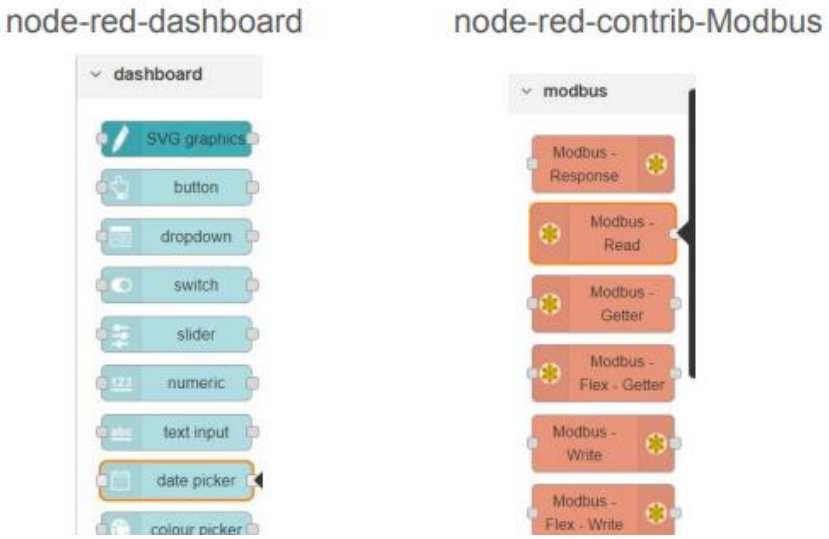
Gambar 2. 32 Test Menyalakan LED

o. Platform IoT Node-RED

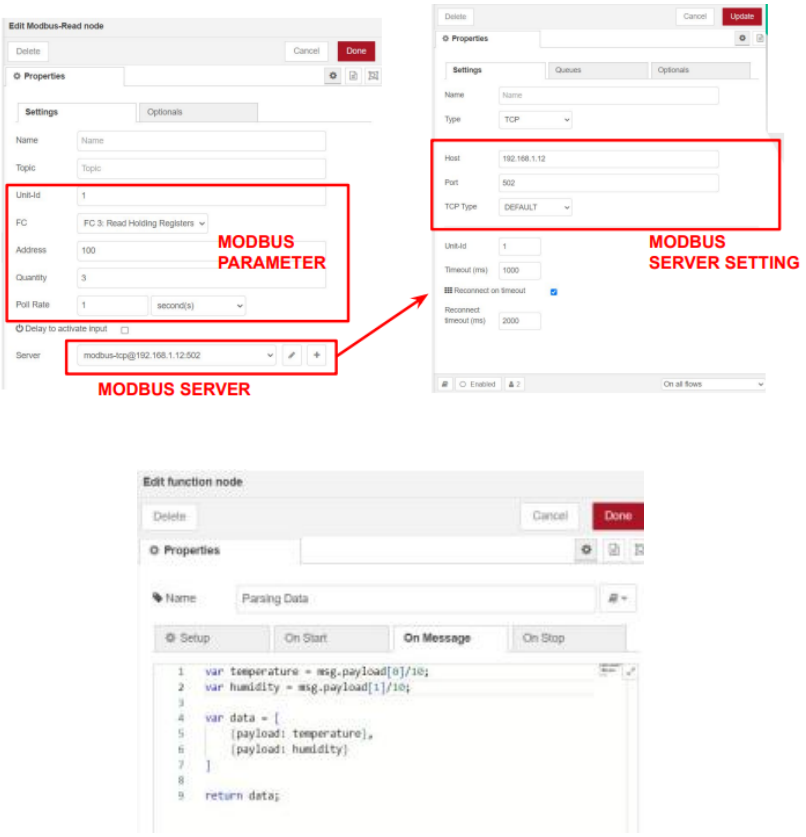
Node-RED merupakan tool berbasis browser untuk membuat aplikasi IoT yang mana lingkungan pemrograman visualnya berupa flow. Flow ini terbentuk dari beberapa node yang saling berhubungan di mana tiap node melakukan tugas tertentu. Kenapa Menggunakan Node Red :

1. Open Source (100% Gratis)
2. Pemrograman relatif lebih mudah karena Berbasis Visual
3. Bisa di install di Local (pada PC/Laptop) atau di Cloud (Amazon Web Service, Google Cloud, Microsoft Azure, dll)
4. Banyak dukungan oleh komunitas online

p. Membuat Dashboard pada Node-RED

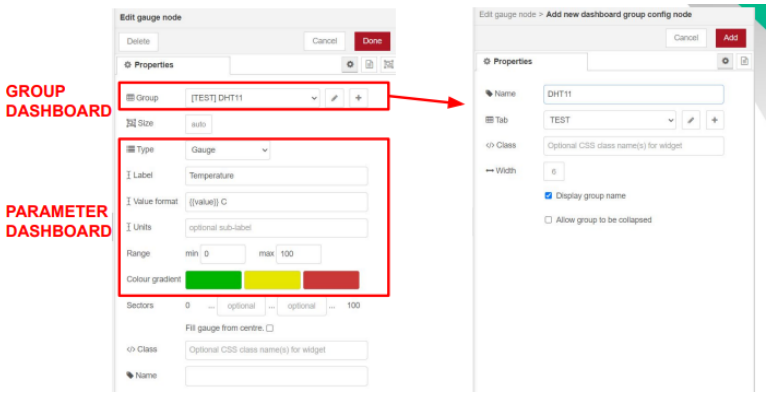


Gambar 2.33 Node palette yang perlu diinstal

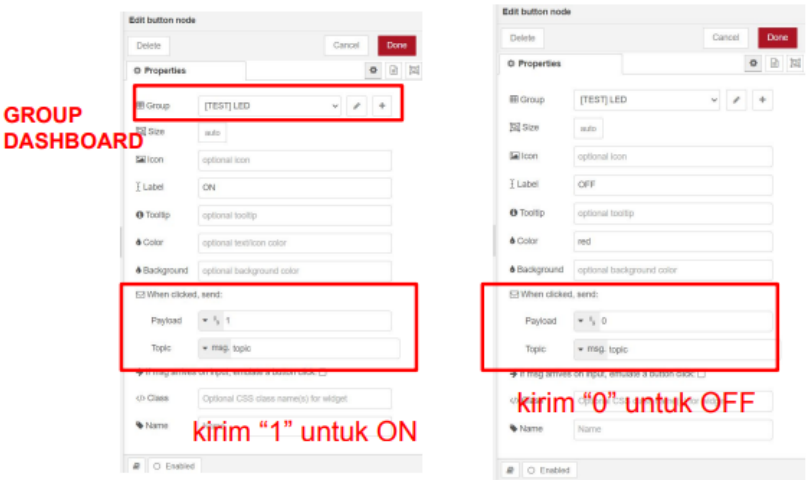


Gambar 2. 34 Setup Modbus

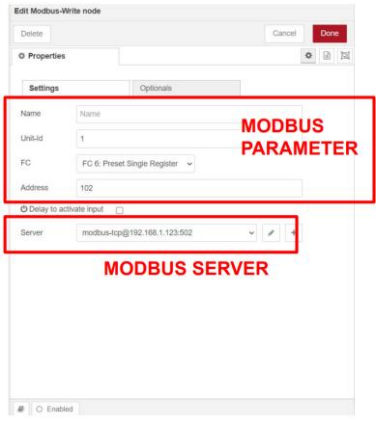
Code ini untuk parsing/memilah data Array keluaran modbus read, kemudian data yang sudah dipilah tersebut akan diteruskan ke masing-masing node Dashboard untuk divisualisasikan.



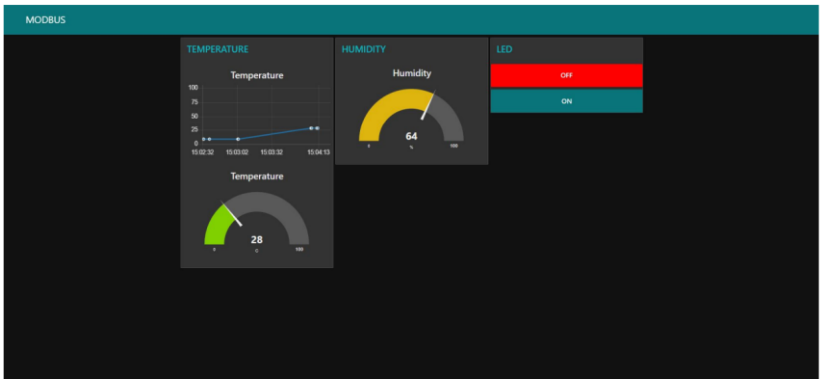
Gambar 2.35 Setup Modbus Dashboard



Gambar 2.36 Setup Button LED Dashboard



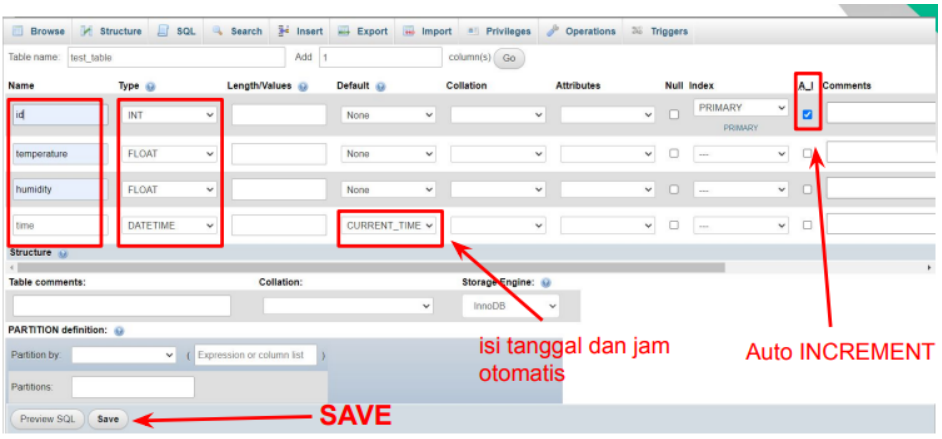
Gambar 2. 37 Setup Modbus Write



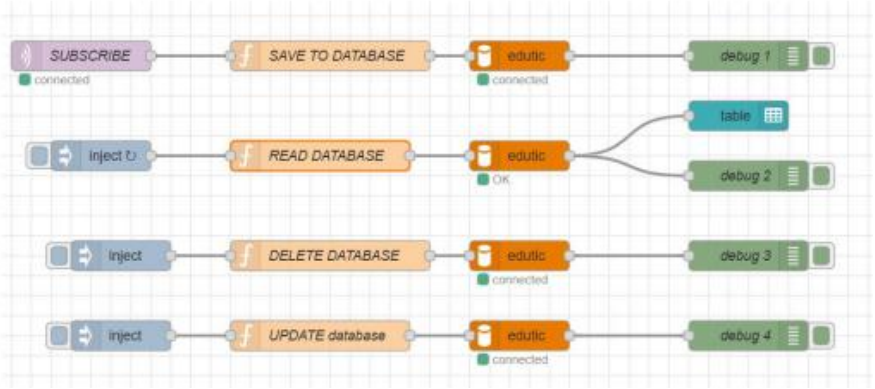
Gambar 2.38 Membuka Dashboard Alamat http://127.0.0.1:1880/ui di Browser

q. Membuat Database Untuk Penyimpanan Data IoT

MySQL merupakan sebuah DBMS (DataBase Management System) yang bertugas untuk mengatur/manage sebuah database. Secara garis besar DBMS memiliki 4 fungsi, yaitu : membuat, mencari, memperbarui, dan menghapus data. atau dalam istilah lain biasa disebut CRUD (Create, Read, Update, Delete). untuk menjalankan MySQL, kita memerlukan sebuah Server, yang mana dalam skenario ini kita akan menggunakan XAMPP sebagai server dari MySQL.

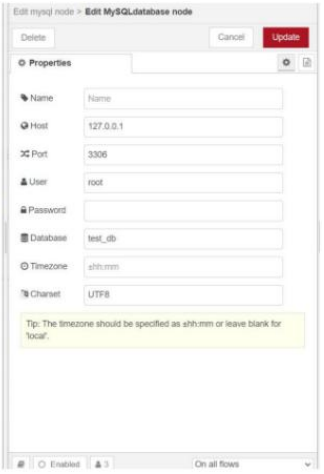


Gambar 2.39 Membuat Kolom Data di MySQL

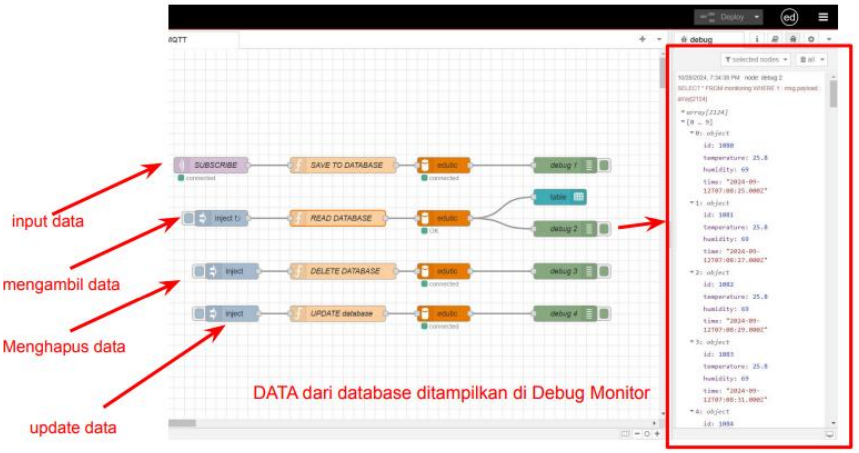


Gambar 2.40 Program Node Red

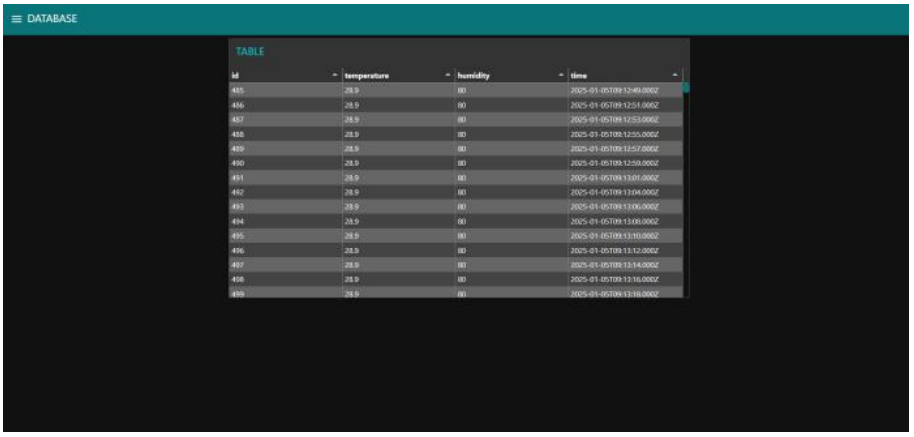
- Host 127.0.0.1
- Port 3306
- User root
- password (kosong)
- database (isi nama database)



Gambar 2.41 Setup Node MySQL



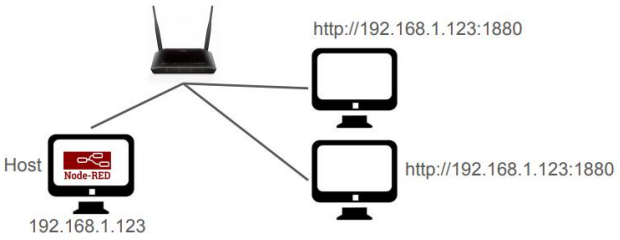
Gambar 2.42 Akses Data dari Node Red



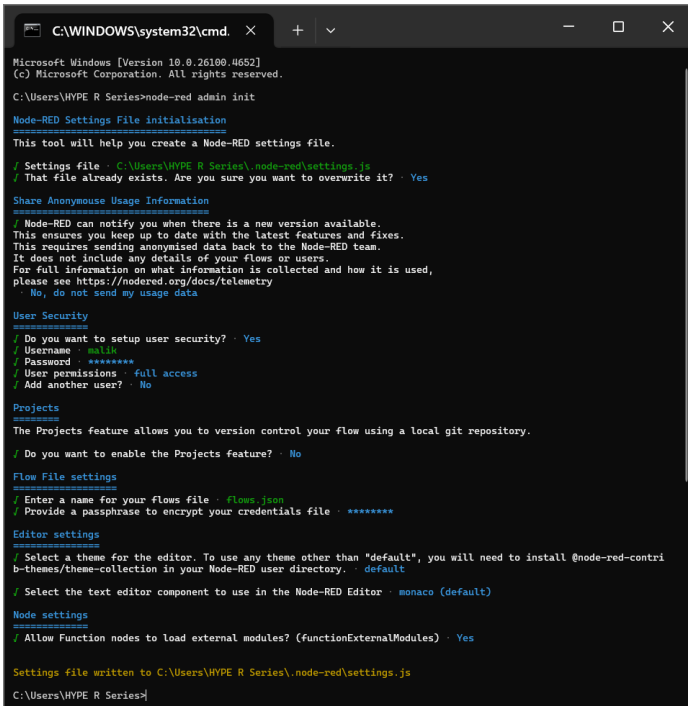
Gambar 2.43 Menampilkan Data dari Database ke Dashboard

r. Keamanan pada Node RED

Penerapan sistem keamanan menjadi hal yang sangat penting pada sebuah sistem IoT. Hal paling mendasar dari sistem keamanan adalah memberi sistem autentikasi pengguna. Singkatnya, pengguna harus memasukan Username dan password untuk mengakses node-red. Dalam satu jaringan lokal, kita dapat mengakses node-red milik orang lain hanya dengan memasukkan alamat IP komputer disestai port Node-red (192.168.1.123:1880)



Gambar 2. 44 Topologi Node Red pada Jaringan Lokal



Gambar 2.45 Membuat Sistem Autentikasi Pengguna

Gambar 2.46 Tampilan Autentikasi Pengguan Dengan Membuka http://localhost:1880 Atau http://127.0.0.1:1880 Pada Browser

2.2. Silabus

Silabus pelatihan merupakan panduan pembelajaran yang dirancang untuk membekali peserta dengan pengetahuan dan keterampilan praktis. Silabus mencakup pokok bahasan skema kompetensi:

Tabel 2.3 Informasi Program

Skema	Perekayasaan Perangkat Internet of Things
Silabus	1. Mengintegrasikan Perangkat Keras dan Perangkat Lunak (Firmware) untuk Device IoT 2. Menguji Coba Device IoT 3. Membuat Program Visual Antarmuka pada Perangkat Berbasis Web yang Terintegrasi dengan Mikrokontroler 4. Menguji Coba Aplikasi IoT 5. Melakukan Instalasi Perangkat IoT (Device) Sesuai Desain 6. Melakukan Instalasi Firmware pada Perangkat (Device) Secara Over The Air (OTA) 7. Menerapkan Perimeter Keamanan pada Perangkat IoT 8. Mengaplikasikan Patch Keamanan pada Perangkat IoT
Prasyarat	1. Pdf Ijazah Terakhir (Minimal D3) 2. Foto Formal Background Merah (ukuran 3x4). 3. Surat Pengalaman Kerja dan Curriculum Vitae (CV). 5. Portofolio Project/Dokumentasi Pekerjaan/Produk IoT 6. E-Sertifikat Pelatihan Online E-Learning Academy.edutic.id 7. Sertifikat Pelatihan Online berbasis Kompetensi 8. KTP / Indentitas Diri
Output Produk	1. Firmware Over-the-Air (FOTA/OTA) Programming dan Smart Monitoring 2. Smart Monitoring Temperature, Humidity, Lumen, dan Control Relay dengan Platform IoT UBIDOTS
Media dan Bahan Ajar	1. *Learning Management System* (LMS) 2. Materi format PDF dan Video *Recording*

Tabel 2.4 Silabus

Unit Kompetensi	Capaian Unit Kompetensi	Kriteria Capaian	Pokok Bahasan
1. Mengintegrasikan Perangkat Keras dan Perangkat Lunak (*Firmware*) untuk Device IoT	Mampu mengintegrasikan Perangkat Keras dan Perangkat Lunak (Firmware) untuk Device IoT	1. Mengakses Datasheet, dokumentasi, dan pustaka - pustaka software dari komponen-komponen aktif diidentifikasi, agar dapat digunakan untuk keperluan komunikasi dengan Mikrokontroler / Microprosesor pada perangkat keras IoT. 2. Komponen-komponen sensor diakses oleh Mikrokontroler/ microprocessor dengan menulis firmware berdasarkan pustaka-pustaka, dan diuji hasil pembacaannya. 3. Komponen-komponen yang berpotensi rusak atau tidak dapat berkomunikasi dengan Mikrokontroler/ microprocessor serta gejala-gejala yang diamati harus	1. K3 Elektronik 2. Pengenalan Komponen Elektronik Aktif dan Pasif 3. Pengenalan alat pengembangan Firmware 4. Pengenalan Protokol komunikasi 5. Pengenalan Push Button dan LED 6. Pengenalan Potensio dan LDR 7. Pengenalan PWM dengan LED 8. Pengenalan Ultrasonic Sensor HC-SR04 9. Pengenalan Sensor Suhu dan Kelembaban DHT11 10. Pengenalan LCD 16x2 i2C

		ditulis ke dalam sebuah dokumen. 4. Komponen-komponen perangkat keras dipastikan sudah terpasang dengan baik pada Printed Circuit Board (PCB) akhir dari perangkat IoT, dilanjutkan dengan Pengembangan firmware sesuai kebutuhan perangkat IoT.	
2. Menguji Coba Device IoT.	Mampu Menguji Coba Device IoT Sesuai Prosedur	1. Spesifikasi sistem IoT diidentifikasi sesuai dengan kebutuhan pengguna. 2. Macam komponen dan flow kerja aplikasi beserta fitur dan karakteristiknya diidentifikasi.	Uji Coba Platform Ubidots
3. Membuat Program Visual Antarmuka pada Perangkat Berbasis Web yang Terintegrasi dengan Mikrokontroler	Mampu Menggelar Aplikasi IoT Berbasis Web Sesuai Desain dan Standard	1. Spesifikasi perangkat IoT berbasis web diidentifikasi. 2. Fitur aplikasi IoT berbasis web diidentifikasi. 3. Integrasi sistem & compatibility test dilakukan terhadap aplikasi IoT berbasis web. 4. User test	1. Monitoring Suhu dan kelembaban berbasis web 2. Halaman Antarmuka berbasis web untuk Firmware Over The Air (FOTA)

		terhadap aplikasi IoT berbasis web dilakukan sesuai dengan kebutuhan. 5. Evaluasi terhadap aplikasi IoT berbasis web dilakukan sesuai standar yang ditetapkan.	
4. Menguji Coba Aplikasi IoT	Mampu Menguji Coba Aplikasi IoT Sesuai Prosedur	1. Spesifikasi sistem IoT diidentifikasi sesuai dengan kebutuhan pengguna. 2. Macam komponen dan flow kerja aplikasi beserta fitur dan karakteristiknya diidentifikasi. 3. Kesesuaian UI diuji berdasarkan desain. 4. UX test dilakukan sesuai kebutuhan pengguna.	Uji Coba Platform Ubidots
5. Melakukan Instalisasi Perangkat IoT (Device) sesuai Desain	Kesesuaian fungsionalitas perangkat IoT yang terinstalasi dengan dokumen hasil uji	Rancangan sistem manajemen serta SOP perangkat lunak (firmware) IoT yang mendukung instalasi secara OTA dituangkan ke dalam dokumen tertulis.	Menerapkan Firmware over the Air (FOTA) untuk ESP32.
6. Melakukan Instalasi Firmware pada Perangkat	Memahami Firmware Over The Air (FOTA) serta	1. Fungsionalitas sistem manajemen firmware secara	Membuat sistem monitoring suhu dan

(Device) Secara Over The Air (OTA)	mampu mendemonstrasikan Instalasi Firmware pada Perangkat (Device) Secara Over The Air (OTA)	OTA dibangun sesuai kebutuhan. 2. Infrastruktur software serta jaringan IoT untuk mendukung instalasi firmware secara OTA ditanam pada perangkat IoT.	kelembaban berbasis web local dengan fitur FOTA
7. Menerapkan Perimeter Keamanan pada Perangkat IoT	Memahami dan mampu mengimplementasikan Perimeter Keamanan pada Perangkat IoT	1. Potensi gangguan dan ancaman terhadap keamanan sistem perangkat IoT diidentifikasi sesuai dengan model risiko yang ada. 2. Strategi pemulihan sistem perangkat IoT setelah kejadian gangguan dan ancaman ditentukan secara detail. 3. Laporan hasil analisis data gangguan dan ancaman keamanan sistem perangkat IoT disusun secara lengkap sesuai dengan aspek teknis dan non teknis. 4. Rencana	1. Implementasi sistem reconnect jika koneksi Internet terputus 2. Implementasi token untuk izin pengiriman data ke Aplikasi IoT

		pencegahan dan pemulihan sistem perangkat IoT setelah kejadian gangguan dan ancaman disusun secara lengkap sesuai dengan SLA	
8. Mengaplikasikan Patch Keamanan pada Perangkat IoT	Memahami Patch keamanan dan mampu mengaplikasikan Patch Keamanan pada Perangkat IoT	1. Sistem keamanan IoT yang terdiri atas keamanan perangkat, jaringan, platform dan aplikasi diidentifikasi. 2. Penempatan patch serta alur kerja pada sistem keamanan pada perangkat IoT diidentifikasi. 3. Pengujian dan pengaplikasian patch keamanan perangkat IoT dengan memperhatikan keberlangsungan bisnis dilakukan sesuai prosedur. 4. Parameter kesuksesan pengaplikasian patch keamanan perangkat IoT diukur sesuai dengan standar yang ditetapkan	

2.3. Platform yang digunakan dalam proses pelatihan

Proses pelatihan berbasis kompetensi dilaksankan secara 2 (dua) sesi terdiri dari Sinkron dan Asinkron dengan pelaksanaan sebagai berkut :

a. Asinkron : peserta dapat melaksanakan pembelajaran secara mandiri dengan mengakses *Learning Management System* (LMS) yang telah disediakan. LMS dapat dipelajari setelah menyelesaikan administrasi pembayaran

b. Sinkronus : Pelaksanaan lima hari secara daring, dimulai dari hari Senin, 19 Mei 2025 s.d. Jum'at, 23 Mei 2025, pada pukul 19.00 s.d. 22.00 WIB, pada sesi kelas ini didampingi oleh instruktur dari Academy.Edutic.Id

Proses pelatihan menggunakan beberapa platform digital untuk memastikan pembelajaran berjalan efektif dan sesuai kebutuhan peserta, terdiri dari :

1. https://academy.edutic.id/

   Platform Acedemy Edutic merupakan sistem E-learning, digunakan untuk melihat materi pembelajaran dan mengikuti kuis setiap kali pertemuan, peserta wajib menyelesaikan E-learning.

2. Zoom Meeting

   Zoom digunakan untuk sesi kelas secara live bersama instruktur dan untuk sesi mentoring, dimana peserta bisa berkomunikasi dengan instruktur terkait topik yaan dipelajari, menyelesaikan permasalahan teknis, atau meminta masukan terkait proyek tugas akhir.

2. WhatsApp

   Selain sesi mentoring, ada juga kesempatan tanya-jawab melalui platform grup WhatsApp, yang digunakan untk berkomunkasi dengan instruktur di luar jam kelas. Grup WhatsApp juga digunakan oleh admin untuk menyampaikan informasi penting kepada peserta

3. Google Drive

   Pada akhir pelatihan, peserta akan diberikan tugas akhir berupa proyek, proyek ini adalah kesempatan bagi peserta untuk mengaplikasikan keterampilan yang telah dipelajari dalam kurikulum. Proyek ini nantinya akan disimpan di google drive yang telah disediakan oleh panitia penyelenggara untuk diakses sebagai portofolio digital peserta yang akan dipertanggung jawabkan saat uji kompetensi.

2.4. Proses Pelaksanaan Uji Kompetensi

Proses pelaksanaan uji sertifikasi Perekayasaan Perangkat IoT diawali dengan pendaftaran melalui website resmi Lembaga Sertifikasi Profesi – Telekomunikasi Digital Indonesia. Calon peserta mendaftar sebagai bagian dari mitra Academy.Edutic.id, setelah pendaftaran, peserta diminta untuk menandatangani surat pernyataan tidak merekam pelaksanaan sertifikasi. lalu peserta menerima tugas praktik berupa studi kasus menyiapkan :

1. Firmware Over-the-Air (FOTA/OTA) Programming dan Smart Monitoring,

2. Smart Monitoring Temperature, Humidity, Lumen, dan Control Relay dengan Platform IoT UBIDOTS.

Peserta memiliki waktu tiga hari untuk menyelesaikan tugas praktik sebelum interview dengan asesor. Ketentuan teknis yang harus dipenuhi meliputi skema sertifikasi yaitu :

Tabel 2.5 Skema Sertfikasi Perekayasaan Perangkat *Internet of Things*

No	Kode Unit	Judul Unit Kompetensi
1	J.61IOT01.004.1	Mengintegrasikan Perangkat Keras dan Perangkat Lunak (Firmware) untuk Device IoT
2	J.61IOT01.005.1	Menguji Coba Device IoT
3	J.61IOT01.008.1	Membuat Program Visual Antarmuka pada Perangkat Berbasis Web yang Terintegrasi dengan Mikrokontroler
4	J.61IOT01.010.1	Menguji Coba Aplikasi IoT
5	J.61IOT01.018.1	Melakukan Instalasi Perangkat IoT (Device) Sesuai Desain
6	J.61IOT01.019.1	Melakukan Instalasi Firmware pada Perangkat (Device) Secara Over The Air (OTA)
7	J.61IOT01.026.1	Menerapkan Perimeter Keamanan pada Perangkat IoT
8	J.61IOT01.041.1	Mengaplikasikan Patch Keamanan pada Perangkat IoT

Detail Pelaksanaan Uji Kompetensi secara luring pada tanggal 25 Mei 2025, dengan jumlah asesor satu orang, dan jumlah asesi dua orang. Selama sesi uji kompetensi, asesor menilai hasil dengan metode VATM (Valid, Asli, Terkini, Memadai) yang terdiri dari :

a. Valid: Bukti yang dikumpulkan harus relevan dengan unit kompetensi yang dinilai.

b. Asli: Bukti harus berasal dari peserta uji sendiri.

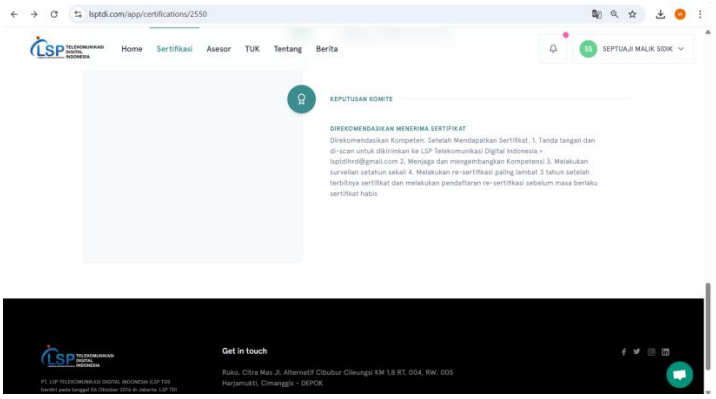
c. Terkini: Bukti harus menunjukkan bahwa kompetensi peserta uji masih relevan dan terkini.

d. Memadai: Bukti harus cukup untuk menunjukkan bahwa peserta uji memiliki kompetensi yang diperlukan.

2.5. Sistem Verifikasi Sertifikasi LSP Telekomunikasi Digital

Lembaga Sertifikasi Profesi – Telekomunikasi Digital Indonesia menyediakan sistem verifikasi melalui website resmi, Pemilik Sertifikat atau pihak yang ingin memverifikasi dapat ditunjukan oleh pemilik sertfikat untuk mengunjungi Lembaga Sertifikasi Profesi – Telekomunikasi Digital Indonesia, untuk cek keaslian sertifikat yang dimiliki dapat dilihat pada link berikut :

https://lsptdi.com/app/certifications/2550 (catatan : dibutuhkan log in)



Gambar 2.47 Bukti Keaslian Sertifikat BNSP