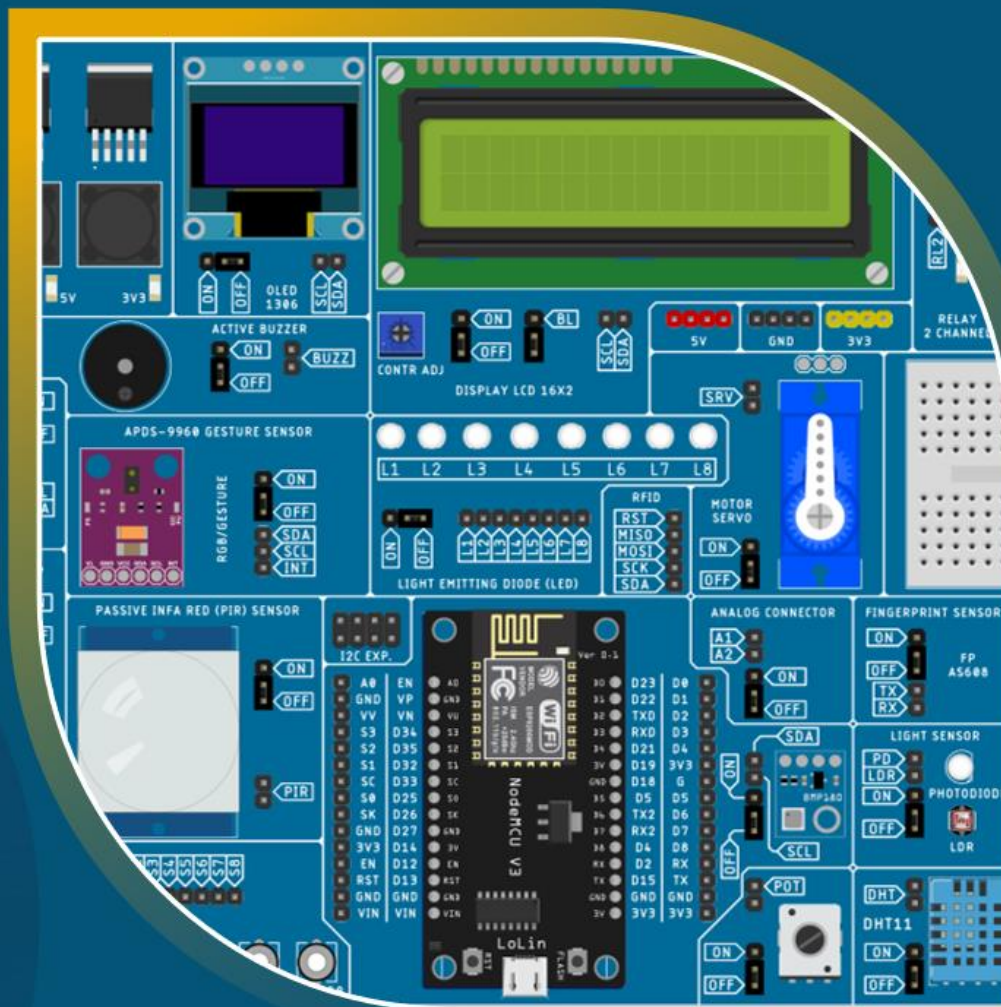


# MODUL TRAINER KIT INTERNET OF THINGS





Scan QR Code untuk mengunduh file pendukung  
Trainer Kit Internet of Things (IoT)



<https://electins.id/iot-kit/>

DAFTAR ISI.....	iii
PANDUAN UMUM .....	1
TRAINER KIT INTERNET OF THINGS.....	2
<b>MODUL 1   INTERNET OF THINGS .....</b>	<b>3</b>
A. Pengertian Internet of Things (IoT).....	3
B. Unsur Pembentuk IoT.....	3
C. Cara Kerja IoT .....	3
D. Penerapan IoT .....	4
E. Kesimpulan .....	4
<b>MODUL 2   ESP8266 NODEMCU .....</b>	<b>6</b>
A. ESP8266 NodeMCU.....	6
B. Deskripsi Pin NodeMCU Lolin V3.....	6
C. Spesifikasi dan Fitur NodeMCU .....	6
D. Referensi Pin NodeMCU.....	7
<b>MODUL 3   PEMROGRAMAN ARDUINO IDE.....</b>	<b>8</b>
A. Struktur Program .....	8
B. Variabel .....	10
C. Variable Scope .....	11
D. Tipe Data.....	11
E. Operator Aritmetic.....	12
F. Konstanta.....	13
G. Flow Control.....	14
H. Digital I/O.....	16
I. Analog I/O .....	16
J. Time .....	17
K. Math.....	17
L. Random .....	18
M. Serial.....	18
<b>MODUL 4   INSTALASI DAN KONFIGURASI ARDUINO IDE.....</b>	<b>19</b>
A. Instalasi Software Arduino IDE pada Sistem Operasi Windows... ..	19
B. Instalasi Board ESP8266 .....	22
C. Menambahkan Library .....	24
D. Pengenalan Software Arduino IDE .....	24
E. Menginstal Driver CH340.....	25
<b>MODUL 5   INSTALASI DAN KONFIGURASI APLIKASI IoT KIT .....</b>	<b>27</b>
A. Instalasi Aplikasi IoT Kit.....	27
B. Registrasi Akun dan Firebase Database .....	27
C. Registrasi Akun IoT KIT.....	33
D. Pengenalan Aplikasi IoT KIT .....	37
<b>MODUL 6   ANTARMUKA DENGAN LED, BUZZER DAN RELAY.....</b>	<b>41</b>
A. Pengenalan LED .....	41
B. Pengenalan Buzzer .....	41

C. Pengenalan Relay .....	41
D. Perangkat yang Dibutuhkan.....	43
E. Wiring Diagram dan Penyesuaian Aplikasi.....	43
F. Petunjuk Praktikum .....	43
<b>MODUL 7   ANTARMUKA DENGAN MOTOR SERVO .....</b>	<b>47</b>
A. Pengenalan Motor Servo .....	47
B. Perangkat yang Dibutuhkan.....	47
C. Wiring Diagram dan Penyesuaian Aplikasi.....	48
D. Petunjuk Praktikum .....	48
<b>MODUL 8   ANTARMUKA DENGAN LCD 16x2 I2C.....</b>	<b>52</b>
A. Pengenalan LCD 16x2 I2C.....	52
B. Perangkat yang Dibutuhkan.....	53
C. Wiring Diagram dan Penyesuaian Aplikasi.....	53
D. Petunjuk Praktikum .....	54
<b>MODUL 9   ANTARMUKA DENGAN OLED DISPLAY.....</b>	<b>57</b>
A. Pengenalan OLED Display.....	57
B. Perangkat yang Dibutuhkan.....	57
C. Wiring Diagram dan Penyesuaian Aplikasi.....	58
D. Petunjuk Praktikum .....	58
<b>MODUL 10   ANTARMUKA DENGAN SENSOR CAHAYA ANALOG .....</b>	<b>62</b>
A. Pengenalan Sensor Cahaya Analog.....	62
B. Perangkat yang Dibutuhkan.....	63
C. Wiring Diagram dan Penyesuaian Aplikasi.....	63
D. Petunjuk Praktikum .....	64
<b>MODUL 11   ANTARMUKA DENGAN BH1750.....</b>	<b>68</b>
A. Pengenalan Sensor Cahaya BH1750 .....	68
B. Perangkat yang Dibutuhkan.....	68
C. Wiring Diagram dan Penyesuaian Aplikasi.....	68
D. Petunjuk Praktikum .....	69
<b>MODUL 12   ANTARMUKA DENGAN POTENSIOMETER.....</b>	<b>73</b>
A. Pengenalan Potensiometer.....	73
B. Perangkat yang Dibutuhkan.....	74
C. Wiring Diagram dan Penyesuaian Aplikasi.....	74
D. Petunjuk Praktikum .....	74
<b>MODUL 13   ANTARMUKA DENGAN SENSOR PIR.....</b>	<b>79</b>
A. Pengenalan Sensor PIR .....	79
B. Perangkat yang Dibutuhkan.....	80
C. Wiring Diagram dan Penyesuaian Aplikasi.....	80
D. Petunjuk Praktikum .....	80
<b>MODUL 14   ANTARMUKA DENGAN SENSOR ULTRASONIK .....</b>	<b>84</b>
A. Pengenalan Sensor Ultrasonik .....	84

B. Perangkat yang Dibutuhkan.....	85
C. Wiring Diagram dan Penyesuaian Aplikasi.....	85
D. Petunjuk Praktikum .....	85
<b>MODUL 15   ANTARMUKA DENGAN DHT11 .....</b>	<b>90</b>
A. Pengenalan Sensor DHT11 .....	90
B. Perangkat yang Dibutuhkan.....	90
C. Wiring Diagram dan Penyesuaian Aplikasi.....	91
D. Petunjuk Praktikum .....	91
<b>MODUL 16   ANTARMUKA DENGAN BMP180 .....</b>	<b>96</b>
A. Pengenalan Sensor BMP180 .....	96
B. Wiring Diagram yang Dibutuhkan.....	96
C. Rangkaian dan Penyesuaian Aplikasi.....	97
D. Petunjuk Praktikum .....	97
<b>MODUL 17   ANTARMUKA DENGAN SOIL MOISTURE/RAIN SENSOR..</b>	<b>102</b>
A. Pengenalan Sensor Kelembaban Tanah dan Sensor Hujan.....	102
B. Perangkat yang Dibutuhkan.....	103
C. Wiring Diagram dan Penyesuaian Aplikasi.....	103
D. Petunjuk Praktikum .....	103
<b>MODUL 18   ANTARMUKA DENGAN GESTURE SENSOR.....</b>	<b>108</b>
A. Pengenalan Sensor Gesture APDS 9960 .....	108
B. Perangkat yang Dibutuhkan.....	108
C. Wiring Diagram.....	109
D. Petunjuk Praktikum .....	109
<b>MODUL 19   ANTARMUKA DENGAN RFID .....</b>	<b>115</b>
A. Pengenalan RFID .....	115
B. Perangkat yang Dibutuhkan.....	116
C. Wiring Diagram dan Penyesuaian Aplikasi.....	116
D. Petunjuk Praktikum .....	116
<b>MODUL 20   ANTARMUKA DENGAN SENSOR FINGERPRINT.....</b>	<b>123</b>
A. Pengenalan Sensor Fingerprint .....	123
B. Perangkat yang Dibutuhkan.....	123
C. Wiring Diagram.....	124
D. Petunjuk Praktikum .....	124

## PANDUAN UMUM

### PENTING!!!

Sebelum mengoperasikan Trainer Kit Internet of Things (IoT) mohon untuk memperhatikan **Poin** berikut :

1. Jangan gunakan Adaptor di atas 12 Volt (Rekomendasi 9-12V dengan Arus 1-2 Ampere). Menggunakan Adaptor dengan tegangan yang lebih tinggi dapat menyebabkan kerusakan pada trainer.
2. Jangan melakukan *short circuit* pada jalur tegangan 5V, 3.3V dan GND. **Short** tegangan dapat menyebabkan kerusakan pada trainer.
3. Jauhkan benda yang bersifat konduktif atau benda tajam yang dapat merusak trainer.
4. Jangan menahan putaran motor servo karena dapat menyebabkan tegangan *drop* yang menyebabkan kerusakan pada trainer.
5. Pastikan ketika mencabut *module* atau melakukan penggantian *part* adaptor dalam keadaan tidak tercolok dan pemasangan tidak terbalik.
6. Relay 1 (RL1) hanya untuk perangkat keluaran yang membutuhkan tegangan 12V. **Jangan dihubungkan ke tegangan AC!**
7. Relay 2 (RL2) bisa digunakan untuk sakelar tegangan DC maupun AC.

### Catatan:

\* Gunakan WiFi dan Pasword Anda

```
#define WIFI_SSID "SSID_WIFI"  
#define WIFI_PASSWORD "PASS_WIFI"
```

\*\*Gunakan URL Firebase dan Token Anda

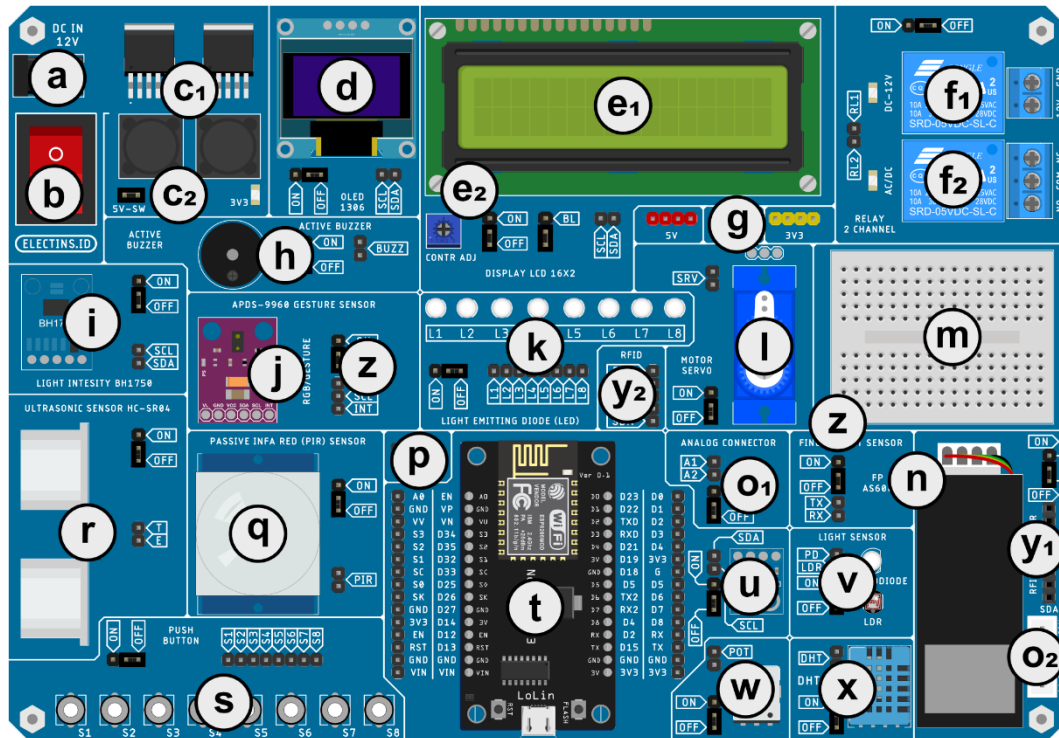
```
#define DATABASE_URL "project_id.firebaseio.com"  
#define API_KEY "database_secret"
```

\*\*\*Gunakan user id yang di daftarkan pada aplikasi IoT KIT

```
String user = "user_id";
```



## TRAINER INTERNET OF THINGHS (IoT)



### Bagian-Bagian Trainer

- |                                      |                                    |
|--------------------------------------|------------------------------------|
| a. Socket DC IN                      | n. Sensor Sidik jari               |
| b. Sakelar ON-OFF                    | o1. Header Analog Konektor         |
| c1. Regulator <i>Stepdown</i>        | o2. Soket Sensor Analog            |
| c2. 5V <i>Switch Cap Jumper</i>      | p. Perluasan Pin I2C               |
| d. OLED <i>Display</i>               | q. Sensor PIR                      |
| e1. LCD 16x2 I2C                     | r. Sensor Ultrasonik               |
| e2. Pengatur Kecerahan LCD           | s. Sakelar <i>Push Button</i>      |
| f1. Relay (Output 12V)               | t. NodeMCU                         |
| f2. Relay (Sakelar AC/DC)            | u. Sensor BMP180                   |
| g. Sumber tegangan 5V, GND, 3.3V     | v. Sensor Cahaya Analog            |
| h. Aktif Buzzer                      | w. Potensiometer                   |
| i. Sensor Cahaya BH1750              | x. Sensor DHT11                    |
| j. Sensor Gesture APDS9960           | y1. Konektor RFID                  |
| k. <i>Light Emitting Diode</i> (LED) | y2. <i>Header</i> RFID             |
| l. Motor Servo                       | z. <i>Header Cap Jumper</i> ON-OFF |
| m. <i>Mini Breadboard</i>            |                                    |





#### D. Penerapan IoT

Penerapan IoT dalam kehidupan sehari-hari tanpa kita sadari banyak kita temui bahkan sangat dekat dengan kita. Berikut beberapa penerapan IoT dalam berbagai bidang :

1. Bidang kesehatan

Salah satu contohnya yaitu dengan adanya alat kesehatan yang mampu mengukur detak jantung, kadar gula, pengecekan suhu tubuh dan lain sebagainya yang disimpan ke penyimpanan data berskala besar yang dapat dimuat dan digunakan kapan saja.

2. Bidang pertanian

Salah satu contohnya yaitu penerapan sistem IoT pada *smart agriculture* yang dapat mengumpulkan data suhu, kadar air tanah, curah hujan dan pemantauan hama.

3. Bidang transportasi

Salah satu contohnya adalah mobil dengan kemampuan *autopilot* yang dapat berjalan dan parkir sendiri. Penerapan IoT diharapkan dapat mengurangi angka kecelakaan

4. Bidang Industri

Beberapa contoh penerapan IoT pada sektor industri seperti pemantauan peralatan otomatis, pemeliharaan prediktif, kontrol kualitas produksi, keamanan produktivitas dan banyak lainnya.

5. Otomasi Rumah (*Smart Home*)

Beberapa contohnya seperti pencahayaan otomatis yang diatur melalui ponsel, keamanan rumah berbasis RFID dan *fingerprint*, penyiraman tanaman, pakan otomatis hewan peliharaan dan lain sebagainya.

6. Lingkungan

Penerapan IoT pada lingkungan seperti pemantauan kualitas udara, suhu udara setempat, pengecekan kondisi air, pengindraan jarak jauh dalam mitigasi bencana oleh LAPAN dan masih banyak lagi yang menggunakan teknologi *Internet of Things* (IoT).

#### E. Kesimpulan

*Internet of Things* (IoT) adalah sebuah konsep dimana objek mampu mengirimkan data menggunakan jaringan untuk melakukan aktivitas kerja tanpa bantuan dari manusia atau interaksi dengan perangkat komputer.

Contoh penerapan IoT dapat dilakukan di berbagai bidang seperti kesehatan, pertanian, transportasi, lingkungan, industri, rumah, lingkungan dan lain sebagainya. Manfaat utama dari *Internet of Things* adalah ketercapaian efisiensi, efektivitas dan konektivitas.

## MODUL 2 | ESP8266 NODEMCU

### A. ESP8266 NodeMCU

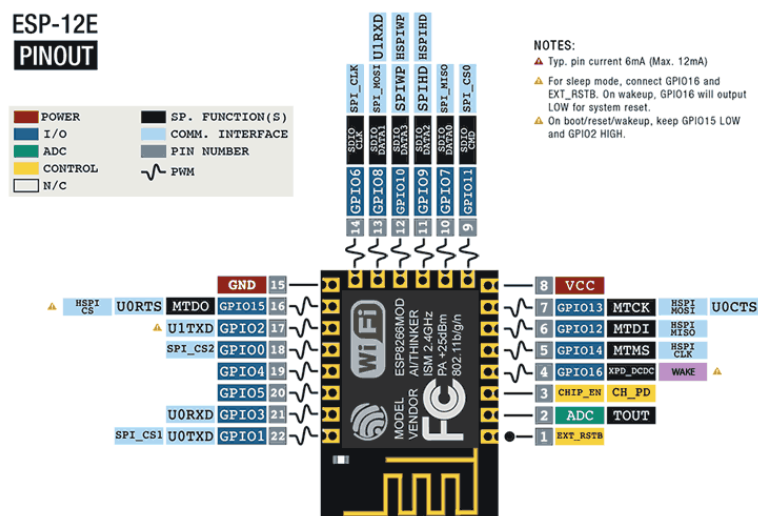
NodeMCU adalah sebuah platform IoT yang bersifat open source. Terdiri dari perangkat keras berupa *System On Chip* (SoC) ESP8266 buatan Espressif System. NodeMCU ESP8266 dilengkapi dengan modul ESP-12E yang berisi *chip* ESP8266 yang memiliki mikroprosesor Tensilica Xtensa LX106 RISC 32-bit. NodeMCU menggunakan *firmware* dengan bahasa pemrograman *scripting* Lua.



Gambar 2.1 NodeMCU Lolin V3

NodeMCU Lolin V3 menggunakan *chip* ESP8266 (ESP-12E) yang lebih stabil dari ESP-12. Beberapa fitur yang dimiliki antara lain :

- |                             |  |
|-----------------------------|--|
| 1. Memiliki 17 GPIO         | 7. <i>Wi-Fi Direct</i> (P2P), Soft-AP        |
| 2. Antarmuka I2C dan SPI    | 8. <i>Protocol stack</i> TCP/IP terintegrasi |
| 3. Antarmuka 1 Wire         | 9. CPU 32-bit berdaya rendah                 |
| 4. UART                     | 10. SDIO 2.0, SPI, UART                      |
| 5. ADC 10-bit               |  |
| 6. 11 b/g/n <i>protocol</i> |  |



Gambar 2.2 ESP8266 (ESP-12E) pinout

## B. Deskripsi Pin NodeMCU Lolin V3

Kategori Pin	Nama Pin	Detail
Power	Micro-USB, 3V3, GND, Vin	<p><b>Micro-USB:</b> NodeMCU dapat dinyalakan melalui <i>port</i> USB.</p> <p><b>3V3 :</b> Tegangan 3.3V ter-regulasi yang digunakan untuk menyuplai <i>board</i> NodeMCU.</p> <p><b>GND:</b> Pin <i>ground</i> (-).</p> <p><b>Vin:</b> Tegangan input ke NodeMCU untuk sumber daya eksternal.</p>
Pin Kontrol	EN, RST	Pin dan tombol untuk mengatur ulang mikrokontroler.
Pin Analog	A0	Digunakan untuk mengukur tegangan analog 0-3.3V.
Pin SPI	D5 (SCK), D6 (MISO), D7 (MOSI), D8 (SS)	Digunakan untuk komunikasi SPI NodeMCU.
Serial	RX, TX	Digunakan untuk mengirim dan menerima data serial TTL.
I2C	D1 (SCL), D2 (SDA)	Digunakan untuk komunikasi I2C.

## C. Spesifikasi dan Fitur NodeMCU

- Mikrokontroler: Tensilica 32-bit RISC CPU Xtensa LX106
- Tegangan Operasi: 3.3V
- Tegangan Input: 7-12V
- Digital I/O: 16
- Analog Input : 1 (10-bit)
- Antarmuka UART: 1
- Antarmuka SPI: 1
- Antarmuka I2C: 1
- Memori Flash: 4 MB
- SRAM: 64 KB
- *Clock Speed:* 80 MHz

Beberapa hal penting yang perlu diperhatikan pada ESP8266 adalah penggunaan GPIO yang tidak sesuai dengan label pada NodeMCU.

Misalnya GPIO 16 dengan label D0, GPIO 5 dengan label D1. Berikut referensi penggunaan pin NodeMCU untuk menghindari kesalahan penggunaan.

#### D. Referensi Pin NodeMCU

Label	GPIO	Input	Output	Keterangan
D0	GPIO 16	Tidak ada interupsi	Tidak ada PWM atau I2C	<b>Output HIGH saat boot</b> Digunakan untuk keluar dari <i>deep sleep</i> .
D1	GPIO 5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Pin I2C (SCL)
D2	GPIO 4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Pin I2C (SDA)
D3	GPIO 0	<i>Pull up</i>	<input checked="" type="checkbox"/>	Terhubung ke tombol <i>flash</i> , <i>boot</i> gagal jika pin <i>pull</i> LOW
D4	GPIO 2	<i>Pull up</i>	<input checked="" type="checkbox"/>	<b>Output HIGH saat boot</b> Terhubung ke LED <i>on-board</i> , <i>boot</i> gagal jika pin <i>pull</i> LOW
D5	GPIO 14	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Pin SPI (SCLK)
D6	GPIO 12	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Pin SPI (MISO)
D7	GPIO 13	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Pin SPI (MOSI)
D8	GPIO 15	<i>Pull GND</i>	<input checked="" type="checkbox"/>	Pin SPI (CS) <i>Boot</i> gagal jika pin <i>pull</i> HIGH
RX	GPIO 3	<input checked="" type="checkbox"/>	Pin RX	<b>Output HIGH saat boot</b>
TX	GPIO 1	Pin TX	<input checked="" type="checkbox"/>	<b>Output HIGH saat boot</b> Debug output saat <i>boot</i> , <i>boot</i> gagal jika pin <i>pull</i> LOW
A0	ADC0	Analog Input	-	-

## MODUL 3 | DASAR PEMROGRAMAN ARDUINO IDE

Bahasa pemrograman Arduino pada dasarnya menggunakan bahasa pemrograman C. Bahasa C sendiri merupakan bahasa tingkat tinggi yang sangat populer dan banyak digunakan oleh para *programmer*. Dengan demikian aturan penulisan dan penggunaan dari bahasa Arduino akan sama dengan bahasa C.

Modul ini akan membahas mengenai *Structure program, Variable, Variable cope, Data type, Operator arimetric, Constants, Flow controls, Digital I/O, Analog I/O, Time, Math, Random* dan *Serial*.

### A. Struktur Program

#### 1. Struktur

Struktur dasar bahasa pemrograman Arduino sangat sederhana hanya terdiri dari dua bagian. Dua bagian tersebut dapat juga disebut sebagai fungsi utama yaitu `setup()` dan `loop()`.

```
void setup()  
{  
  // Statement  
}  
void loop()  
{  
  // Statement  
}
```

Di mana `void setup()` adalah bagian untuk inisialisasi yang hanya dijalankan sekali di awal program, sedangkan `void loop()` untuk mengeksekusi bagian program yang akan dijalankan berulang-ulang untuk selamanya.

#### **setup()**

Fungsi `setup()` hanya di panggil satu kali ketika program pertama kali dijalankan. Ini digunakan untuk mendefinisikan mode pin atau memulai komunikasi serial. Fungsi `setup()` harus di ikut sertakan dalam program walaupun tidak ada *statement* yang dijalankan.

```
void setup()  
{  
  pinMode(D4, OUTPUT); // mengeset pin D4 sebagai output  
}
```

## loop()

Setelah menjalankan fungsi `setup()` maka secara langsung akan melakukan fungsi `loop()` secara berurutan dan melakukan instruksi-instruksi yang ada dalam fungsi `loop()` terus menerus.

```
void loop()
{
  digitalWrite(D4, HIGH); // nyalakan led built-in pada pin D4
  delay(1000);           // jeda selama 1 detik
  digitalWrite(D4, LOW); // matikan led built-in pada pin D4
  delay(1000);           // jeda selama 1 detik
}
```

## 2. Fungsi

*Function* (fungsi) adalah blok pemrograman yang mempunyai nama dan mempunyai *statement* yang akan di eksekusi ketika *function* tersebut di panggil. Fungsi `void setup()` dan `void loop()` telah di bahas di atas dan pembuatan fungsi yang lain akan di bahas selanjutnya.

```
type functionName(parameters)
{
  // Statement
}
```

Contoh:

```
int delayVal()
{
  int v; // membuat variabel dengan nama v bertipe integer
  v = analogRead(A0); // baca nilai analog A0
  v /= 4; // konversi 0-1023 ke 0-255
  return v; // return nilai v
}
```

Pada contoh di atas fungsi tersebut memiliki nilai balik `int` (integer), karena kalau tidak menghendaki adanya nilai balik maka type *function* harus *void*.

## 3. Curly braces { }

*Curly brace* mendefinisikan awal dan akhir dari sebuah blok fungsi. Apabila ketika memprogram dan *programmer* lupa memberi *curly brace* tutup maka ketika di *compile* akan terdapat laporan *error*.



```
type function()
{
  statements;
}
```

#### 4. Semicolon

Tanda titik koma adalah sintak wajib dalam pemrograman Arduino. Biasanya sintak ini ditempatkan pada akhir pernyataan.

```
int y = 30;
```

#### 5. Blok Komentar /\*...\*/

*Semicolon* harus di berikan pada setiap *statement* program yang kita buat ini merupakan pembatas setiap *statement* program yang di buat.

```
/*
  Ini adalah blok komentar
  Jangan lupa tutup komentar
  dengan tanda penutup
*/
```

#### 6. Baris Komentar //

Sama halnya dengan blok komentar, baris komentar pun sama hanya saja yang dijadikan komen adalah per baris.

```
// ini adalah baris komentar
```

### B. Variabel

Variabel adalah sebuah penyimpan nilai yang dapat digunakan dalam program. Variabel dapat di rubah sesuai dengan instruksi yang kita buat. Ketika mendeklarasikan variabel harus diikut sertakan tipe variabel serta nilai awal variabel.

```
type variableName = 0;
```

Contoh:

```
//mendefinisikan sebuah variabel bernama inputVariable dengan nilai awal 0
```

```
int inputVariable = 0;
```

```
//menyimpan nilai yang ada di pin analog A0 ke inputVariable
inputVariable = analogRead(A0);
```

### C. Variable Scope

Sebuah variabel dapat dideklarasikan pada awal program sebelum `void setup()`, secara lokal di dalam sebuah fungsi, dan terkadang di dalam sebuah blok `statement` pengulangan. Sebuah variabel global hanya satu dan dapat digunakan pada semua blok fungsi dan `statement` di dalam program. Variabel global di deklarasikan pada awal program sebelum fungsi `setup()`. Sebuah variabel lokal dideklarasikan di setiap blok fungsi atau di setiap blok `statement` pengulangan dan hanya dapat digunakan pada `block` yang bersangkutan saja. Contoh penggunaan:

```
int value; // 'value' adalah variabel global dan dapat digunakan pada semua blok fungsi
```

```
void setup()
{
  value = 0;
}

void loop()
{
  value = 12 + 30;
}
```

### D. Tipe Data

#### 1. byte

Tipe data *byte* dapat menyimpan 8-bit nilai angka bilangan asli tanpa koma. *Byte* memiliki *range* 0 – 255.

```
byte biteVariable = 160; // mendeklarasikan 'biteVariable' sebagai tipe data byte
```

#### 2. integer

*Integer* merupakan tipe data utama untuk menyimpan nilai bilangan bulat tanpa koma. Penyimpanan *integer* sebesar 16-bit dengan *range* 32.767 sampai -32.768.

```
int integerValue = 1500; // mendeklarasikan 'integerVariable' sebagai tipe data integer
```

#### 3. long

Perluasan ukuran untuk *long integer*, penyimpanan *long integer* sebesar 32-bit dengan *range* 2.147.483.647 sampai -2.147.483.648.

```
long longVariable = 900000; // mendeklarasikan 'longVariable'  
sebagai tipe data long
```

#### 4. **float**

*Float* adalah tipe data yang dapat menampung nilai *decimal*, *float* merupakan penyimpan yang lebih besar dari integer dan dapat menyimpan sebesar 32-bit dengan *range* 3.4028235E+38 sampai -3.4028235E+38.

```
float floatVariable = 3.14; // mendeklarasikan 'floatVariable' sebagai  
tipe data float
```

#### 5. **array**

*Array* adalah kumpulan nilai yang dapat di akses dengan nomor indeks, nilai yang terdapat dalam *array* dapat di panggil dengan cara menuliskan nama *array* dan nomor indeks. *Array* dengan indeks 0 merupakan nilai pertama dari *array*. *Array* perlu di deklarasikan dan kalau perlu diberi nilai sebelum digunakan.

```
int myArray[] = {value0, value1, value2 ... }
```

Contoh penggunaan *array*:

```
int myArray[] = {2,4,6,8,10}  
int x = myArray[4]; // x sekarang sama dengan 10
```

### E. **Operator Aritmetic**

#### 1. **Aritmetic**

operator aritmatik terdiri dari penjumlahan, pengurangan, perkalian, dan pembagian.

```
y = y + 3;  
x = x - 8;  
i = i * 5;  
r = r / 9;
```

Dalam menggunakan operan aritmatik harus hati-hati dalam menentukan tipe data yang digunakan jangan sampai terjadi kelebihan *range* data.

#### 2. **Compound Assignments**

*Compound assignments* merupakan kombinasi dari aritmatik dengan sebuah variabel. Ini biasanya dipakai pada pengulangan.

```
x ++; // sama seperti x = x + 1 atau menambah nilai x sebesar 1
x --; // sama seperti x = x - 1 atau mengurangi nilai x sebesar 1
x += y; // sama seperti x = x + y
x -= y; // sama seperti x = x - y
x *= y; // sama seperti x = x * y
x /= y; // sama seperti x = x / y
```

### 3. Comparison

*Statement* ini membandingkan dua variabel dan apabila terpenuhi akan bernilai 1 atau *true*. *Statement* ini banyak digunakan dalam operator bersyarat.

```
x == y; // x sama dengan y
x != y; // x tidak sama dengan y
x < y; // x lebih kecil dari y
x > y; // x lebih besar dari y
x <= y; // x lebih kecil dari sama dengan y
x >= y; // x lebih besar dari sama dengan y
```

### 4. Logic Operator

Operator logika digunakan untuk membandingkan dua ekspresi dan mengembalikan nilai balik benar atau salah tergantung dari operator yang digunakan. Terdapat 3 operator logika AND, OR, dan NOT, yang biasanya digunakan pada *if statement*. Contoh penggunaan:

#### a. Logical AND

```
if(x>0 && x<5) // bernilai benar apabila kedua dari operator
pembanding terpenuhi
```

#### b. Logical OR

```
if(x>0 || y>0) // bernilai benar apabila salah satu dari operator
pembanding terpenuhi
```

#### c. Logical NOT

```
if(!x > 0) // bernilai benar apabila ekspresi operator salah
```

## F. Konstanta

Arduino mempunyai beberapa variabel yang sudah dikenal yang kita sebut konstanta. Ini membuat memprogram lebih mudah untuk dibaca. Konstanta di klasifikasi berdasarkan grup.

### 1. true/false

Merupakan konstanta *boolean* yang mendefinisikan *logic level*. *False* dapat didefinisikan sebagai 0 dan *True* sebagai 1.

```
if(b == true);  
{  
  // Laksanakan rencana  
}
```

## 2. **high/low**

Konstanta ini digunakan untuk menentukan kondisi pin pada level HIGH atau LOW ketika membaca dan menulis dari/ke pin digital. HIGH didefinisikan sebagai *logic 1*, ON atau 3.3V volt sedangkan LOW sebagai *logic 0*, OFF atau 0 volt.

```
digitalWrite(D4, HIGH);
```

## 3. **input/output**

Konstanta ini digunakan dengan fungsi `pinMode()` untuk mendefinisikan mode pin digital, sebagai INPUT atau OUTPUT.

```
pinMode(D4, OUTPUT);
```

## G. **Flow Control**

### 1. **if**

Operator *if* mengetes sebuah kondisi apakah sudah tercapai/benar atau belum, dicontohkan seperti pengetesan nilai analog apakah sudah berada di bawah nilai yang kita kehendaki atau belum, apabila terpenuhi maka akan mengeksekusi baris program yang ada dalam *brackets {...}* kalau tidak terpenuhi maka akan melewati baris program yang ada dalam *brackets*.

```
if(someVariable ?? value)  
{  
  // Laksanakan rencana  
}
```

### 2. **if...else**

Operator *if...else* mengetes sebuah kondisi apabila tidak sesuai dengan kondisi maka akan mengeksekusi baris program yang ada di *else*.

```
if(inputPin == HIGH)  
{  
  // Laksanakan rencana A  
}  
else  
{
```

```
// Laksanakan rencana B  
}
```

### 3. **for**

Operator *for* digunakan untuk mengulang blok *statement* di dalam *bracket*, beberapa kali sesuai dengan jumlah yang ditentukan. Setiap perulangan *for()* mempunyai tiga parameter dan dipisahkan menggunakan titik koma (;):

```
for(initialization; condition; expression)  
{  
  // Laksanakan rencana perulangan  
}
```

Contoh penggunaan:

```
for(index=0; index<=3; index++)  
{  
  counter++;  
}
```

- **index=0; Sesuatu yang dikerjakan sebelum dimulai:** Membuat `index = 0`.
- **index<=3; Operasi logika yang dites, selama hasilnya benar (true) akan terus looping:** Jika `index` lebih kecil atau sama dengan 3, akan menjalankan kode yang ada di dalam *bracket* {}. Ketika `index = 4`, akan keluar dari *loop* dan melanjutkan baris kode selanjutnya di dalam *sketch*.
- **index++ Sesuatu yang dilakukan setelah menjalankan satu baris statement:** Meletakkan "++" setelah sebuah variabel bermaksud menambahkan variabel tersebut dengan satu. Dapat juga menggunakan "`index = index + 1`".

### 4. **while**

Operator *while* akan terus mengulang baris perintah yang ada dalam *bracket* sampai ekspresi sebagai kondisi pengulangan bernilai salah.

```
while(someVariable ?? value)  
{  
  // Laksanakan rencana  
}
```

## 5. **do... while**

Sama halnya dengan `while()` hanya saja pada operator `do...while` tidak melakukan pengecekan pada awal tapi di akhir, sehingga otomatis akan melakukan satu kali baris perintah walaupun pada awalnya sudah terpenuhi.

```
do
{
  // Laksanakan rencana
}
while (someVariable ?? value);
```

## H. **Digital I/O**

*Board* NodeMCU mempunyai jumlah pin yang berlabel digital D0 – D8 sebanyak 9 dengan pengalamatan D0 – D8.

### 1. **pinMode(pin, mode)**

Biasa digunakan dalam `void setup()` untuk mengkonfigurasi pin apakah sebagai INPUT atau OUTPUT. Arduino digital pin secara *default* dikonfigurasi sebagai input sehingga untuk mengubahnya harus menggunakan operator `pinMode(pin, mode)`.

```
pinMode (D4, OUTPUT); // mengeset pin D4 sebagai output
digitalWrite(D4, HIGH); // mengeset keluaran nilai HIGH (3.3 volt)
pada pin D4
```

### 2. **digitalRead(pin)**

Membaca nilai dari pin yang kita kehendaki dengan hasil HIGH atau LOW.

```
value = digitalRead(D2); // mengeset 'value' sama dengan nilai pin D2
```

### 3. **digitalWrite(pin, value)**

Digunakan untuk mengeset pin yang kita kehendaki dalam kondisi level tegangan HIGH atau LOW (nyala atau mati). Pin digital NodeMCU sebanyak 9 (D0 – D8).

```
digitalWrite (D4, HIGH); // set pin D4 ke kondisi HIGH
```

## I. **Analog I/O**

Selain pin digital, NodeMCU dilengkapi juga oleh pin analog yang berfungsi untuk membaca input tegangan variabel antara 0 – 3.3V volt dengan resolusi ADC (*Analog to Digital*) 10-bit. Pada NodeMCU ada 1 pin yang berlabel analog input dengan pengalamatan A0. Tidak seperti pin digital



yang dapat difungsikan selain sebagai digital input dapat juga sebagai digital output, namun pada pin analog hanya dapat berfungsi sebagai analog input saja tanpa ada output, apabila kamu ingin membuat NodeMCU mengeluarkan output tegangan analog dapat dilakukan dengan menggunakan pin digital (D1, D2, D3, D4, D5, D6, D7, D8), tegangan analog yang dikeluarkan NodeMCU adalah dengan memanipulasi output digital secara pulsa.

### 1. **analogRead(pin)**

Membaca nilai pin analog yang memiliki resolusi 10-bit. Fungsi ini hanya dapat bekerja pada analog pin (A0). Hasil dari pembacaan berupa nilai integer dengan *range* 0 sampai 1024.

```
value = analogRead(A0); // mengeset 'value' sama dengan nilai analog pin A0
```

### 2. **analogWrite(pin, value)**

Mengirimkan nilai analog dengan metode *Pulse Width Modulation* (PWM) ke pin (D1, D2, D3, D4, D5, D6, D7, D8). Nilai yang dapat digunakan adalah dari 0 - 255.

```
analogWrite(D4, 128); // mengeluarkan nilai tegangan analog ke pin D4
```

## J. **Time**

### 1. **delay(ms)**

Menghentikan program untuk sesaat sesuai dengan yang dikehendaki, satuannya dalam *millisecond*.

```
delay(1000); // menunggu selama satu detik
```

### 2. **millis()**

Mengembalikan nilai dalam *millisecond* dihitung sejak NodeMCU *board* menyala. Penampungnya harus dengan tipe data long integer.

```
value = millis(); // mengatur variabel value sama dengan millis()
```

## K. **Math**

### 1. **min(x,y)**

Membandingkan dua variabel dan akan mengembalikan nilai yang paling kecil.

```
value = min(value, 100); // mengatur variabel value sebagai nilai yang paling kecil dari kedua nilai
```

## 2. **max(x,y)**

Membandingkan dua variabel dan akan mengembalikan nilai yang paling besar.

```
value = max(value, 100); // mengatur variabel 'value' sebagai nilai yang paling besar dari kedua nilai
```

## L. **Random**

### 1. **randomSeed(seed)**

Mengeset sebuah nilai sebagai titik awal fungsi *random()*.

```
randomSeed(value); // mengeset variabel 'value' dengan nilai acak
```

### 2. **random(min,max)**

Menghasilkan sebuah bilangan acak pada *range* yang di batasi oleh angka *min* dan *max*.

```
value = random(100, 200); // mengeset 'value' ke nilai acak antara 100 - 200
```

## M. **Serial**

### 1. **Serial.begin(rate)**

*Statement* ini digunakan untuk mengaktifkan komunikasi serial dan mengatur *baudrate*.

```
void setup()
{
  Serial.begin(115200); // buka serial pada baudrate 115200 bps
}
```

### 2. **Serial.println(data)**

Mengirimkan data ke serial *port*, diikuti oleh karakter *Carriage Return* dan *Line Feed* (CR dan LF) atau yang kita kenal kode untuk *Enter*. Perintah ini lebih sering digunakan karena setelah selesai data di kirim di ikuti *enter* untuk data selanjutnya akan di tampilkan pada baris/alinea baru di bawahnya untuk memudahkan pembacaan data.

```
Serial.println(100);
```

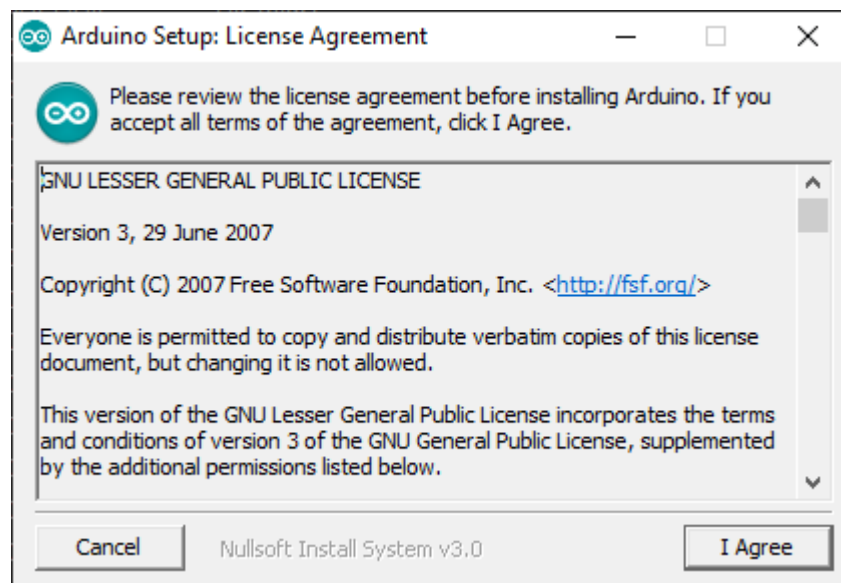
## MODUL 4 | INSTALASI DAN KONFIGURASI ARDUINO IDE

Dalam modul ini akan dijelaskan langkah-langkah melakukan instalasi *software* Arduino IDE dan fungsi-fungsi yang terdapat dalam *software* Arduino IDE.

### A. Instalasi Software Arduino IDE pada Sistem Operasi Windows

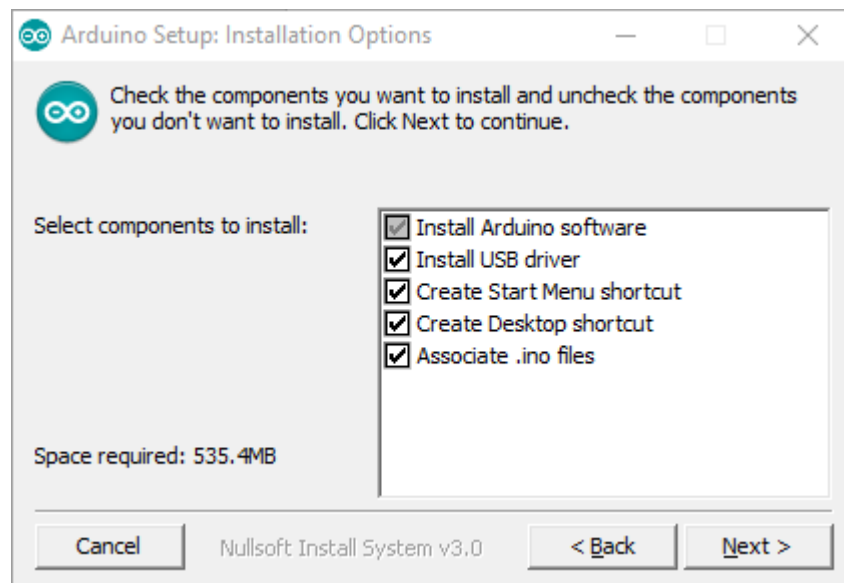
Anda dapat mengunduh *software* Arduino IDE pada laman web Arduino melalui *link* berikut <https://www.arduino.cc/en/software>. Pada laman web tersebut dapat diunduh *software* Arduino IDE yang sesuai dengan sistem operasi yang digunakan. Setelah mengunduh *software* yang tersedia pada laman tersebut langkah selanjutnya adalah melakukan proses instalasi pada komputer. Berikut langkah-langkah instalasi *software* Arduino IDE pada komputer:

1. Unduh *software* Arduino IDE melalui Link Berikut <https://www.arduino.cc/en/software>.
2. *Double click* pada file `arduino-1.8.19-windows.exe` yang telah di unduh (ini adalah versi terbaru dari *Software* Arduino IDE saat panduan ini dibuat), kemudian akan muncul tampilan *License Agreement* dan *click* tombol *I Agree*.



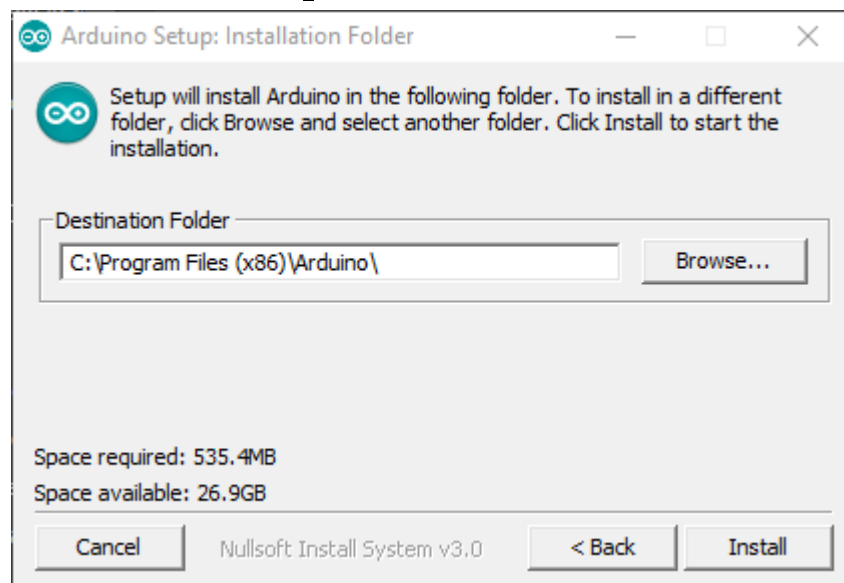
Gambar 4.1. License Agreement

3. Selanjutnya akan muncul tampilan *Installation Options* seperti pada gambar di bawah kemudian centang seluruh komponen yang akan di instal dan klik tombol *Next >*.



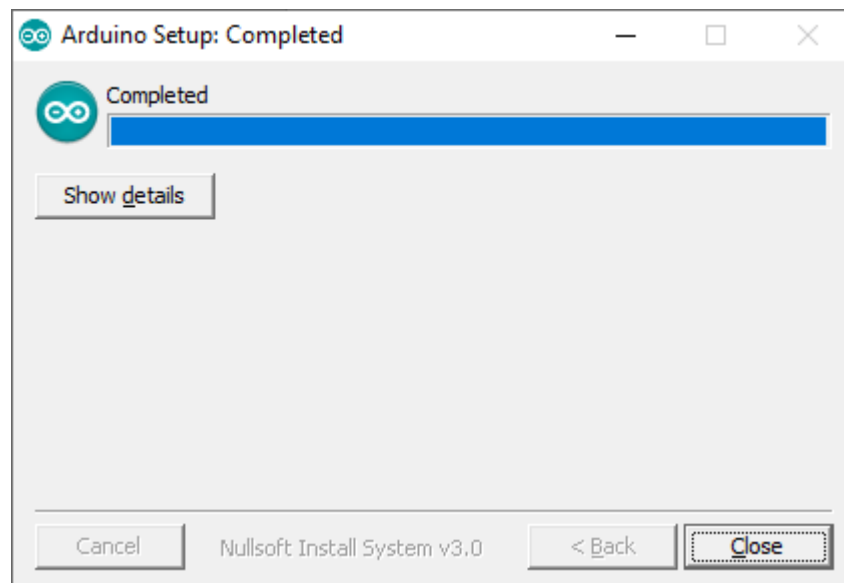
Gambar 4.2 Installation Options

4. Selanjutnya akan muncul tampilan *Installation Folder* seperti pada gambar di bawah, pada tampilan ini anda bisa mengatur folder instalasi dengan menekan tombol *Browse...* kemudian atur folder instalasi dan klik tombol *Install*.



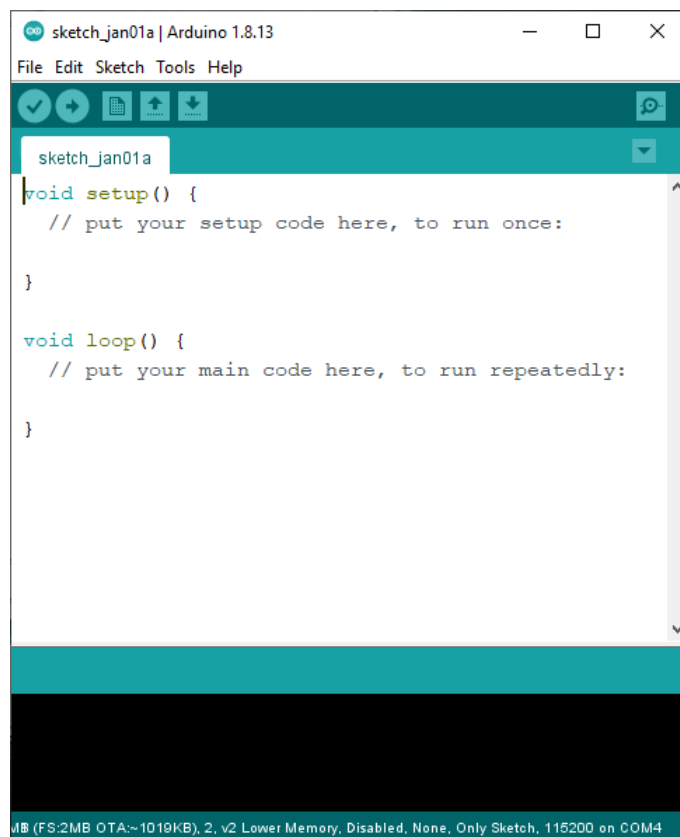
Gambar 4.3. Installation Folder

5. Setelah seluruh proses instalasi selesai kemudian akan muncul tampilan seperti gambar di bawah dan dapat menutup proses instalasi dengan menekan tombol *Close*.



Gambar 4.4. Installation Completed

6. Sekarang anda dapat membuka program Arduino IDE melalui *shortcut* aplikasi yang terdapat pada menu *start* atau *dektop*.

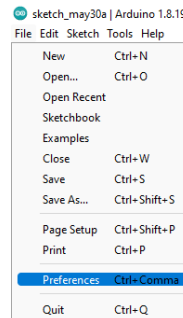


Gambar 4.5. Tampilan utama *software* Arduino IDE

## B. Instalasi Board ESP8266

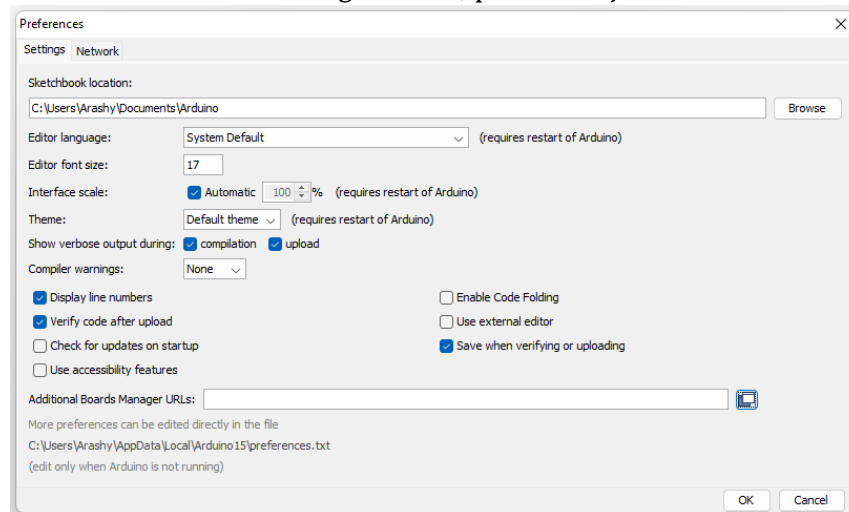
Sebelum memulai instalasi board ESP8266, pastikan telah menginstal Arduino IDE versi terbaru. Berikut petunjuk instalasi *board* ESP8266 pada Arduino IDE:

1. Pada Arduino IDE, pilih **File > Preferences**



Gambar 4.6. Menu *Preferences* Arduino IDE

2. Pada *Additional Board Manager URLs*, pilih ikon jendela

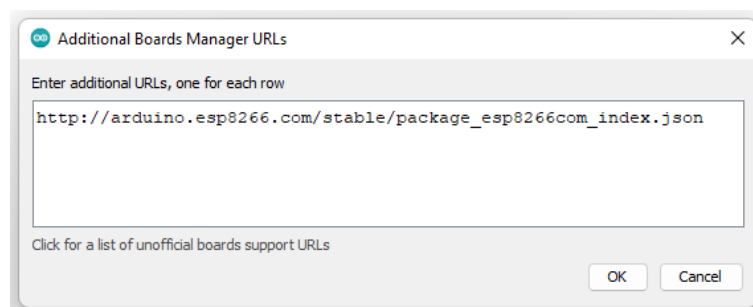


Gambar 4.7. *Preferences Settings* Arduino IDE

3. Masukkan URL *board* ESP8266 lalu pilih OK

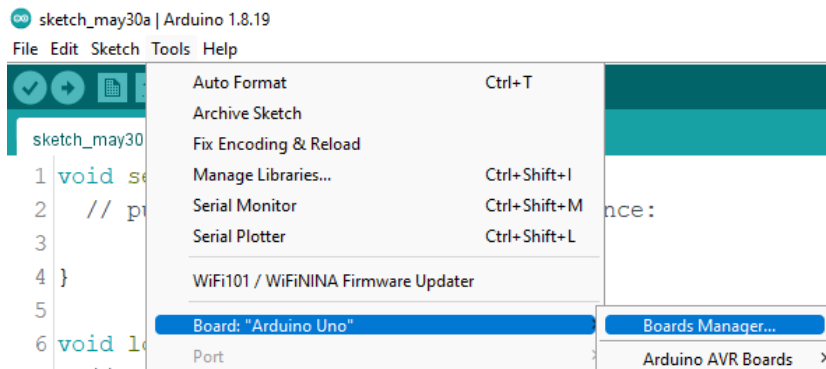
ESP8266 URL *Board*:

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)



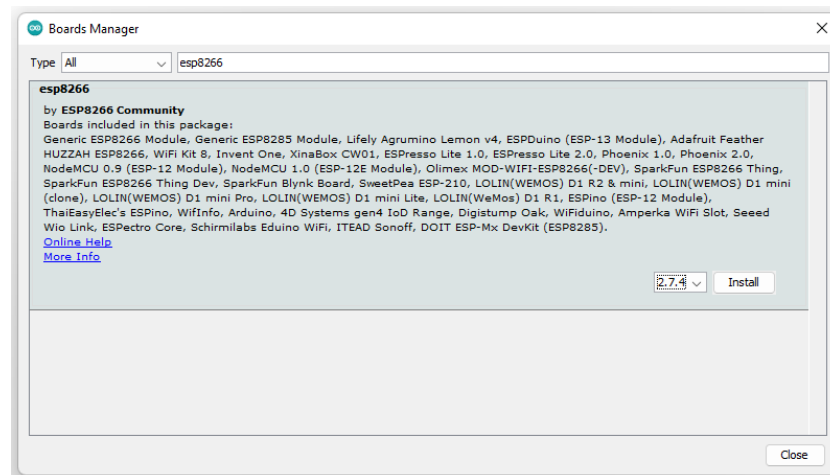
Gambar 4.8. *Additional Boards Manager URLs*

4. Buka *Board Manager*, pilih **Tools > Board > Boards Manager...**



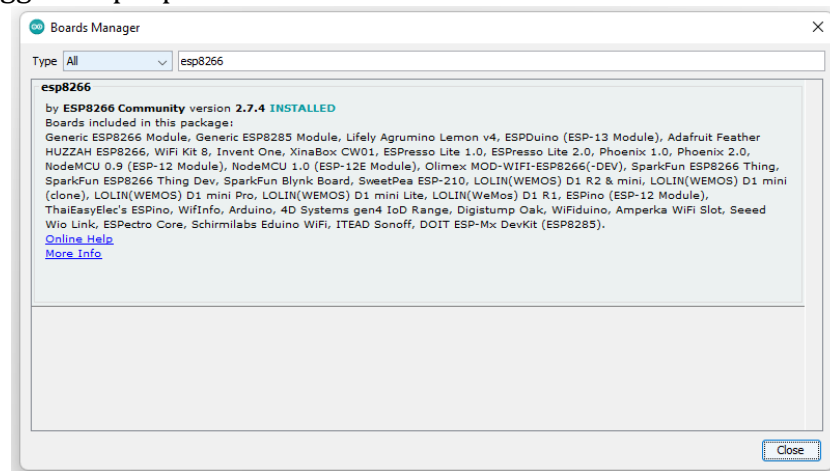
Gambar 4.9. Menu *Board Manager* Arduino IDE

5. Cari ESP8266, pilih esp8266 by *ESP8266 Community*.  
Pastikan memilih **versi 2.7.4** kemudian Install



Gambar 4.10. *Boards Manager* Arduino IDE

6. Tunggu sampai proses instalasi selesai



Gambar 4.11. *Board* ESP826 berhasil diinstal

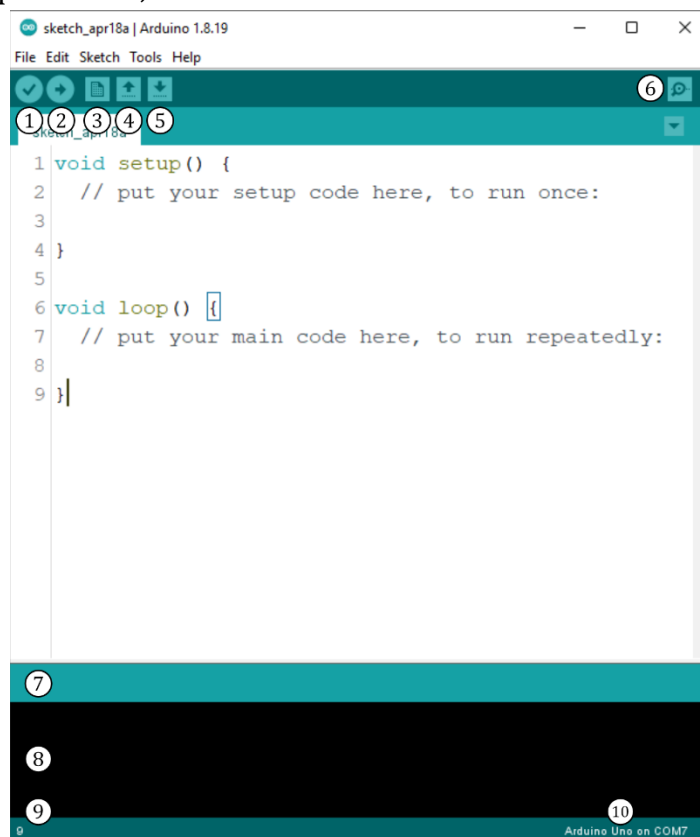


### C. Menambahkan Library

Download library pada laman web Electins melalui *link* berikut <https://www.electins.id/iot-kit/>. Ekstrak dan pindahkan semua folder ke **Documents > Arduino > Libraries**. Library siap digunakan.

### D. Pengenalan Software Arduino IDE

Seperti teks editor pada umumnya yaitu memiliki fitur untuk *cut/paste* dan untuk *find/replace* teks. Pada bagian keterangan aplikasi memberikan pesan balik saat menyimpan dan mengekspor dan juga sebagai tempat menampilkan kesalahan. Konsol log menampilkan output teks dari Arduino Software (IDE), termasuk pesan kesalahan yang lengkap dan informasi lainnya. Pojok kanan bawah jendela menampilkan papan dikonfigurasi dan *port* serial. Tombol *toolbar* memungkinkan Anda untuk memverifikasi dan meng-*upload* program, membuat, membuka, dan menyimpan *sketch*, dan membuka monitor serial.



Gambar 4.12. Tampilan dan keterangan *software* Arduino IDE

1. **Verify** pada versi sebelumnya dikenal dengan istilah *Compile*. Sebelum aplikasi di-*upload* ke *board* Arduino, biasakan untuk memverifikasi terlebih dahulu *sketch* yang dibuat. Jika ada kesalahan pada *sketch*,

nanti akan muncul *error*. Proses *Verify/Compile* mengubah *sketch* ke *binary code* untuk di-*upload* ke mikrokontroler.


2. **Upload** tombol ini berfungsi untuk meng-*upload sketch* ke board Arduino. Walaupun kita tidak mengklik tombol *verify*, maka *sketch* akan di-*compile*, kemudian langsung di-*upload* ke board. Berbeda dengan tombol *verify* yang hanya berfungsi untuk memverifikasi *source code* saja.
3. **New Sketch** Membuka *window* dan membuat *sketch* baru.
4. **Open Sketch** Membuka *sketch* yang sudah pernah dibuat. *Sketch* yang dibuat dengan IDE Arduino akan disimpan dengan ekstensi file *.ino*
5. **Save Sketch** menyimpan *sketch*, tapi tidak disertai dengan meng-*compile*.
6. **Serial Monitor** Membuka *interface* untuk komunikasi serial.
7. **Keterangan Aplikasi** pesan yang dilakukan aplikasi akan muncul di sini, misal “*Compiling*” dan “*Done Uploading*” ketika kita meng-*compile* dan meng-*upload sketch* ke board Arduino
8. **Konsol log** Pesan yang dikerjakan aplikasi dan pesan-pesan tentang *sketch* akan muncul pada bagian ini. Misal, ketika aplikasi meng-*compile* atau ketika ada kesalahan pada *sketch* yang kita buat, maka informasi *error* dan baris akan diinformasikan di bagian ini.
9. **Line Number** bagian ini akan menunjukkan posisi baris kursor yang sedang aktif pada *sketch*.
10. **Informasi Board dan Port** Bagian ini menginformasikan *port* yang dipakai oleh board Arduino.

#### E. Menginstal Driver CH340

NodeMCU Lolin V3 menggunakan *chip* serial CH340 untuk komunikasi serial dengan komputer. Jika *driver* CH340 belum terinstal pada komputer maka *port* pada Arduino IDE tidak bisa dipilih. Berikut langkah-langkah menginstal *driver* CH340 pada sistem operasi Windows:

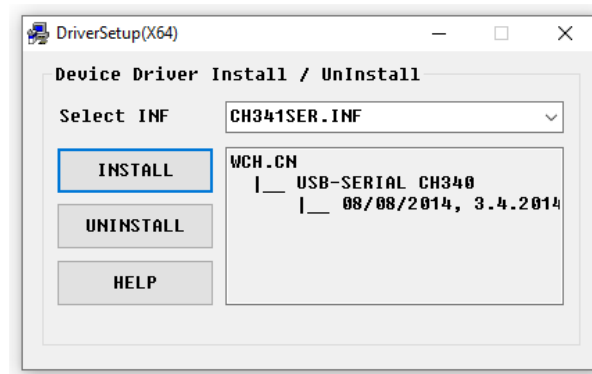
1. Unduh *driver* CH340 pada *link* berikut <https://www.electins.id/iot-kit/>.

2. Ekstrak file yang sudah diunduh sehingga akan tampil seperti gambar berikut.

Name	Date modified	Type	Size
 CH341SER.EXE	20/12/2017 15:12	Application	238 KB

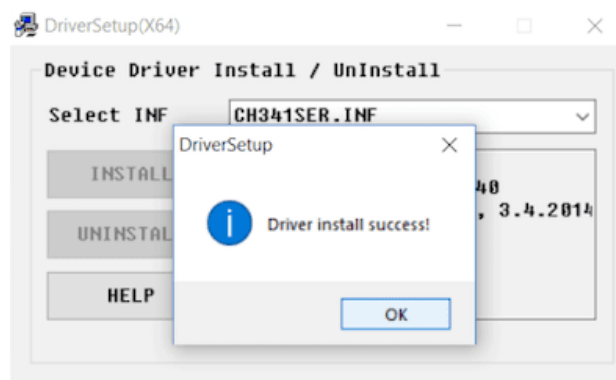
Gambar 4.13. Driver CH340

3. Buka file untuk melakukan instalasi *driver*, maka akan tampil seperti gambar berikut.



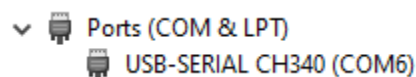
Gambar 4.14. Instalasi driver CH340

4. Pilih INSTALL untuk menginstal *driver*, jika berhasil akan tampil seperti gambar berikut.



Gambar 4.15. Driver CH340 berhasil diinstal

5. *Driver* berhasil diinstal, untuk memastikan driver telah terinstal hubungkan kabel USB dari komputer ke NodeMCU pada trainer, lalu buka *Device Manager* kemudian pilih *Ports* jika berhasil tampil seperti gambar berikut.



Gambar 4.16. USB Serial CH340 pada Device Manager

## MODUL 5 | INSTALASI DAN KONFIGURASI APLIKASI IoT KIT

Modul ini akan dijelaskan langkah-langkah instalasi dan petunjuk penggunaan aplikasi IoT KIT pada *smartphone* android.

### A. Instalasi Aplikasi IoT KIT

Anda dapat mengunduh aplikasi IoT KIT pada laman web Electins melalui *link* berikut <https://www.electins.id/iot-kit/>. Pada laman web tersebut dapat mengunduh file pendukung IoT Starter Kit. Berikut langkah-langkah instalasi aplikasi IoT pada *smartphone* android:

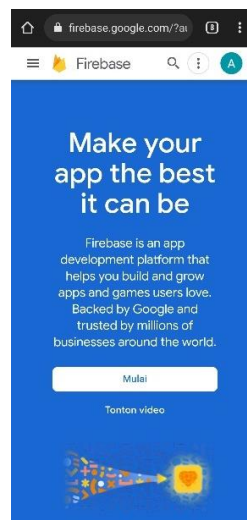
1. Unduh aplikasi IoT KIT pada laman Electins melalui *link* berikut <https://www.electins.id/iot-kit/>.
2. Izinkan instalasi aplikasi dari sumber yang tidak dikenal.
3. Instal aplikasi IoT KIT, tunggu sampai proses selesai.
4. Proses instalasi telah berhasil.

### B. Registrasi Firebase Realtime Database

Sebelum melakukan registrasi akun, perlu disiapkan sebuah akun gmail yang telah *login* pada *smartphone* android untuk mendaftarkan *database* yang nantinya akan di sinkronkan dengan aplikasi IoT KIT sebagai *dashboard/User Interface* (UI) dalam membuat proyek berbasis IoT.

Proyek yang akan dikerjakan dalam modul ini menggunakan *database* dari google yaitu Firebase *Real-Time Database*, untuk lebih mudah berikut langkah-langkah membuat *database* menggunakan Chrome dari *smartphone*:

1. Pada Chrome masuk ke alamat firebase melalui *link* berikut <https://firebase.google.com/>.
2. Pada halaman firebase pilih mulai.



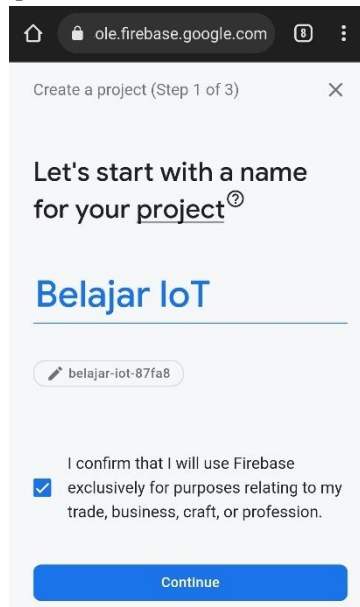
Gambar 5.1. Halaman Firebase

3. Untuk membuat *database*, pilih *create a project*



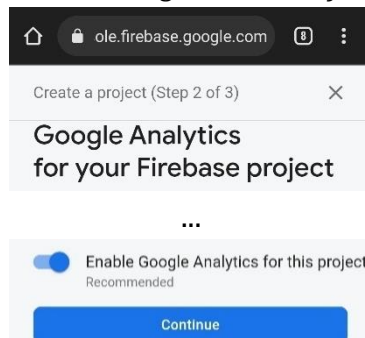
Gambar 5.2. Halaman pembuatan proyek Firebase

4. Buat nama proyek, pada *project name* lalu centang pernyataan konfirmasi, kemudian pilih *Continue*.



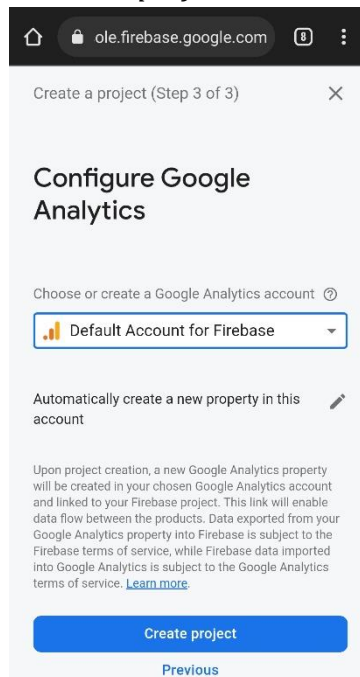
Gambar 5.3. Halaman pembuatan nama proyek

5. Pilih *Continue* pada halaman *Google Analytics for your Firebase project*



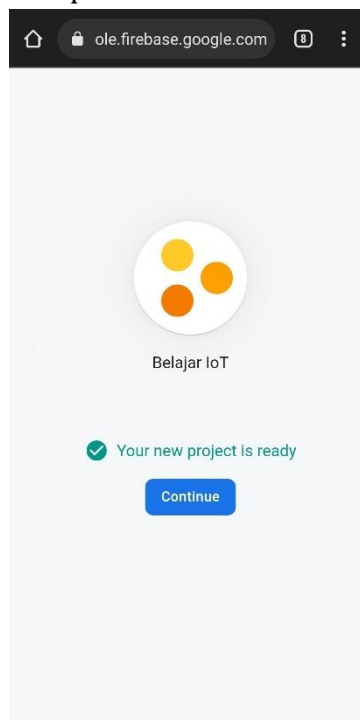
Gambar 5.4. Halaman Aktivasi *Google Analytics*

6. Pilih *Default Account for Firebase*, pada halaman *Configure Google Analytics* kemudian pilih *Create project*.



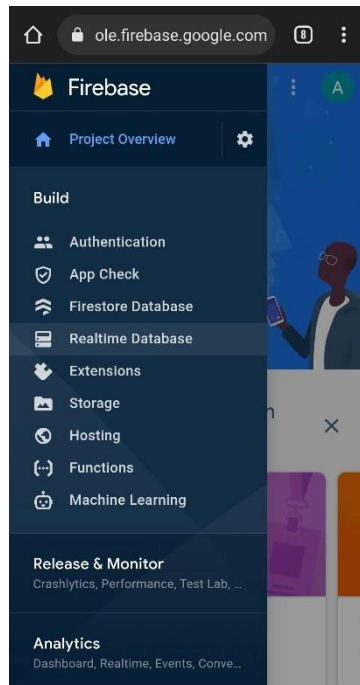
Gambar 5.5. Halaman Konfigurasi *Google Analytics*

7. Tunggu beberapa saat sampai *project* berhasil di buat dan akan tampil seperti berikut, kemudian pilih *Continue*.



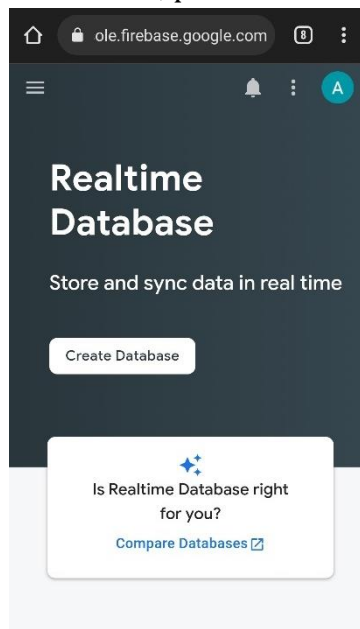
Gambar 5.6. Proyek Firebase berhasil dibuat

8. Secara otomatis akan masuk ke *project* yang telah dibuat. Untuk membuat *database realtime*, pilih menu pada pojok kiri atas, kemudian pilih *Realtime Database*.



Gambar 5.7. Halaman Menu Firebase

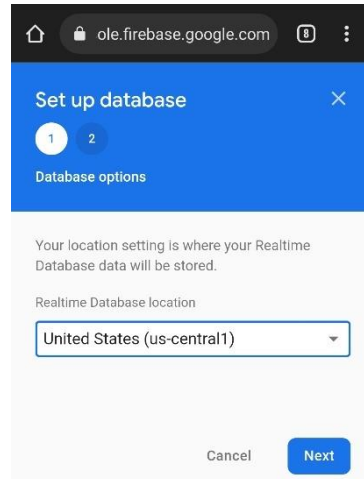
9. Pada halaman *Realtime Database*, pilih *Create Database*



Gambar 5.8. Halaman Pembuatan *Realtime Database*

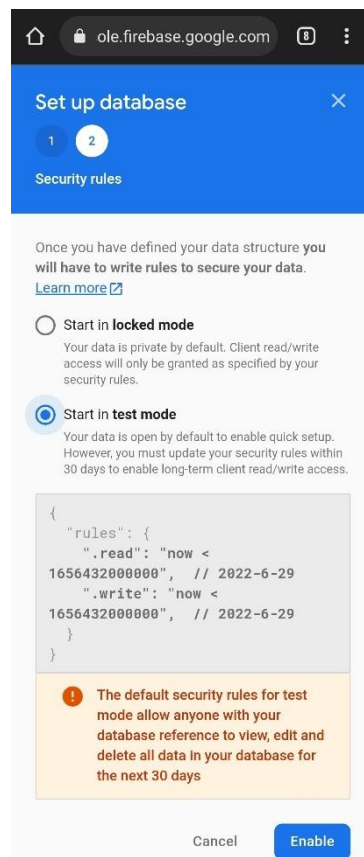


10. Pada halaman *Set up database (1)*, pilih United States (us-central1) sebagai lokasi *realtime database* kemudian pilih *Next*.



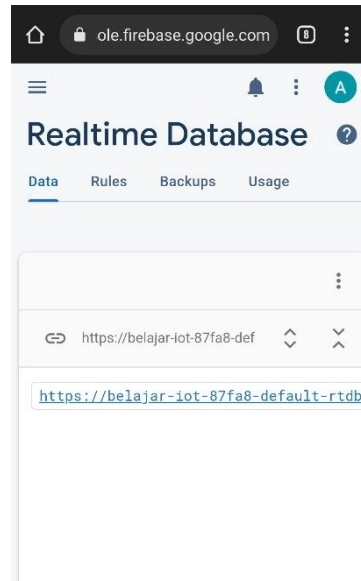
Gambar 5.9. Halaman Pengaturan Lokasi Database

11. Pada halaman *Set up database (2)*, pilih *Start in test mode* pada *Security rules* kemudian pilih *Enable* untuk mengatur *database* agar bisa baca dan tulis data.



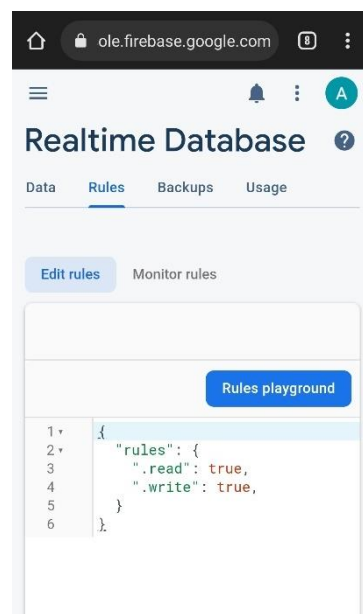
Gambar 5.10. Halaman Pengaturan *Security Rules Database*

12. Setelah selesai akan masuk pada halaman *Data Realtime Database*.



Gambar 5.11. Halaman *Data Realtime Database*

Sampai di sini, *Firebase Realtime Database* berhasil dibuat. Karena pada *security rules, test mode* hanya mengizinkan baca tulis data selama 1 bulan sejak *database* dibuat, maka *security rules* harus di perbarui setiap sebulan, untuk mengaktifkan baca tulis data secara permanen pilih tab *Rules* pada halaman *Realtime Database* kemudian ubah nilai *.read* dan *.write* menjadi *true* kemudian pilih *publish* seperti gambar berikut:

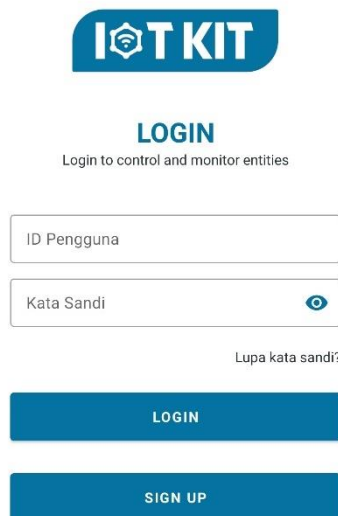


Gambar 5.12. Mengubah *Security Rules Realtime Database*

### C. Registrasi Akun IoT KIT

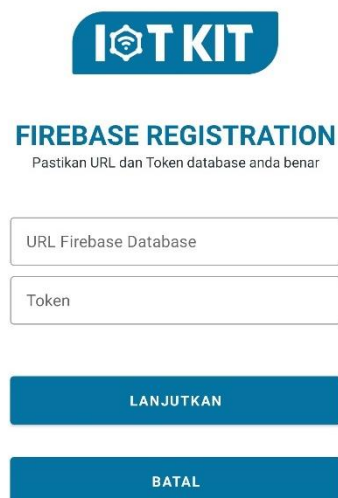
Setelah membuat *database* pada *Firebase Realtime Database*, proses selanjutnya adalah melakukan registrasi pada aplikasi IoT KIT sehingga nantinya yang *database* yang sebelumnya telah dibuat dapat digunakan dan di sinkronkan pada aplikasi IoT KIT. Berikut petunjuk melakukan registrasi pada aplikasi IoT KIT:

1. Buka Aplikasi IoT KIT, Tekan tombol SIGN UP.



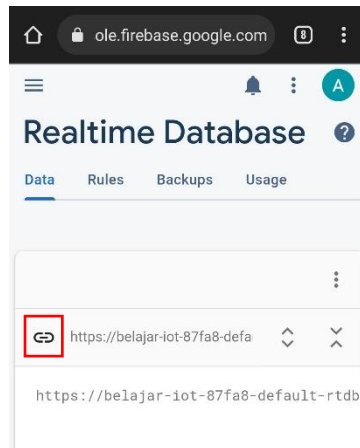
Gambar 5.13. Halaman Login Aplikasi IoT KIT

2. Pada halaman registrasi Firebase terdapat 2 *form* yaitu URL *Firebase Database* dan *Token*.



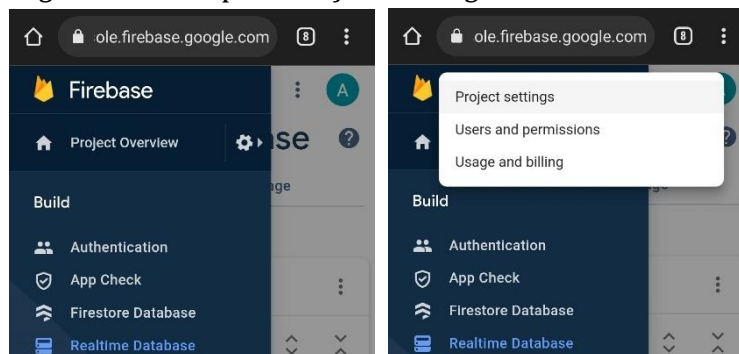
Gambar 5.14. Halaman Registrasi *Firebase Database*

- Untuk mendapatkan URL, kembali pada halaman Data *Realtime Database*. Terdapat URL yang akan digunakan untuk registrasi. Klik ikon untuk menyalin URL untuk di paste pada URL *form*.



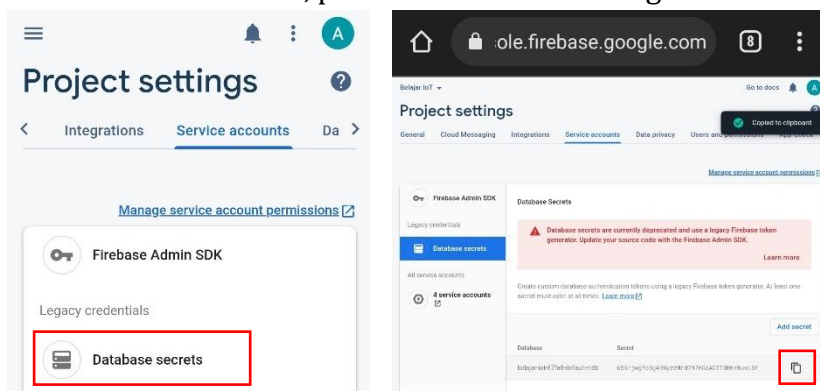
Gambar 5.15. URL Firebase *Realtime Database*

- Untuk mendapatkan *Token*, pilih menu pada pojok kanan atas > pilih ikon *settings* kemudian pilih *Project settings*.



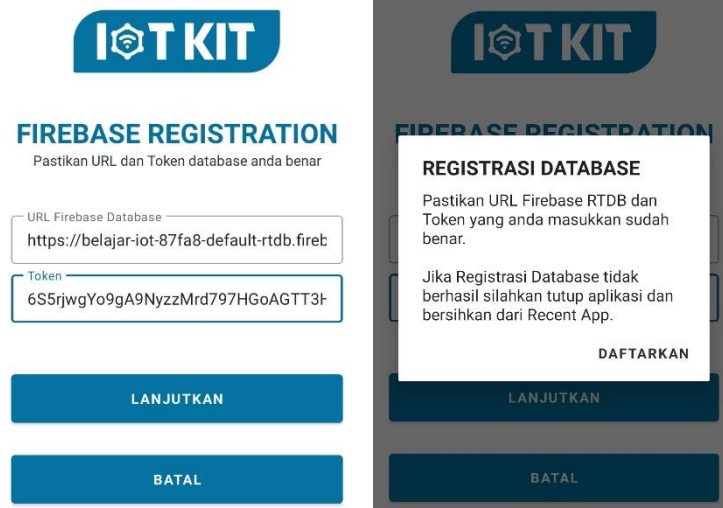
Gambar 5.16. *Project Settings* Firebase

- Pada halaman *Project Settings*, geser ke kanan dengan menekan ikon arah kanan kemudian pilih *Service accounts* lalu pilih *Database secrets*. Pada *database seceret*, pilih *Show* lalu salin dengan menekan ikon *copy*.



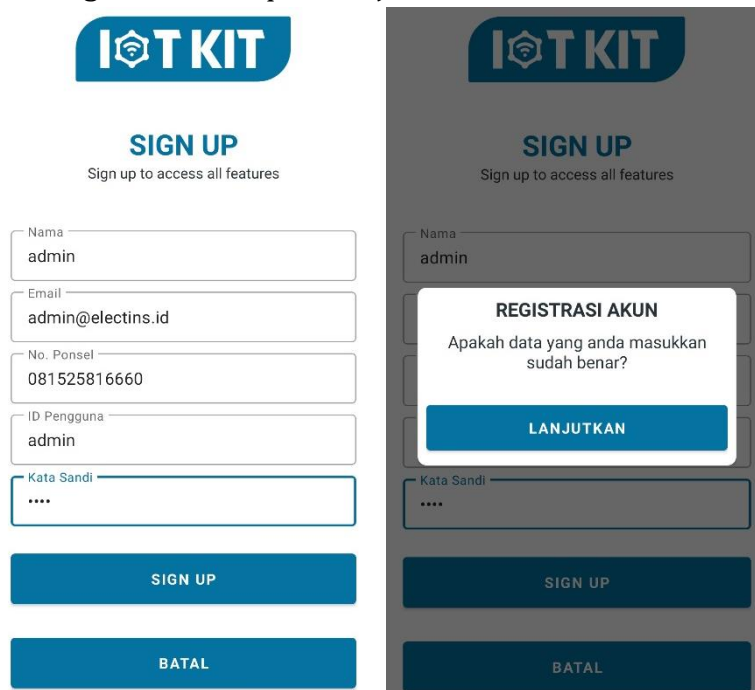
Gambar 5.17. *Token* Firebase *Database*

6. Paste URL Firebase *Realtime Database* dan *Token* pada halaman registrasi firebase *database* kemudian pilih lanjutkan. Pada *pop-up* dialog konfirmasi pilih daftarkan.



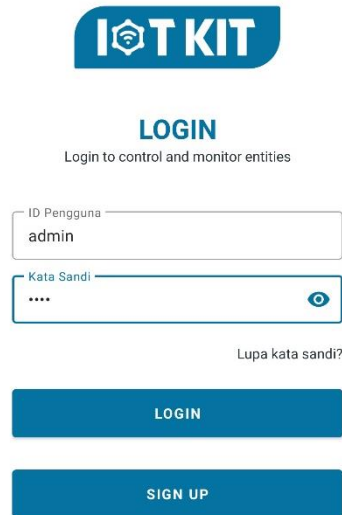
Gambar 5.18. Registrasi *Firestore Database*

7. Pada halaman *Sign Up*, lengkapi *form* dengan mengisi Nama, Email, No. Ponsel, ID Pengguna dan Kata sandi. Penting untuk mengingat ID Pengguna dan Kata Sandi untuk *login* ke *dashboard* aplikasi IoT KIT. Setelah mengisi *form* tekan tombol SIGN UP. Pada *pop-up* dialog konfirmasi registrasi akun pilih lanjutkan.



Gambar 5.19. Registrasi Akun Pengguna IoT KIT

8. Tunggu beberapa saat akan dialihkan ke halaman *login*. Masukkan ID Pengguna dan kata sandi yang telah di daftarkan kemudian tekan tombol LOGIN.



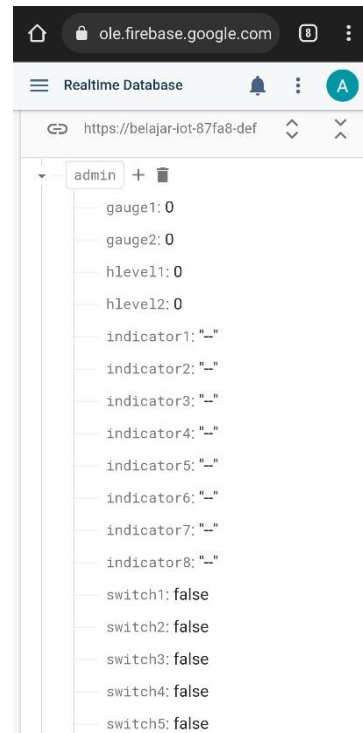
Gambar 5.20. Halaman Login Aplikasi IoT KIT

9. Jika *login* berhasil akan diarahkan pada halaman *dashboard* aplikasi IoT KIT. Fitur pada aplikasi telah siap digunakan.



Gambar 5.21. Tampilan *Dashboard* Aplikasi IoT KIT

10. Untuk memastikan registrasi telah berhasil, buka *database* pada Data *Realtime Database*, pastikan sudah terisi data yang dibuat otomatis pada saat registrasi seperti gambar berikut.



Gambar 5.22. Data Realtime Database Aplikasi IoT KIT

#### D. Pengenalan Aplikasi IoT KIT

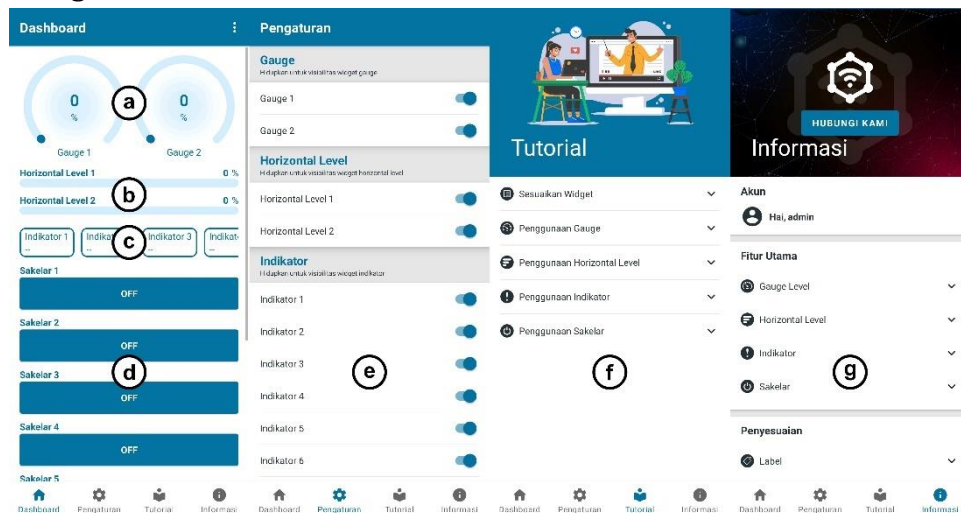
Aplikasi IoT KIT merupakan aplikasi yang dikembangkan oleh [Electins.id](https://electins.id) untuk para penggiat mikrokontroler khususnya pada sektor *Internet of Things* (IoT). Aplikasi ini ditujukan bagi pengguna yang mengembangkan sistem IoT menggunakan *Firebase Realtime Database* sebagai IoT platform. Aplikasi ini sangat cocok untuk pemula yang ingin belajar dan membangun sistem IoT dengan menggunakan *database* sendiri. Aplikasi ini dapat digunakan untuk memantau dan mengontrol suatu perangkat secara *realtime* dengan basis cerdas Industri 4.0 yang dikemas dalam Aplikasi IoT KIT.

Aplikasi IoT KIT mempunyai beberapa halaman yang bisa diakses, seperti *dashboard* sebagai halaman utama untuk menampilkan data dan melakukan pengontrolan, pengaturan untuk visibilitas *widget* yang digunakan, tutorial untuk mendapatkan petunjuk penggunaan *widget* dan informasi untuk data-data terkait aplikasi IoT KIT. Berikut detail halaman aplikasi IoT KIT.



## 1. Halaman Utama

Terdapat 4 halaman utama yang dapat di akses pada aplikasi IoT KIT sebagai berikut:



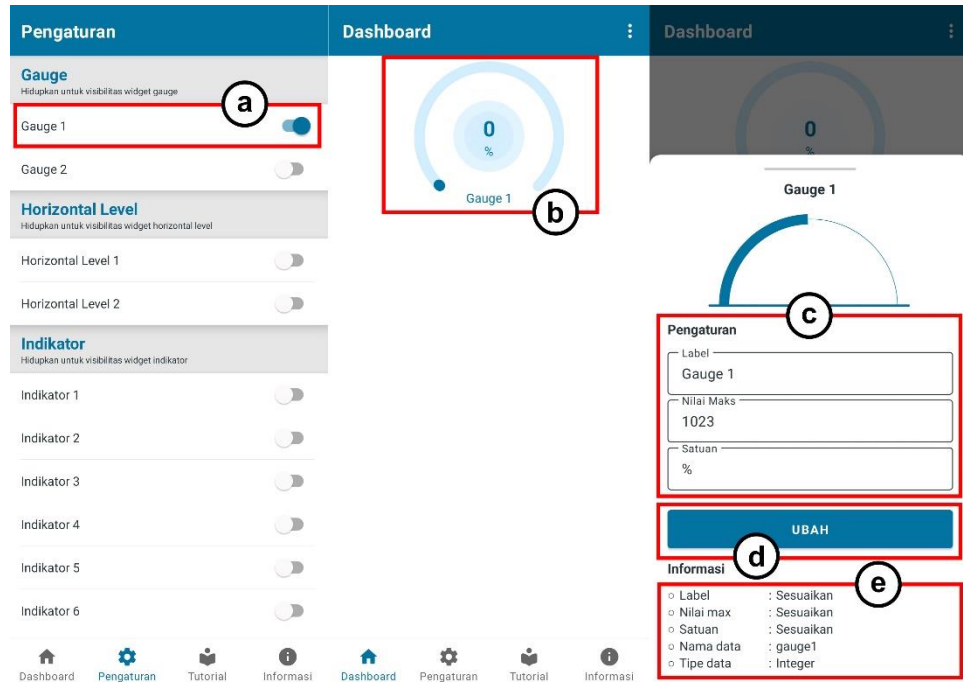
Gambar 5.23. Tampilan Aplikasi IoT KIT

- a. **Gauge level** merupakan *widget* untuk memvisualisasikan data dalam bentuk grafik 270 derajat.
- b. **Horizontal level** merupakan *widget* untuk memvisualisasikan data dalam bentuk bar grafik horizontal.
- c. **Indikator** merupakan *widget* untuk menampilkan data berupa teks. Data dapat berupa keterangan label dan sebagainya.
- d. **Sakelar** merupakan *widget* untuk mengirimkan data ke perangkat IoT untuk mengaktifkan atau memberikan instruksi pada perangkat keras yang terhubung.
- e. **Visibilitas Widget** merupakan pengaturan untuk melakukan *custom* tampilan pada *dashboard*. *Widget* yang tidak dibutuhkan dapat di sembunyikan dengan mengatur visibilitas *widget*.
- f. **Tutorial** merupakan halaman yang memuat petunjuk penggunaan *widget* beserta dengan contoh programnya.
- g. **Informasi** merupakan halaman yang memuat semua informasi tentang aplikasi IoT KIT.

## 2. Penyesuaian Widget

Penyesuaian *widget* dapat dilakukan pada aplikasi IoT KIT dengan tujuan aplikasi dapat menyesuaikan dengan proyek yang dibuat. Misalnya pada proyek hanya menampilkan data suhu dan kelembaban penggunaan *widget gauge* sudah mencukupi sehingga *widget* yang lain dapat disembunyikan melalui pengaturan visibilitas.

Berikut petunjuk melakukan penyesuaian *widget* pada aplikasi IoT KIT:



Gambar 5.24. Custom Widget Aplikasi IoT KIT

#### a. Mengatur visibilitas widget

Masuk pada pengaturan, pilih dan aktifkan untuk menampilkan *widget*. Sebaliknya matikan untuk menyembunyikan *widget*.

#### b. Memuat pengaturan widget

Untuk memuat atribut pengaturan *widget*, tekan dan tahan pada *widget* yang akan diubah sampai dialog pengaturan *widget* muncul.

#### c. Mengatur atribut widget

Setiap *widget* mempunyai atribut yang dapat diatur seperti label, nilai maksimal dan satuan.

#### d. Menyimpan pengaturan

Setelah mengatur atribut *widget* simpan pengaturan dengan menekan tombol ubah, dengan demikian *widget* telah berhasil disesuaikan.

#### e. Informasi

Terdapat informasi tambahan seperti label, nilai maks, satuan yang dapat disesuaikan, nama data (*key*) dan tipe data (*value*) yang digunakan sesuai dengan *database*

### 3. Logout dari Aplikasi IoT KIT

*Logout* dari aplikasi IoT KIT dapat dilakukan pada halaman *dashboard* melalui menu dipojok kanan atas atau melalui halaman formasi paling bawah terdapat tombol *logout*.



Gambar 5.25. *Logout* aplikasi IoT KIT

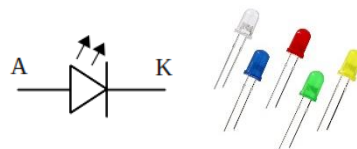
Jika proses *logout* berhasil, maka akan keluar dari halaman utama aplikasi dan diarahkan ke halaman *login*. Anda bisa *login* kembali dengan menggunakan ID Pengguna dan Kata Sandi yang sebelumnya terdaftar.

## MODUL 6 | ANTARMUKA DENGAN LED, BUZZER DAN RELAY

Pada modul ini akan membahas mengenai antarmuka dengan LED (*Light Emitting Diode*), Buzzer dan Relay dengan proyek mengontrol *Device* dengan aplikasi IoT KIT.

### A. Pengenalan LED

*Light Emitting Diode* atau LED adalah komponen elektronika yang dapat memancarkan cahaya monokromatik ketika diberikan tegangan maju. LED merupakan keluarga Dioda yang terbuat dari bahan semikonduktor. Warna-warna Cahaya yang dipancarkan oleh LED tergantung pada jenis bahan semikonduktor yang dipergunakannya.



Gambar 6.1. Simbol dan Bentuk LED

Ketika LED dialiri tegangan maju atau bias *forward* yaitu dari *Anoda* (P) menuju ke *Katoda* (K), kelebihan elektron pada N-Type material akan berpindah ke wilayah yang kelebihan *hole* (lubang) yaitu wilayah yang bermuatan positif (*P-Type material*). Saat elektron berjumpa dengan *hole* akan melepaskan *photon* dan memancarkan cahaya monokromatik (satu warna).

Spesifikasi LED :

- Tegangan kerja : 1.2 – 4.0 Volt (Berdasarkan warna)
- Arus Kerja : 20 mA
- Panjang Gelombang : 430 – 940 nm (Berdasarkan warna)

### B. Pengenalan Buzzer

Buzzer adalah sebuah komponen elektronika yang dapat mengubah sinyal listrik menjadi getaran suara. Buzzer yang merupakan sebuah perangkat audio yang sering digunakan pada rangkaian alarm anti-maling, Bel Rumah dan alat peringatan bahaya lainnya. Jenis Buzzer yang sering ditemukan dan digunakan adalah Buzzer yang berjenis *Piezoelectric*. Buzzer bekerja dengan baik dalam menghasilkan frekuensi di kisaran 1 – 5 kHz hingga 100 kHz.

Buzzer bekerja dengan tegangan listrik yang diberikan ke bahan *Piezoelectric* menyebabkan gerakan mekanis, gerakan tersebut kemudian diubah menjadi suara atau bunyi yang dapat didengar oleh telinga manusia dengan menggunakan diafragma dan resonator.

Spesifikasi Buzzer:

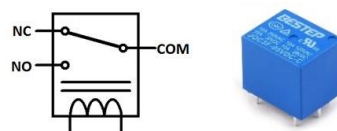
1. Tegangan kerja 3 – 12 Volt
2. Tahanan dalam 16 Ohm
3. Output suara 80-85 dB



Gambar 6.2. Simbol dan bentuk *buzzer*

### C. Pengenalan Relay

Relay adalah Saklar (*Switch*) yang dioperasikan secara listrik dan merupakan komponen Elektromekanikal yang terdiri dari dua bagian utama yakni Elektromagnet (*Coil*) dan Mekanikal (Kontak Saklar/*Switch*). Relay menggunakan prinsip elektromagnetik untuk menggerakkan kontak saklar sehingga dengan arus listrik yang kecil (*low power*) dapat menghantarkan listrik dengan tegangan lebih tinggi. Misalnya saklar listrik 220V AC.



Gambar 6.3. Simbol dan bentuk relay

Kontak Poin (*Contact Point*) Relay 5 pin terdiri dari 2 bagian yaitu :

1. *Normally Close* (NC) yaitu kondisi awal sebelum diaktifkan akan selalu berada di posisi CLOSE (tertutup)
2. *Normally Open* (NO) yaitu kondisi awal sebelum diaktifkan akan selalu berada di posisi OPEN (terbuka)

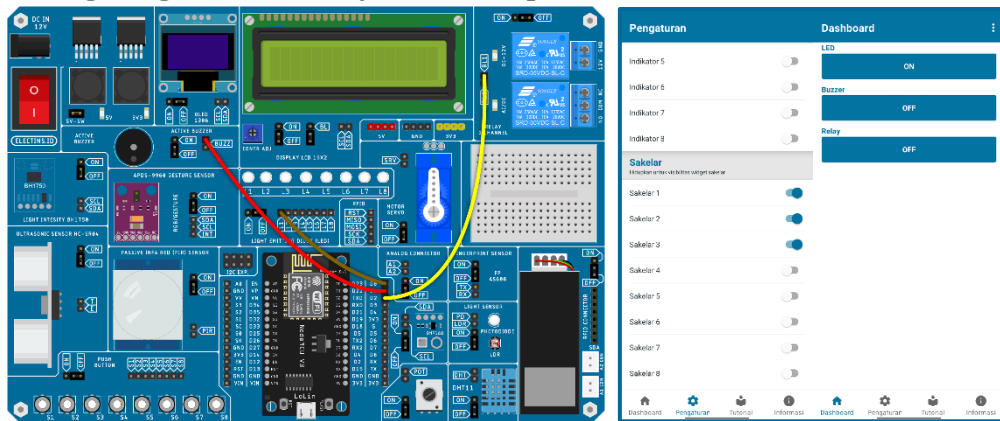
Kumparan *Coil* jika diberikan arus listrik, maka akan timbul gaya Elektromagnet yang kemudian menarik kontak sakelar untuk berpindah dari Posisi sebelumnya (NC) ke posisi baru (NO) sehingga menjadi Saklar yang dapat menghantarkan arus listrik pada posisi barunya (NO). Pada saat tidak dialiri arus listrik, kontak sakelar akan kembali lagi ke posisi Awal (NC). *Coil* yang digunakan oleh relay untuk menarik kontak sakelar ke Posisi (NO) pada umumnya hanya membutuhkan arus listrik yang relatif kecil.

Relay 1 pada trainer dapat digunakan untuk tegangan dengan output DC 12V sedangkan pada relay 2 untuk sakelar tegangan DC maupun AC 220V.

#### D. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. LED                               | 1 Buah |
| 3. Buzzer                            | 1 Buah |
| 4. Relay                             | 1 Buah |
| 5. Kabel <i>Jumper Female-Female</i> | 3 Buah |

#### E. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 6.4. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	HEADER MCU
L1	PIN D0
BUZZ	PIN D1
RL1	PIN D2

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Sakelar 1	LED	-	-
Sakelar 2	Buzzer	-	-
Sakelar 3	Relay	-	-

#### F. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* LED, BUZZ dan RL1 ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LED, Buzzer dan Relay ke posisi ON.
3. Hidupkan trainer.

4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools > Board > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module)*.
7. Pilih *port* yang terhubung pada *Tools > Port > COM x*.
8. *Verify (compile)* dan *Upload* program.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

// PIN LED, Buzzer dan relay
#define LED_PIN D0
#define BUZZ_PIN D1
#define RELAY_PIN D2
void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Mengatur PIN LED, Buzzer dan Relay sebagai OUTPUT
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZ_PIN, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);

  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  // Menampilkan status koneksi dan alamat IP
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Menampilkan versi client firebase
  Serial.printf("Firebase Client v%s\n\n",
```



```

        FIREBASE_CLIENT_VERSION);
// Memulai koneksi dengan database
// Re-koneksi jika WiFi terputus
Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);
}
void loop() {
// Membaca database pada alamat */user/switch1
// Jika bernilai true LED ON
// Jika bernilai false LED OFF
if (Firebase.getString(fbdo, "/" + user + "/switch1")) {
  if(fbdo.to<bool>() == true){
    digitalWrite(LED_PIN, HIGH);
    Serial.println("LED On");
  }else{
    Serial.println("LED Off");
    digitalWrite(LED_PIN, LOW);
  }
}
// Membaca database pada alamat */user/switch2
// Jika bernilai true Buzzer ON
// Jika bernilai false Buzzer OFF
if (Firebase.getString(fbdo, "/" + user + "/switch2")) {
  if(fbdo.to<bool>() == true){
    digitalWrite(BUZZ_PIN, HIGH);
    Serial.println("Buzzer On");
  }else{
    Serial.println("Buzzer Off");
    digitalWrite(BUZZ_PIN, LOW);
  }
}
// Membaca database pada alamat */user/switch3
// Jika bernilai true Relay ON
// Jika bernilai false Relay OFF
if (Firebase.getString(fbdo, "/" + user + "/switch3")) {
  if(fbdo.to<bool>() == true){
    digitalWrite(RELAY_PIN, HIGH);
    Serial.println("Relay On");
  }else{
    Serial.println("Relay Off");
    digitalWrite(RELAY_PIN, LOW);
  }
}
}
}

```

### Keterangan Program:

```

#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

```

Library ESP8266 WiFi dan Firebase ESP8266.

```

#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

```

SSID dan Password WiFi.

```

#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

```

URL dan *token firebase database*.

```
FirebaseData fbdo;
```

Firestore objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user ID* yang didaftarkan pada aplikasi IoT KIT.

```
#define LED_PIN D0  
#define BUZZ_PIN D1  
#define RELAY_PIN D2
```

Pin LED, Buzzer dan Relay masing-masing terhubung ke Pin D0, D1 dan D2.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
pinMode(LED_PIN, OUTPUT);  
pinMode(BUZZ_PIN, OUTPUT);  
pinMode(RELAY_PIN, OUTPUT);
```

Mengatur Pin LED, Buzzer dan Relay sebagai Output.

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firestore Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firestore.begin(DATABASE_URL, API_KEY);  
Firestore.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan *firebase database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
if(Firebase.getString(fbdo, "/" + user + "/switch1")) {  
    ...  
}
```

Membaca *database* pada alamat */user/switch1*, LED ON jika bernilai *true* dan LED OFF jika bernilai *false*.

```
if(Firebase.getString(fbdo, "/" + user + "/switch2")) {  
    ...  
}
```

Membaca *database* pada alamat */user/switch2*, Buzzer ON jika bernilai *true* dan Buzzer OFF jika bernilai *false*.

```
if(Firebase.getString(fbdo, "/" + user + "/switch3")) {  
    ...  
}
```

Membaca *database* pada alamat */user/switch3*, Relay ON jika bernilai *true* dan Relay OFF jika bernilai *false*.

## MODUL 7 | ANTARMUKA DENGAN MOTOR SERVO

Pada modul ini akan membahas mengenai antarmuka dengan Motor Servo dengan proyek mengontrol Motor Servo dengan aplikasi IoT KIT.

### A. Pengenalan Motor Servo

Motor servo adalah sebuah perangkat atau aktuator putar (motor) yang dirancang dengan sistem kontrol umpan balik *loop* tertutup servo, sehingga dapat di *set-up* atau diatur untuk menentukan dan memastikan posisi sudut dari poros output motor. Motor servo merupakan perangkat yang terdiri dari motor DC, serangkaian *gear*, rangkaian kontrol dan potensiometer. Serangkaian *gear* yang melekat pada poros motor DC akan memperlambat putaran poros dan meningkatkan torsi motor servo, sedangkan potensiometer dengan perubahan resistansinya saat motor berputar berfungsi sebagai penentu batas posisi putaran poros motor servo.



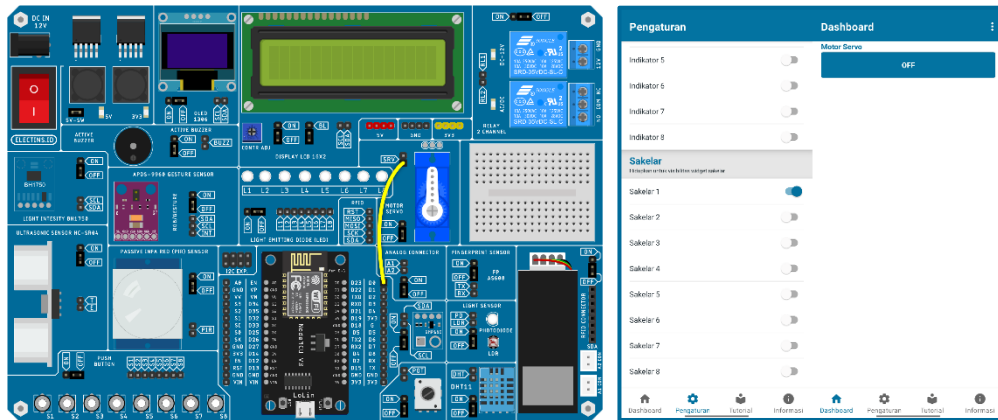
Gambar 7.1. Motor Servo

Motor servo dikendalikan dengan memberikan sinyal modulasi lebar pulsa (*Pulse Wide Modulation/PWM*) melalui kabel kontrol. Lebar pulsa sinyal kontrol yang diberikan akan menentukan posisi sudut putaran dari poros motor servo. Sebagai contoh, lebar pulsa dengan waktu 1,5 ms (mili detik) akan memutar poros motor servo ke posisi sudut  $90^{\circ}$ . Bila pulsa lebih pendek dari 1,5 ms maka akan berputar ke arah posisi  $0^{\circ}$  atau ke kiri (berlawanan dengan arah jarum jam), sedangkan bila pulsa yang diberikan lebih lama dari 1,5 ms maka poros motor servo akan berputar ke arah posisi  $180^{\circ}$  atau ke kanan (searah jarum jam).

### B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. Motor Servo                       | 1 Buah |
| 3. Kabel <i>Jumper Female-Female</i> | 1 Buah |

### C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 7.1. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	HEADER MCU
SRV	PIN D0

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Sakelar 1	Motor Servo	-	-

### D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* SRV dan BUZZ ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* Motor Servo ke posisi ON.
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools* > *Board* > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module).
7. Pilih *port* yang terhubung pada *Tools* > *Port* > COM x.
8. *Verify (compile)* dan *Upload* program.

#### Contoh Program:

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

// Library Servo dan Pin Servo
#include <Servo.h>
```

```
#define SERVO_PIN    D0

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// Servo objek dengan nama servo
Servo servo;
// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Konfigurasi pin pada objek servo
  servo.attach(SERVO_PIN);

  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  // Menampilkan status koneksi dan alamat IP
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Menampilkan versi client firebase
  Serial.printf("Firebase Client v%s\n\n",
    FIREBASE_CLIENT_VERSION);
  // Memulai koneksi dengan database
  // Re-koneksi jika WiFi terputus
  Firebase.begin(DATABASE_URL, API_KEY);
  Firebase.reconnectWiFi(true);
}

void loop() {
  // Membaca database pada alamat */user/switch1
  // Jika nilainya true servo berputar ke posisi 180 derajat
  // Jika nilainya false servo berputar ke posisi 0 derajat
  if(Firebase.getString(fbdo, "/" + user + "/switch1")) {
    if(fbdo.to<bool>() == true){
      servo.write(180);
      Serial.println("Servo 0 derajat");
    }else{
```

```
servo.write(0);  
Serial.println("Servo 180 derajat");  
}  
}  
}
```

**Keterangan Program:**

```
#include <ESP8266WiFi.h>  
#include <FirebaseESP8266.h>
```

Library ESP8266 WiFi dan Firebase ESP8266.

```
#include <Servo.h>  
#define SERVO_PIN D0
```

Library dan Pin Servo terhubung ke Pin D0 NodeMCU

```
#define WIFI_SSID "SSID_WIFI"  
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan Password WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"  
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
FirebaseData fbdo;  
Servo servo;
```

Firestore dan Servo objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
Serial.begin(115200);
```

Serial monitor pada baudrate 115200.

```
servo.attach(SERVO_PIN);
```

Konfigurasi pin servo pada objek servo

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
if(Firebase.getString(fbdo, "/" + user + "/switch1")) {  
  if(fbdo.to<bool>() == true){  
    servo.write(180);  
    Serial.println("Servo 0 derajat");  
  }  
}
```

```
}else{  
  servo.write(0);  
  Serial.println("Servo 180 derajat");  
}  
}
```

Membaca *database* pada alamat `/user/switch1`, servo berputar ke posisi 180 derajat jika bernilai *true* dan berputar ke posisi 0 derajat jika bernilai *false*.

## MODUL 8 | ANTARMUKA DENGAN LCD 16X2 I2C

Pada modul ini membahas mengenai antarmuka dengan LCD 16x2 I2C dengan proyek menampilkan teks pada LCD melalui Aplikasi IoT KIT.

### A. Pengenalan LCD 16x2

LCD atau *Liquid Crystal Display* adalah suatu jenis media *display* (tampilan) yang menggunakan kristal cair (*liquid crystal*) untuk menghasilkan gambar yang terlihat. Teknologi *Liquid Crystal Display* (LCD) atau penampil kristal cair sudah banyak digunakan pada produk-produk seperti layar laptop, layar ponsel, layar kalkulator, layar jam digital, layar multimeter, monitor komputer, televisi, layar *game* portabel, layar termometer digital dan produk-produk elektronik lainnya.

LCD 16x2 merupakan komponen elektronika yang berfungsi untuk menampilkan suatu data dapat berupa karakter, huruf, simbol maupun grafik dengan jumlah kolom 16 karakter dan mempunyai 2 baris. Karena ukurannya yang kecil maka LCD banyak dipasangkan dengan Mikrokontroler. LCD tersedia dalam bentuk modul yang mempunyai pin data, kontrol catu daya, dan pengatur kontras.



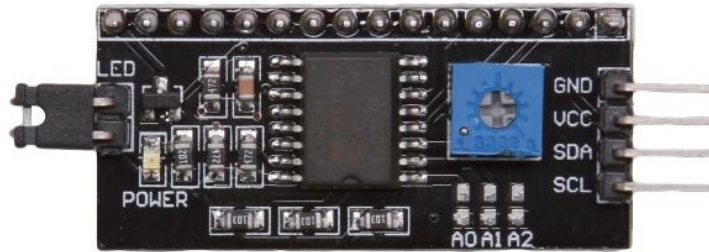
Gambar 8.1. LCD 16x2

LCD 16x2 pinout:

Pin	Fungsi
GND	Catu daya 0 Volt DC
VCC	Catu daya positif 5V DC
Contrast	Kontras tulisan pada LCD
RS (Register Select)	High untuk mengirim data Low untuk mengirim instruksi
R/W (Read/Write)	High untuk mengirim data Low untuk mengirim instruksi
E (Enable)	Untuk mengontrol LCD
D0-D7	Data bus
Backlight +	Lampu latar VCC 5VDC
Backlight -	Lampu latar GND 0VDC



Pada trainer menggunakan modul I2C yang di kendalikan dengan protokol I2C (*Inter Integrated Circuit*) atau TWI (*Two Wire Interface*) sehingga LCD dapat dikontrol hanya dengan dua jalur data SDA dan SCL.

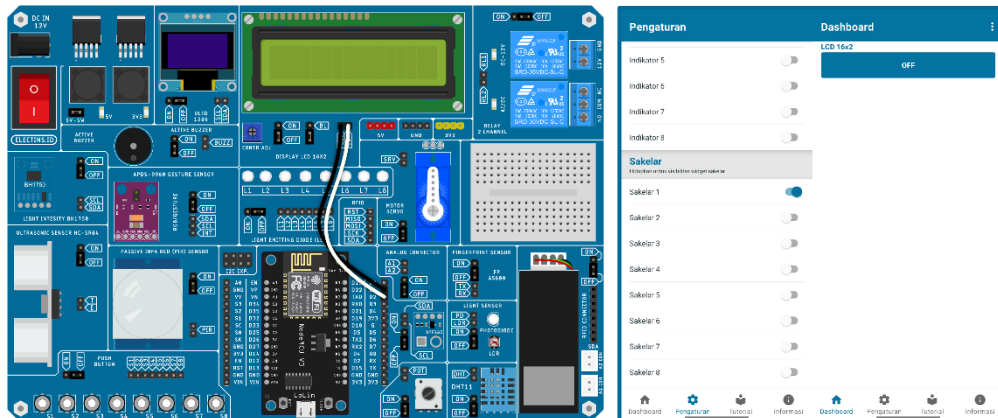


Gambar 8.2. Module I2C LCD

### B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. LCD 16 x 2 I2C                    | 1 Buah |
| 3. Kabel <i>Jumper Female-Female</i> | 2 Buah |

### C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 8.3. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	HEADER MCU
LCD 16x2 I2C - SDA	PIN D2 - SDA
LCD 16x2 I2C - SDL	PIN D1 - SCL

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Sakelar 1	LCD 16x2	-	-

#### D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C ke posisi ON dan *Backlight* LCD ke posisi (BL).
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools > Board > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module)*.
7. Pilih *port* yang terhubung pada *Tools > Port > COM x*.
8. *Verify (compile)* dan *Upload* program.

#### Contoh Program:

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);
// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  lcd.init(); // Inisialisasi LCD
  lcd.backlight(); // Menyalakan Backlight LCD
  // Menampilkan Teks pada LCD
  lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");
  lcd.setCursor(0,1); lcd.print("Connecting-WiFi");

  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
```

```

    Serial.print(".");
    delay(300);
  }
  // Menampilkan status koneksi dan alamat IP
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Menampilkan versi client firebase
  Serial.printf("Firebase Client v%s\n\n",
               FIREBASE_CLIENT_VERSION);
  // Memulai koneksi dengan database
  // Re-koneksi jika WiFi terputus
  Firebase.begin(DATABASE_URL, API_KEY);
  Firebase.reconnectWiFi(true);

  delay(2000);      // Jeda tampilan teks pada LCD
  lcd.clear();     // Membersihkan tampilan LCD
}
void loop() {
  // Mengatur kursor pada kolom pertama baris pertama
  // Menampilkan teks
  lcd.setCursor(0, 0);
  lcd.print("Menampilkan Teks");

  // Membaca database pada alamat */user/switch1
  // Jika bernilai true menampilkan Data sakelar ON
  // Jika bernilai false menampilkan Data sakelar OFF
  if(Firebase.getString(fbdo, "/" + user + "/switch1")) {
    if(fbdo.to<bool>() == true){
      lcd.setCursor(0, 1);
      lcd.print("Data Sakelar ON ");
    }else{
      lcd.setCursor(0, 1);
      lcd.print("Data Sakelar OFF");
    }
  }
}
}

```

### Keterangan Program:

```

#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

```

Library ESP8266 WiFi dan Firebase ESP8266.

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

```

Library LCD 16x2 I2C.

```

#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

```

SSID dan Password WiFi.

```

#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

```

URL dan token firebase database.

```
FirebaseData fbdo;  
LiquidCrystal_I2C lcd(0x27,16,2);  
Firebase dan LCD objek.
```

```
String user = "user_id";  
Variabel user. Gunakan sesuai dengan user ID yang didaftarkan pada aplikasi IoT KIT.
```

```
Serial.begin(115200);  
Serial monitor pada baudrate 115200.
```

```
lcd.init();  
lcd.backlight();  
Inisialisasi dan menyalakan backlight LCD.
```

```
lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");  
lcd.setCursor(0,1); lcd.print("Connecting-WiFi");  
Menampilkan teks pada LCD.
```

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
...  
Serial.println(WiFi.localIP());  
Serial.println();  
Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.
```

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);  
Menampilkan versi klien firebase, memulai koneksi dengan firebase database dan mengaktifkan koneksi ulang jika WiFi terputus.
```

```
delay(2000);  
lcd.clear();  
Jeda tampilan teks selama 2000ms atau 2 detik kemudian tampilan LCD dibersihkan.
```

```
lcd.setCursor(0, 0);  
lcd.print("Menampilkan Teks");  
Mencetak teks pada kolom pertama baris pertama.
```

```
if(Firebase.getString(fbdo, "/" + user + "/switch1")) {  
...  
}  
Membaca database pada alamat /user/switch1, LCD menampilkan teks pada baris kedua "Data Sakelar ON " jika bernilai true dan "Data Sakelar OFF" jika bernilai false.
```

## MODUL 9 | ANTARMUKA DENGAN OLED DISPLAY 128 x 64

Pada modul ini akan membahas mengenai antarmuka dengan *display* OLED 128 x 64 dengan proyek menampilkan teks pada *display* OLED melalui aplikasi IoT KIT.

### A. Pengenalan OLED Display

*Organic Light Emitting Diode* (OLED) adalah merupakan sebuah semikonduktor sebagai pemancar cahaya yang terbuat dari lapisan organik. OLED digunakan dalam teknologi elektroluminensi, seperti pada tampilan layar atau *display*. Teknologi ini terkenal fleksibel dengan ketipisannya yang mencapai kurang dari 1 mm.

OLED merupakan peranti penting dalam teknologi elektroluminensi. Teknologi tersebut memiliki dasar konsep pancaran cahaya yang dihasilkan oleh piranti akibat adanya medan listrik yang diberikan. Teknologi OLED dikembangkan untuk memperoleh tampilan yang luas, fleksibel, murah dan dapat digunakan sebagai layar yang efisien untuk berbagai keperluan layar tampilan atau *display*.

Pada proyek mikrokontroler jenis OLED yang banyak digunakan adalah OLED SSD1306 atau OLED *Display* 128 x 64. Untuk OLED ini memiliki beberapa warna yang beredar di pasaran, yaitu warna dengan tulisan putih, biru, dan ada juga yang campuran kuning dan biru. Selain jenis warna juga terdapat dua jenis komunikasi yaitu I2C dan SPI. Pada trainer mikrokontroler ini menggunakan OLED *Display* 128 x 64 dengan komunikasi I2C.

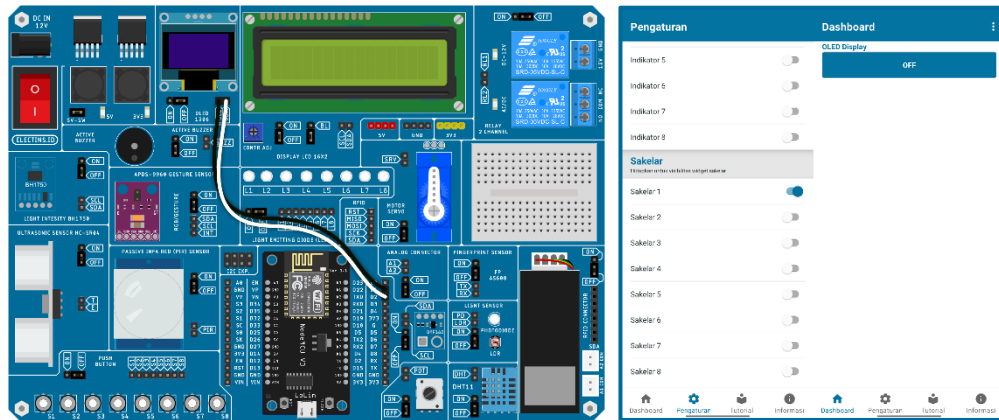


Gambar 9.1. OLED Display 128 x 64 I2C

### B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. OLED <i>Display</i> 128x4         | 1 Buah |
| 3. Kabel <i>Jumper Female-Female</i> | 2 Buah |

### C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 9.2. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	HEADER MCU
OLED Display - SDA	PIN D2 - SDA
OLED Display - SCL	PIN D1 - SCL

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Sakelar 1	OLED Display	-	-

### D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* OLED Display ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* OLED Display ke posisi ON.
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools* > *Board* > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module).
7. Pilih *port* yang terhubung pada *Tools* > *Port* > COM x.
8. *Verify (compile)* dan *Upload* program.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

// Library OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Mengatur OLED Display dengan resolusi 128x64
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
                          OLED_RESET);

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Memulai komunikasi dengan OLED Display
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  // Menampilkan status koneksi dan alamat IP
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Menampilkan versi client firebase
  Serial.printf("Firebase Client v%s\n\n",
                FIREBASE_CLIENT_VERSION);
  // Memulai koneksi dengan database
  // Re-koneksi jika WiFi terputus
```



```

Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);

// Membersihkan tampilan display
display.clearDisplay();
}

void loop() {
display.setTextSize(1);           // Mengatur ukuran teks = 1
display.setTextColor(WHITE);     // Mengatur teks warna putih
display.setCursor(1,1);          // Mengatur kursor (x:1, y:1)
display.print("Trainer IoT");    // Mencetak Teks "Trainer IoT"

// Membaca database pada alamat */user/switch1
// Jika bernilai true menampilkan Data ON
// Jika bernilai false menampilkan Data OFF
if(Firebase.getString(fbdo, "/" + user + "/switch1")) {
  if(fbdo.to<bool>() == true){
    display.setTextSize(2);      // Mengatur ukuran teks = 2
    display.setTextColor(WHITE); // Mengatur teks warna putih
    display.setCursor(1,25);     // Mengatur kursor (x:1, y:25)
    display.print("Data ON");    // Mencetak Teks "Data ON";
  }else{
    display.setTextSize(2);      // Mengatur ukuran teks = 2
    display.setTextColor(WHITE); // Mengatur teks warna putih
    display.setCursor(1,25);     // Mengatur kursor (x:1, y:25)
    display.print("Data OFF");   // Mencetak Teks "Data OFF";
  }
}
display.display();              // Menampilkan data
display.clearDisplay();         // Membersihkan tampilan display
}

```

### Keterangan Program:

```

#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

```

*Library* ESP8266 WiFi dan Firebase ESP8266.

```

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

```

*Library* OLED Display.

```

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

```

Mengatur OLED Display dengan resolusi 128x64.

```

#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

```

SSID dan Password WiFi.



```
#define DATABASE_URL "project_id.firebaseio.com"  
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
FirebaseData fbdo;
```

Firestore objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
```

Memulai komunikasi dengan OLED *Display*.

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
Serial.print("Connecting to Wi-Fi");
```

```
...
```

```
Serial.println(WiFi.localIP());
```

```
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
```

```
Firestore.begin(DATABASE_URL, API_KEY);
```

```
Firestore.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
display.clearDisplay();
```

Membersihkan tampilan *display*.

```
display.setTextSize(1);
```

```
...
```

```
display.print("Trainer IoT");
```

Mencetak teks "Trainer IoT".

```
if(Firebase.getString(fbdo, "/" + user + "/switch1")) {
```

```
...
```

```
}
```

Membaca *database* pada alamat `/user/switch1`, OLED menampilkan teks "Data ON" jika bernilai *true* dan "Data OFF" jika bernilai *false*.

```
display.display();
```

```
display.clearDisplay();
```

Menampilkan data pada *display* kemudian dibersihkan agar tidak tumpang tindih.

## MODUL 10 | ANTARMUKA DENGAN SENSOR CAHAYA ANALOG

Pada modul ini akan membahas mengenai antarmuka dengan sensor cahaya, proyek IoT *light meter* dengan sensor cahaya analog.

### A. Pengenalan Sensor Cahaya Analog

#### 1. Light Dependent Resistor (LDR)

*Light Dependent Resistor* atau LDR adalah jenis resistor yang nilai hambatannya atau nilai resistansinya tergantung pada intensitas cahaya yang diterimanya. Nilai hambatannya akan menurun pada saat cahaya terang dan nilai hambatannya akan menjadi tinggi jika dalam kondisi gelap. Dengan kata lain, fungsi LDR (*Light Dependent Resistor*) adalah untuk menghantarkan arus listrik jika menerima sejumlah intensitas cahaya (Kondisi Terang) dan menghambat arus listrik dalam kondisi gelap.

LDR merupakan komponen elektronika peka cahaya ini sering digunakan atau diaplikasikan dalam rangkaian elektronika sebagai sensor pada lampu penerang jalan, lampu kamar tidur, rangkaian anti maling, *shutter* kamera, alarm dan lain sebagainya.



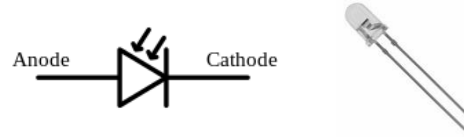
Gambar 10.1. Simbol dan bentuk LDR

Naik turunnya nilai hambatannya akan sebanding dengan jumlah cahaya yang diterimanya. Pada umumnya, nilai hambatannya LDR akan mencapai 200 Kilo Ohm ( $k\Omega$ ) pada kondisi gelap dan menurun menjadi 500 Ohm ( $\Omega$ ) pada kondisi cahaya terang.

#### 2. Photo diode

Photodiode adalah komponen elektronika yang dapat mengubah cahaya menjadi arus listrik. Photodiode merupakan komponen aktif yang terbuat dari bahan semikonduktor dan tergolong dalam keluarga Dioda. Seperti Dioda pada umumnya, Photodiode memiliki dua kaki terminal yaitu kaki terminal *Katoda* dan kaki terminal *Anoda*, namun photodiode memiliki Lensa dan Filter Optik yang terpasang di permukaannya sebagai pendeteksi cahaya. Cahaya yang dapat dideteksi oleh photodiode di antaranya seperti cahaya matahari, cahaya tampak, sinar inframerah, sinar ultra-violet (UV) hingga sinar X. Oleh karena itu, photodiode banyak diaplikasikan ke berbagai perangkat Elektronika dan listrik

seperti sensor penghitung, sensor cahaya kamera, alat-alat medis, *scanner barcode* dan peralatan keamanan.

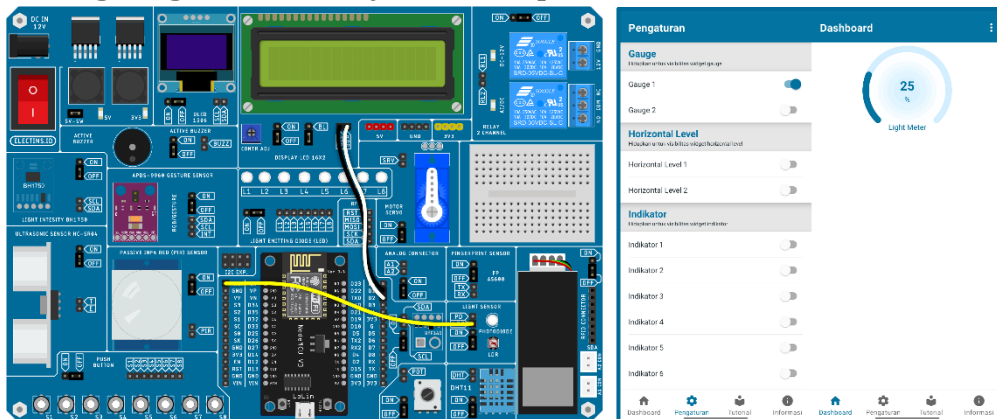


Gambar 10.2. Simbol dan bentuk photodiode

### B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. Sensor Cahaya LDR/Photodiode      | 1 Buah |
| 3. LCD 16x2 I2C                      | 1 Buah |
| 4. Kabel <i>Jumper Female-Female</i> | 3 Buah |

### C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 10.3. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	HEADER MCU
LCD 16x2 I2C - SDA	PIN D2 - SDA
LCD 16x2 I2C - SDL	PIN D1 - SCL
PD/LDR	PIN A0

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Gauge 1	Light Meter	100	%

#### D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C dan Light Sensor PD/LDR ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C, Light Sensor ke posisi ON dan *Backlight* LCD ke posisi (BL).
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools* > *Board* > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module).
7. Pilih *port* yang terhubung pada *Tools* > *Port* > *COM x*.
8. *Verify (compile)* dan *Upload* program.

#### Contoh Program:

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// PIN Sensor Cahaya LDR/PD terhubung ke PIN A0 NodeMCU
#define LS A0

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);

// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

// Variabel untuk menyimpan data ADC dan data konversi sensor
int ls_adc, ls_value;
// Variabel untuk menyimpan karakter data sensor
char ls_data[3];

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  lcd.init(); // Inisialisasi LCD
  lcd.backlight(); // Menyalakan Backlight LCD
```

```
// Menampilkan Teks pada LCD
lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");
lcd.setCursor(0,1); lcd.print("Connecting-WiFi");

// Memulai koneksi WiFi
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED)
{
    Serial.print(".");
    delay(300);
}
// Menampilkan status koneksi dan alamat IP
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();

// Menampilkan versi client firebase
Serial.printf("Firebase Client v%s\n\n",
              FIREBASE_CLIENT_VERSION);
// Memulai koneksi dengan dengan database
// Re-koneksi jika WiFi terputus
Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);

delay(2000); // Jeda tampilan teks pada LCD
lcd.clear(); // Membersihkan tampilan LCD
}

void loop() {
    // Membaca data ADC sensor
    // ADC disimpan pada 'ls_adc'
    // Data konversi disimpan pada 'ls_value'
    ls_adc = analogRead(LS);
    ls_value = map(ls_adc, 0, 1024, 0, 100);

    // Mencetak data pada serial monitor
    Serial.print("Cahaya : " + String(ls_value) + " %");
    Serial.println();

    // Mengirim data sensor ke database
    // dengan alamat */user/gaugel
    Firebase.setInt(fbdo, "/" + user + "/gaugel", ls_value);

    // Mencetak data sensor pada LCD 16x2 I2C
    lcd.setCursor(0,0);
    lcd.print("Light Sensor (%");
    lcd.setCursor(0,1);
    sprintf(ls_data, "LS: %3d", ls_value);
    lcd.print(ls_data);

    // Jeda 150 ms agar perubahan nilai lebih halus
    delay(150);
}
```

**Keterangan Program:**

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
```

Library ESP8266 WiFi dan Firebase ESP8266.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Library LCD 16x2 I2C.

```
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan *Password* WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
#define LS A0
```

Pin Light Sensor (LS) terhubung ke Pin A0 NodeMCU

```
FirebaseData fbdo;
LiquidCrystal_I2C lcd(0x27,16,2);
```

Firestore dan LCD objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
int ls_adc, ls_value;
char ls_data[3];
```

Variabel 'ls\_adc' dan 'ls\_value' untuk menyimpan data ADC dan data hasil konversi sensor. Variabel 'ls\_data' untuk menyimpan data sensor dalam bentuk karakter.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
lcd.init();
lcd.backlight();
```

Inisialisasi dan menyalakan *backlight* LCD.

```
lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");
lcd.setCursor(0,1); lcd.print("Connecting-WiFi");
```

Menampilkan teks pada LCD

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
...
Serial.println(WiFi.localIP());
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
delay(2000);  
lcd.clear();
```

Jeda tampilan teks selama 2000ms atau 2 detik kemudian tampilan LCD dibersihkan.

```
ls_adc = analogRead(LS);  
ls_value = map(ls_adc, 0, 1024, 0, 100);
```

Membaca data ADC sensor disimpan pada variabel 'ls\_adc' kemudian dikonversi menjadi nilai persentase (0-100 %) yang disimpan pada variabel 'ls\_value'.

```
Serial.print("Cahaya : " + String(ls_value) + " %");  
Serial.println();
```

Mencetak data pada serial monitor.

```
Firebase.setInt(fbdo, "/" + user + "/gauge1", ls_value);
```

Mengirim data sensor (tipe data integer) ke *database* dengan perintah `Firebase.setInt` dengan alamat `*/user/gauge1`.

```
lcd.setCursor(0,0);  
lcd.print("Light Sensor (%");  
lcd.setCursor(0,1);  
sprintf(ls_data, "LS: %3d", ls_value);  
lcd.print(ls_data);
```

Mencetak teks "Light Sensor (%)" pada baris pertama dan mencetak nilai sensor cahaya pada baris kedua.

## MODUL 11 | ANTARMUKA DENGAN SENSOR CAHAYA BH1750

Pada modul ini akan membahas mengenai antarmuka dengan sensor cahaya BH1750, proyek IoT *light meter* dengan sensor cahaya BH1750.

### A. Pengenalan Sensor Cahaya BH1750

BH1750 adalah sebuah IC sensor yang digunakan untuk mengukur perubahan intensitas cahaya dalam satuan lux. Sensor ini menggunakan protokol I2C untuk komunikasi dengan mikrokontroler. Jangkauan deteksi sensor ini cukup luas yaitu antara 1-65535 lux. 1 lux berarti 1 lumensi intensitas cahaya pada luas 1 meter persegi.



Gambar 11.1. Sensor Cahaya BH1750

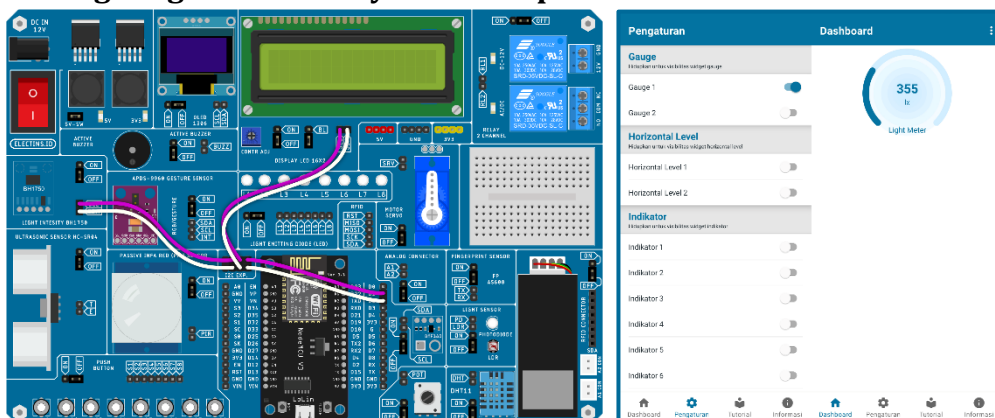
#### Spesifikasi Sensor BH1750

- Antarmuka I2C
- Resolusi 1-65536 lx
- Tegangan kerja *module* 3.3-5V (IC Sensor 2.4-3.6V)
- Konsumsi arus rendah 0.12 mA
- Redaman *noise* cahaya 50/60 Hz
- Suhu operasi -40 sampai 85 °C

### B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. Sensor Cahaya BH1750              | 1 Buah |
| 3. LCD 16x2 I2C                      | 1 Buah |
| 4. I2C <i>Expanded</i>               | 1 Buah |
| 5. Kabel <i>Jumper Female-Female</i> | 6 Buah |

### C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 11.2. Wiring Diagram dan Penyesuaian Aplikasi



Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

**Rangkaian:**

HEADER I/O	I2C EXP	HEADER MCU
LCD 16x2 I2C - SDA	I2C Exp - SDA	PIN D2 - SDA
LCD 16x2 I2C - SCL	I2C Exp - SCL	PIN D1 - SCL
BH1750 - SDA		
BH1750 - SCL		

**Pengaturan Aplikasi:**

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Gauge 1	Light Meter	1200	lx

**D. Petunjuk Praktikum:**

1. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C dan Light Sensor BH1750 ke *I2C Expanded* dan *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C, Light Sensor BH1750 ke posisi ON dan *Backlight* LCD ke posisi (BL).
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools > Board > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module)*.
7. Pilih *port* yang terhubung pada *Tools > Port > COM x*.
8. *Verify (compile)* dan Upload program.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Library Sensor BH 1750
#include <BH1750.h>

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"
// Firebase objek dengan nama fbdo
FirebaseData fbdo;
```

```
// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);
// BH1750 Objek dengan nama lightMeter
BH1750 lightMeter;

// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

// Variabel untuk menyimpan data pembacaan sensor
unsigned int lux;
// Variabel untuk menyimpan karakter data sensor
char light[16];

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  Wire.begin();          // Memulai koneksi I2C
  lightMeter.begin();    // Memulai komunikasi sensor BH1750

  lcd.init();           // Inisialisasi LCD
  lcd.backlight();     // Menyalakan Backlight LCD

  // Menampilkan Teks pada LCD
  lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");
  lcd.setCursor(0,1); lcd.print("Connecting-WiFi");

  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  // Menampilkan status koneksi dan alamat IP
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Menampilkan versi client firebase
  Serial.printf("Firebase Client v%s\n\n",
    FIREBASE_CLIENT_VERSION);
  // Memulai koneksi dengan dengan database
  // Re-koneksi jika WiFi terputus
  Firebase.begin(DATABASE_URL, API_KEY);
  Firebase.reconnectWiFi(true);

  delay(2000);          // Jeda tampilan teks pada LCD
  lcd.clear();         // Membersihkan tampilan LCD
}
```

```
void loop() {
  // Melakukan pembacaan pada sensor
  // Data pembacaan di simpan pada 'lux'
  lux = lightMeter.readLightLevel();

  // Mencetak data sensor pada LCD 16x2 I2C
  sprintf(light, "LIGHT:%5d lx", lux);
  lcd.setCursor(0,0); lcd.print("Light Meter (lx)");
  lcd.setCursor(0,1); lcd.print(light);

  // Mencetak data sensor pada serial monitor
  Serial.print("Cahaya : " + String(lux) + " %");
  Serial.println();

  // Mengirim data sensor ke database
  // dengan alamat *user/gaugel
  Firebase.setInt(fbdo, "/" + user + "/gaugel", lux);

  // Jeda 150 ms agar perubahan nilai lebih halus
  delay(150);
}
```

**Keterangan Program:**

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
```

*Library* ESP8266 WiFi dan Firebase ESP8266.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

*Library* LCD 16x2 I2C.

```
#include <BH1750.h>
```

*Library* sensor cahaya BH1750.

```
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan *Password* WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
FirebaseData fbdo;
LiquidCrystal_I2C lcd(0x27,16,2);
BH1750 lightMeter;
```

Firestore, LCD, BH1750 objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
unsigned int lux;
char light[16];
```

Variabel 'lux' untuk menyimpan data pembacaan sensor. Variabel 'light' untuk menyimpan data sensor dalam bentuk karakter.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
Wire.begin();  
lightMeter.begin();
```

Memulai koneksi dan komunikasi I2C dengan sensor BH1750.

```
lcd.init();  
lcd.backlight();
```

Inisialisasi dan menyalakan *backlight* LCD.

```
lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");  
lcd.setCursor(0,1); lcd.print("Connecting-WiFi");
```

Menampilkan teks pada LCD.

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
delay(2000);  
lcd.clear();
```

Jeda tampilan teks selama 2000ms atau 2 detik kemudian tampilan LCD dibersihkan.

```
lux = lightMeter.readLightLevel();
```

Melakukan pembacaan pada sensor, data disimpan pada variabel 'lux'.

```
sprintf(light, "LIGHT:%5d lx", lux);  
lcd.setCursor(0,0); lcd.print("Light Meter (lx)");  
lcd.setCursor(0,1); lcd.print(light);
```

Mencetak data sensor pada LCD 16x2 I2C.

```
Mencetak data sensor pada serial monitor  
Serial.print("Cahaya : " + String(lux) + " %");  
Serial.println();
```

Mencetak data pada serial monitor.

```
Firebase.setInt(fbdo, "/" + user + "/gauge1", lux);
```

Mengirim data sensor (tipe data integer) ke *database* dengan perintah `Firebase.setInt` dengan alamat `*/user/gauge1`.

## MODUL 12 | ANTARMUKA DENGAN POTENSIOMETER

Pada modul ini akan membahas mengenai antarmuka dengan potensiometer, proyek membaca data analog potensiometer pada aplikasi IoT KIT.

### A. Pengenalan Potensiometer

Potensiometer adalah salah satu jenis resistor yang nilai resistansinya dapat diatur sesuai dengan kebutuhan rangkaian elektronika ataupun kebutuhan pemakainya. Potensiometer merupakan keluarga resistor yang tergolong dalam kategori *Variable Resistor* (VR). Secara struktur, Potensiometer terdiri dari 3 kaki Terminal dengan sebuah *shaft* atau tuas yang berfungsi sebagai pengaturnya.



Gambar 12.1. Simbol dan bentuk potensiometer

Potensiometer terdiri dari sebuah elemen resistif yang membentuk jalur (*track*) dengan terminal di kedua ujungnya. Sedangkan terminal lainnya (biasanya berada di tengah) adalah Penyapu (*Wiper*) yang dipergunakan untuk menentukan pergerakan pada jalur elemen resistif (*Resistive*). Pergerakan Penyapu (*Wiper*) pada jalur elemen resistif inilah yang mengatur naik-turunnya Nilai Resistansi sebuah Potensiometer.

Potensiometer berfungsi sebagai pengatur Volume di peralatan Audio/ Video seperti radio, *walkie talkie*, *tape mobil*, *DVD player* dan *amplifier*. Potensiometer juga sering digunakan dalam rangkaian pengatur terang gelapnya lampu (*Light Dimmer Circuit*) dan pengatur tegangan pada *power supply*.

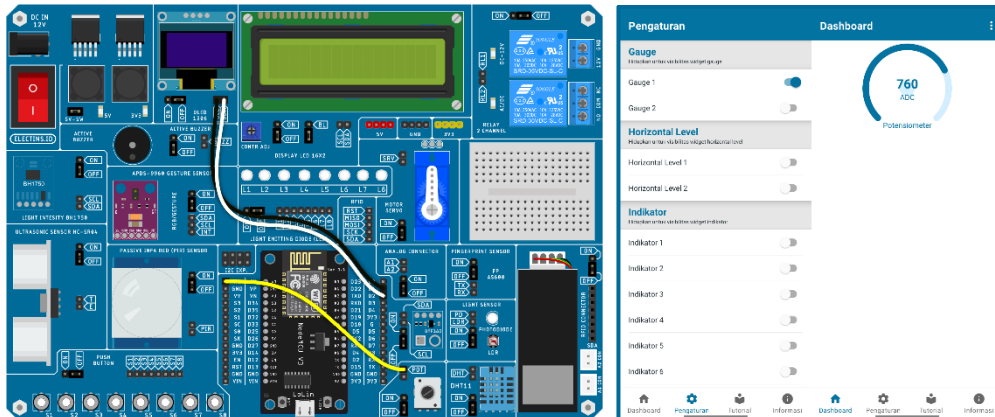
Berdasarkan bentuknya, Potensiometer dapat dibagi menjadi 3 macam, yaitu:

1. **Potensiometer Slider**, yaitu Potensiometer yang nilai resistansinya dapat diatur dengan cara menggeserkan *Wiper*-nya dari kiri ke kanan atau dari bawah ke atas sesuai dengan pemasangannya.
2. **Potensiometer Rotary**, yaitu Potensiometer yang nilai resistansinya dapat diatur dengan cara memutar *Wiper*-nya sepanjang lintasan yang melingkar. Pada trainer menggunakan potensiometer jenis ini.
3. **Potensiometer Trimmer**, yaitu Potensiometer yang bentuknya kecil dan harus menggunakan alat khusus seperti Obeng (*screwdriver*) untuk memutarnya. Potensiometer *Trimmer* ini biasanya dipasangkan di PCB seperti pengatur kontras pada *display LCD*.

## B. Perangkat yang Dibutuhkan

- |                               |        |
|-------------------------------|--------|
| 1. NodeMCU                    | 1 Buah |
| 2. Potensiometer              | 1 Buah |
| 3. OLED Display               | 1 Buah |
| 4. Kabel Jumper Female-Female | 3 Buah |

## C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 12.2. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

### Rangkaian:

HEADER I/O	HEADER MCU
OLED Display - SDA	PIN D2 - SDA
OLED Display - SCL	PIN D1 - SCL
Potensiometer - POT	PIN A0

### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Gauge 1	Potensiometer	1024	ADC

## D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* OLED Display dan Potensiometer ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* OLED Display, Potensiometer ke posisi ON.
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools > Board > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module)*.

7. Pilih *port* yang terhubung pada *Tools > Port > COM x*.
8. *Verify (compile)* dan *Upload* program.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Mengatur OLED Display dengan resolusi 128x64
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
                        OLED_RESET);

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// PIN Potensiometer terhubung ke PIN A0 NodeMCU
#define POT_PIN A0

// Firebase objek dengan nama fbdo
FirebaseData fbdo;

// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

// Variabel untuk menyimpan data ADC potensiometer
int pot_adc;
// Variabel untuk menyimpan karakter data potensiometer
char pot_data[4];

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Memulai komunikasi dengan OLED Display
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
}
```

```

}
// Menampilkan status koneksi dan alamat IP
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();

// Menampilkan versi client firebase
Serial.printf("Firebase Client v%s\n\n",
             FIREBASE_CLIENT_VERSION);
// Memulai koneksi dengan database
// Re-koneksi jika WiFi terputus
Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);

// Membersihkan tampilan display
display.clearDisplay();
}

void loop() {
// Membaca data ADC potensiometer
pot_adc = analogRead(POT_PIN);
sprintf(pot_data, "POT: %4d", pot_adc);

// Mengirim data sensor ke database
// dengan alamat *user/gaugel
Firebase.set(fbdo, "/" + user + "/gaugel", pot_adc);

// Mencetak Teks "ADC Potensiometer" pada OLED
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(1,1);
display.print("ADC Potensiometer");

// Mencetak data potensiometer pada OLED
display.setTextSize(2);
display.setTextColor(WHITE);
display.setCursor(1,25);
display.print(pot_data);

display.display(); // Menampilkan data
display.clearDisplay(); // Membersihkan tampilan display
}

```

### Keterangan Program:

```

#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

```

*Library* ESP8266 WiFi dan Firebase ESP8266.

```

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

```

*Library* OLED Display

```

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

```



```
#define OLED_RESET    -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
                        OLED_RESET);
```

Mengatur OLED *Display* dengan resolusi 128x64.

```
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan *Password* WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
#define POT_PIN A0
```

Pin potensiometer (POT) terhubung ke Pin A0 NodeMCU.

```
FirebaseData fbdo;
```

Firestore objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
int pot_adc;
char pot_data[3];
```

Variabel *pot\_adc* untuk menyimpan data ADC sensor. Variabel *pot\_data* untuk menyimpan data sensor dalam bentuk karakter.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
```

Memulai komunikasi dengan OLED *Display*.

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
Serial.print("Connecting to Wi-Fi");
```

```
...
```

```
Serial.println(WiFi.localIP());
```

```
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
```

```
Firestore.begin(DATABASE_URL, API_KEY);
```

```
Firestore.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
display.clearDisplay();
```

Membersihkan tampilan *display*.

```
pot_adc = analogRead(POT_PIN);
sprintf(pot_data, "POT: %4d", pot_adc);
```

Membaca data ADC potensiometer, disimpan pada variabel 'pot\_adc'. Data dikonversi menjadi karakter disimpan pada variabel 'pot\_data'.

```
Firestore.setInt(fbdo, "/" + user + "/gauge1", pot_adc);
```

Mengirim data ADC potensiometer (tipe data integer) ke *database* dengan perintah `Firestore.setInt` dengan alamat `*/user/gauge1`.

```
display.setTextSize(1);
```

```
...  
display.print("ADC Potensiometer");  
Mencetak teks "ADC Potensiometer".
```

```
display.setTextSize(2);
```

```
...  
display.print(pot_data);
```

Mencetak data ADC potensiometer.

```
display.display();  
display.clearDisplay();
```

Menampilkan data pada *display* kemudian dibersihkan agar tidak tumpang tindih.

## MODUL 13 | ANTARMUKA DENGAN SENSOR PIR

Pada modul ini akan membahas mengenai antarmuka dengan sensor PIR dengan proyek deteksi gerakan dengan sensor PIR berbasis IoT.

### A. Pengenalan Sensor PIR

Sensor PIR (*Passive Infra Red*) adalah sensor yang digunakan untuk mendeteksi adanya pancaran sinar inframerah. Sensor PIR bersifat pasif, artinya sensor ini tidak memancarkan sinar inframerah tetapi hanya menerima radiasi sinar inframerah dari luar. Sensor PIR biasanya digunakan dalam perancangan detektor gerakan berbasis PIR. Karena semua benda memancarkan energi radiasi, sebuah gerakan akan terdeteksi ketika sumber inframerah dengan suhu tertentu (misal: manusia) melewati sumber inframerah yang lain dengan suhu yang berbeda (misal: dinding), maka sensor akan membandingkan pancaran inframerah yang diterima setiap satuan waktu, sehingga jika ada pergerakan maka akan terjadi perubahan pembacaan pada sensor



Gambar 13.1. Sensor PIR

Sensor PIR bekerja dengan cara pancaran inframerah masuk melalui lensa Fresnel dan mengenai sensor *pyroelektrik*, karena sinar inframerah mengandung energi panas maka sensor *pyroelektrik* akan menghasilkan arus listrik. Kemudian menimbulkan tegangan dan dibaca secara analog oleh sensor. Kemudian sinyal ini akan dikuatkan oleh penguat dan dibandingkan oleh komparator dengan tegangan referensi tertentu (keluaran berupa sinyal 1-bit). Jadi sensor PIR hanya akan mengeluarkan logika 0 dan 1. 0 saat sensor tidak mendeteksi adanya pancaran inframerah dan 1 saat sensor mendeteksi inframerah.

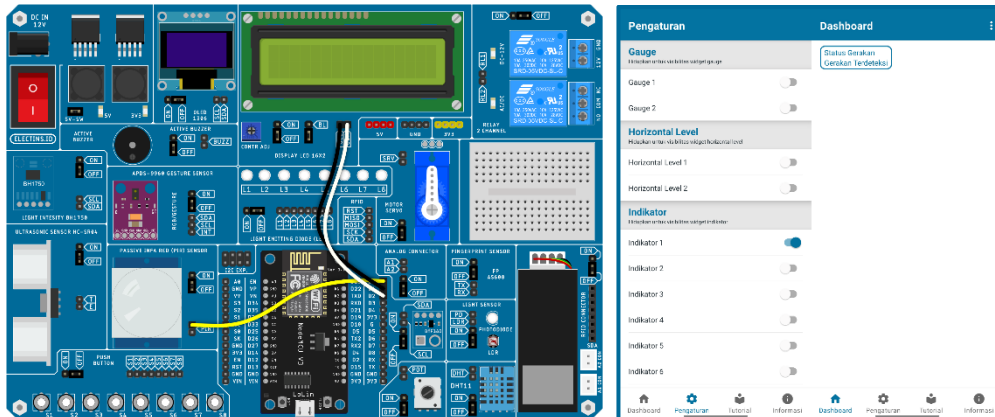
Spesifikasi:

Tegangan kerja	4.7 – 5 Volt
Arus tanpa beban	300 $\mu$ A
Suhu kerja	-20 -50 °C
Jangkauan deteksi	5 meter
Kecepatan respons	0.5 detik

## B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. Sensor PIR                        | 1 Buah |
| 3. LCD 16x2 I2C                      | 1 Buah |
| 4. Kabel <i>Jumper Female-Female</i> | 3 Buah |

## C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 13.2. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

### Rangkaian:

HEADER I/O	HEADER MCU
LCD 16x2 I2C - SDA	PIN D2 - SDA
LCD 16x2 I2C - SCL	PIN D1 - SCL
Sensor PIR - PIR	PIN D6

### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Indikator 1	Status Gerakan	-	-

## D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C, Buzzer dan Sensor PIR ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C, Buzzer dan Sensor PIR ke posisi ON dan *Backlight* LCD ke posisi (BL).
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.

6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools > Board > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module)*.
7. Pilih *port* yang terhubung pada *Tools > Port > COM x*.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// PIN Sensor PIR terhubung ke PIN D0 NodeMCU
#define PIR_PIN D0

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);

// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);
  pinMode(PIR_PIN, INPUT); // Pin PIR sebagai Input

  lcd.init(); // Inisialisasi LCD
  lcd.backlight(); // Menyalakan Backlight LCD
  // Menampilkan Teks pada LCD
  lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");
  lcd.setCursor(0,1); lcd.print("Connecting-WiFi");

  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  // Menampilkan status koneksi dan alamat IP
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
}
```

```
// Menampilkan versi client firebase
Serial.printf("Firebase Client v%s\n\n",
              FIREBASE_CLIENT_VERSION);
// Memulai koneksi dengan database
// Re-koneksi jika WiFi terputus
Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);

delay(2000);      // Jeda tampilan teks pada LCD
lcd.clear();      // Membersihkan tampilan LCD
}

void loop() {
  // Mengatur kursor pada kolom pertama baris pertama
  // Menampilkan teks
  lcd.setCursor(0, 0);
  lcd.print("Deteksi Gerakan:");

  // Jika PIR mendeteksi gerakan output bernilai HIGH
  // Mencetak teks jika gerakan terdeteksi dan tidak terdeteksi
  // Mengirim data gerakan ke database *user/indicator1
  if(digitalRead(PIR_PIN)==HIGH){
    lcd.setCursor(0,1); lcd.print("Gerak Terdeteksi");
    Firebase.setString(fbdo, "/" + user + "/indicator1", "Gerakan
                        Terdeteksi");
  }else{
    lcd.setCursor(0,1); lcd.print("Tidak Terdeteksi");
    Firebase.setString(fbdo, "/" + user + "/indicator1", "Tidak
                        Terdeteksi");
  }
}
}
```

### Keterangan Program:

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
```

*Library* ESP8266 WiFi dan Firebase ESP8266.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

*Library* LCD 16x2 I2C.

```
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan *Password* WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
#define PIR_PIN D0
```

Pin sensor PIR terhubung ke Pin D0 NodeMCU.

```
FirebaseData fbdo;
LiquidCrystal_I2C lcd(0x27,16,2);
```

Firestore dan LCD objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
lcd.init();  
lcd.backlight();
```

Inisialisasi dan menyalakan *backlight* LCD.

```
lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");  
lcd.setCursor(0,1); lcd.print("Connecting-WiFi");
```

Menampilkan teks pada LCD

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
delay(2000);  
lcd.clear();
```

Jeda tampilan teks selama 2000ms atau 2 detik kemudian tampilan LCD dibersihkan.

```
lcd.setCursor(0, 0);  
lcd.print("Deteksi Gerakan:");
```

Menampilkan teks "Deteksi Gerakan:" pada baris pertama.

```
if(digitalRead(PIR_PIN)==HIGH){  
...  
}else{  
...  
}
```

Jika sensor PIR mendeteksi gerakan maka output sensor bernilai HIGH dan mencetak dan mengirim data teks (*String*) ke *database* "Gerak Terdeteksi", sebaliknya jika tidak terdeteksi mencetak dan mengirim data teks (*String*) ke *database* "Tidak Terdeteksi" dengan perintah `Firebase.setString`.

## MODUL 14 | ANTARMUKA DENGAN SENSOR ULTRASONIK

Pada modul ini akan membahas mengenai antarmuka dengan sensor ultrasonik, proyek *range meter* dengan sensor ultrasonik berbasis IoT.

### A. Pengenalan Sensor Ultrasonik

Sensor ultrasonik adalah sebuah sensor yang berfungsi untuk mengubah besaran fisis berupa bunyi menjadi besaran listrik dan sebaliknya. Sensor ini bekerja berdasarkan prinsip dari pantulan suatu gelombang suara, di mana sensor ini menghasilkan gelombang suara yang kemudian menangkap kembali dengan perbedaan waktu sebagai dasar pengindra. Perbedaan waktu yang dipancarkan dan diterima kembali adalah berbanding lurus dengan jarak objek yang memantulkannya.



Gambar 14.1. Sensor Ultrasonik HC-SR04

Sensor ultrasonik ini umumnya digunakan untuk mendeteksi keberadaan suatu objek dalam jarak tertentu di depannya. Sensor ultrasonik mempunyai kemampuan mendeteksi objek lebih jauh terutama untuk benda-benda yang keras. Pada benda-benda yang keras yaitu yang mempunyai permukaan kasar gelombang ini akan dipantulkan lebih kuat daripada benda yang permukaannya lunak. Sensor ultrasonik ini terdiri dari rangkaian pemancar ultrasonik yang disebut *transmitter* dan rangkaian penerima ultrasonik disebut *receiver*.

Spesifikasi Ultrasonik HC-SR04:

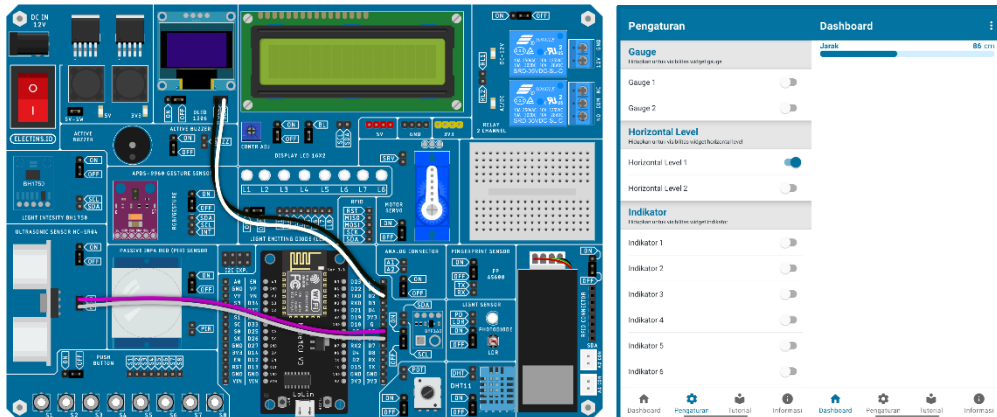
Tegangan kerja	45 Volt DC
Arus tanpa deteksi	<2 mA
Arus dengan deteksi	15 mA
Frekuensi Suara	40 kHz
Jangkauan minimum	2 cm
Jangkauan maksimum	400 cm
Input Trigger	10 $\mu$ S minimum, pulsa level TTL
Pulsa Echo	Sinyal level TTL positif, lebar berbanding proporsional dengan jarak yang dideteksi



## B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. Sensor ultrasonik HC-SR04         | 1 Buah |
| 3. OLED Display                      | 1 Buah |
| 4. Kabel <i>Jumper Female-Female</i> | 4 Buah |

## C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 14.2. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

### Rangkaian:

HEADER I/O	HEADER MCU
OLED <i>Display</i> - SDA	PIN D2 - SDA
OLED <i>Display</i> - SCL	PIN D1 - SCL
Ultrasonik - T	PIN D5
Ultrasonik - E	PIN D6

### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Horizontal Level 1	Jarak	200	cm

## D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C, Buzzer dan Sensor PIR ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C, Buzzer dan Sensor PIR ke posisi ON dan *Backlight* LCD ke posisi (BL).
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.

6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools > Board > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module)*.
7. Pilih *port* yang terhubung pada *Tools > Port > COM x*.
8. *Verify (compile)* dan Upload program.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
// Library Sensor Ultrasonic
#include <NewPing.h>

// Mengatur OLED Display dengan resolusi 128x64
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// Mengatur PIN dan jarak maksimal
#define T D5 // PIN T (Trig) terhubung ke PIN D5 NodeMCU
#define E D6 // PIN E (Echo) terhubung ke PIN D6 NodeMCU
#define Maks 200 // Jarak maksimal yang diukur
NewPing us(T, E, Maks);

// Firebase objek dengan nama fbdo
FirebaseData fbdo;

// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

// Variabel jarak menyimpan data sensor ultrasonic
int jarak;
// Variabel untuk menyimpan karakter data sensor
char val_jarak[3];

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Memulai komunikasi dengan OLED Display
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
```

```
// Memulai koneksi WiFi
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED)
{
    Serial.print(".");
    delay(300);
}
// Menampilkan status koneksi dan alamat IP
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();

// Menampilkan versi client firebase
Serial.printf("Firebase Client v%s\n\n",
             FIREBASE_CLIENT_VERSION);
// Memulai koneksi dengan dengan database
// Re-koneksi jika WiFi terputus
Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);

// Membersihkan tampilan display
display.clearDisplay();
}

void loop() {
    // Data pembacaan sensor disimpan pada 'jarak'
    jarak = us.ping_cm();
    sprintf(val_jarak, "%3d", jarak);

    // Mencetak data pada serial monitor
    Serial.println("Jarak:" + String(jarak) + " cm");

    // Mengirim data sensor ultrasonik ke database
    // dengan alamat *user/gaugel
    Firebase.set(fbdo, "/" + user + "/hlevel1", jarak);

    // Mencetak Teks "Jarak (cm)" pada OLED
    display.setTextSize(2);
    display.setTextColor(WHITE);
    display.setCursor(1,1);
    display.print("Jarak (cm)");

    // Mencetak data sensor pada OLED
    display.setTextSize(3);
    display.setTextColor(WHITE);
    display.setCursor(1,25);
    display.print(val_jarak);

    display.display(); // Menampilkan data
    display.clearDisplay(); // Membersihkan tampilan display
}
```

**Keterangan Program:**

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
```

Library ESP8266 WiFi dan Firebase ESP8266.

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

Library OLED Display.

```
#include <NewPing.h>
```

Library sensor ultrasonik.

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

Mengatur OLED Display dengan resolusi 128x64.

```
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan Password WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"
```

URL dan token firebase database.

```
// Mengatur PIN dan jarak maksimal
#define T D5 // PIN T (Trig) terhubung ke PIN D5 NodeMCU
#define E D6 // PIN E (Echo) terhubung ke PIN D6 NodeMCU
#define Maks 200 // Jarak maksimal yang diukur
NewPing us(T, E, Maks);
```

Mengatur Pin sensor ultrasonik dan jarak maksimal 200 cm pada objek library dengan nama us.

```
FirebaseData fbdo;
```

Firestore objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
int jarak;
char val_jarak[3];
```

Variabel 'jarak' untuk menyimpan data pembacaan sensor. Variabel 'val\_jarak' untuk menyimpan data sensor dalam bentuk karakter.

```
Serial.begin(115200);
```

Serial monitor pada baudrate 115200.

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
```

Memulai komunikasi dengan OLED Display.

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
display.clearDisplay();
```

Membersihkan tampilan *display*.

```
jarak = us.ping_cm();  
sprintf(val_jarak, "%3d", jarak);
```

Data pembacaan sensor disimpan pada variabel 'jarak'. Data dikonversi menjadi karakter disimpan pada variabel 'val\_jarak'.

```
Firebase.setInt(fbdo, "/" + user + "/hlevel1", jarak);
```

Mengirim data sensor ultrasonik (tipe data integer) ke *database* dengan perintah `Firebase.setInt` dengan alamat `*/user/hlevel1`.

```
display.setTextSize(2);  
...  
display.print("Jarak (cm)");  
Mencetak teks "Jarak (cm)".
```

```
display.setTextSize(3);  
...  
display.print(val_jarak);
```

Mencetak data sensor ultrasonik pada OLED.

```
display.display();  
display.clearDisplay();
```

Menampilkan data pada *display* kemudian dibersihkan agar tidak tumpang tindih.

## MODUL 15 | ANTARMUKA DENGAN SENSOR DHT11

Pada modul ini akan membahas mengenai antarmuka dengan sensor suhu dan kelembaban, proyek monitoring suhu dan kelembaban udara.

### A. Pengenalan Sensor DHT 11

DHT11 adalah sensor suhu dan kelembaban yang memiliki keluaran sinyal digital yang dikalibrasi dengan sensor suhu dan kelembaban yang kompleks. Teknologi ini memastikan keandalan tinggi dan sangat baik stabilitasnya dalam jangka panjang. Sensor ini termasuk elemen resistif dan perangkat pengukur suhu NTC memiliki kualitas yang sangat baik, respons cepat dan anti-gangguan.



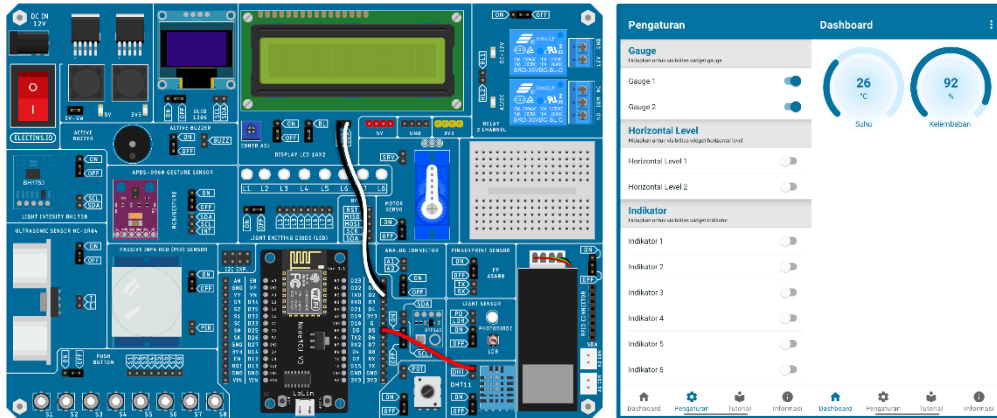
Gambar 15.1. Sensor DHT11

Prinsip kerja dari sensor ini adalah di dalam sensor ini terdapat sebuah *Thermistor* dengan tipe NTC (*Negative Temperature Coefficient*). Seperti kita tahu cara kerja dari *thermistor* adalah naik dan turunnya suhu berpengaruh terhadap naik dan turunnya resistansi *thermistor*. Pada sensor ini menggunakan *thermistor* NTC di mana nilai resistansinya berbanding terbalik dengan kenaikan suhu. Yaitu, semakin tinggi suhu di sekitar sensor maka nilai resistansi NTC akan semakin kecil. Sebaliknya nilai resistansinya akan meningkat ketika suhu ruangan sensor menurun. Berdasarkan naik turunnya resistansi tersebut maka sensor akan mengeluarkan output berupa nilai analog yang akan dibaca dan dikonversi oleh NodeMCU menjadi nilai suhu (dalam bentuk derajat C) dan kelembaban ruangan (dalam bentuk %).

### B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. Sensor DHT11                      | 1 Buah |
| 3. LCD 16x2 I2C                      | 1 Buah |
| 4. Kabel <i>Jumper Female-Female</i> | 3 Buah |

### C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 15.2. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	HEADER MCU
LCD 16x2 I2C - SDA	PIN D2 - SDA
LCD 16x2 I2C - SCL	PIN D1 - SCL
DHT11 - DHT	PIN D5

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Gauge 1	Suhu	80	°C
Gauge 2	Kelembaban	100	%

### D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C dan Sensor DHT11 ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C dan Sensor DHT11 ke posisi ON dan *Backlight* LCD ke posisi (BL).
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools* > *Board* > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module).
7. Pilih *port* yang terhubung pada *Tools* > *Port* > COM x.
8. *Verify (compile)* dan Upload program.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Library DHT11
#include <DHT.h>

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// PIN DHT Terhubung ke PIN D5 NodeMCU
#define DHT_PIN D5

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// LCD objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);
// DHT objek dengan nama dht
DHT dht(DHT_PIN, DHT11);

// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

// Variabel untuk menyimpan data suhu dan kelembaban
float t, h;

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);
  dht.begin();

  lcd.init(); // Inisialisasi LCD
  lcd.backlight(); // Menyalakan Backlight LCD

  // Menampilkan Teks pada LCD
  lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");
  lcd.setCursor(0,1); lcd.print("Connecting-WiFi");
  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  // Menampilkan status koneksi dan alamat IP
  Serial.println();
  Serial.print("Connected with IP: ");
```



```

Serial.println(WiFi.localIP());
Serial.println();

// Menampilkan versi client firebase
Serial.printf("Firebase Client v%s\n\n",
              FIREBASE_CLIENT_VERSION);
// Memulai koneksi dengan database
// Re-koneksi jika WiFi terputus
Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);

delay(2000);      // Jeda tampilan teks pada LCD
lcd.clear();      // Membersihkan tampilan LCD
}

void loop() {
  // Membaca suhu dan kelembaban udara
  // Data disimpan pada masing-masing variabel
  t = dht.readTemperature(); // Membaca suhu dalam Celsius
  h = dht.readHumidity();     // Membaca kelembaban Udara
  // Periksa pembacaan jika gagal akan diulangi
  if(isnan(t) || isnan(h)) {
    Serial.println("Tidak dapat membaca sensor DHT11!");
    return;
  }
  // Set nilai menjadi 0 jika data isnan
  if(isnan(t)) t=0;
  if(isnan(h)) h=0;

  // Mencetak data suhu dan kelembaban pada serial monitor
  Serial.println("Temp:" + String(t) + " °C");
  Serial.println("Humd:" + String(h) + " %");
  Serial.println();

  // Mengirim data suhu ke database *user/gauge1
  // Mengirim data kelembaban ke database *user/gauge2
  Firebase.setFloat(fbdo, "/" + user + "/gauge1", t);
  Firebase.setFloat(fbdo, "/" + user + "/gauge2", h);

  // Menampilkan data suhu pada LCD
  lcd.setCursor(0,0);
  lcd.print("Temp: " + String(t) + char(223) + "C");

  // Menampilkan data kelembaban pada LCD
  lcd.setCursor(0,1);
  lcd.print("Humd: " + String(h) + " %");
}

```

### Keterangan Program:

```

#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

```

*Library* ESP8266 WiFi dan Firebase ESP8266.

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

```

*Library* LCD 16x2 I2C.

```
#include <DHT.h>
```

Library DHT11.

```
#define WIFI_SSID "SSID_WIFI"  
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan *Password* WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"  
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
#define DHT_PIN D5
```

Pin DHT11 terhubung ke Pin D5 NodeMCU.

```
FirebaseData fbdo;  
LiquidCrystal_I2C lcd(0x27,16,2);  
DHT dht(DHT_PIN, DHT11);
```

Firestore, LCD dan DHT objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
float t, h;
```

Variabel untuk menyimpan data suhu dan kelembaban.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
lcd.init();  
lcd.backlight();
```

Inisialisasi dan menyalakan *backlight* LCD.

```
lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");  
lcd.setCursor(0,1); lcd.print("Connecting-WiFi");
```

Menampilkan teks pada LCD.

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
delay(2000);  
lcd.clear();
```

Jeda tampilan teks selama 2000ms atau 2 detik kemudian tampilan LCD dibersihkan.

```
t = dht.readTemperature(); // Membaca suhu dalam Celsius
h = dht.readHumidity();    // Membaca kelembaban Udara
```

Membaca suhu dan kelembaban udara.

```
if(isnan(t) || isnan(h)) {
  Serial.println("Tidak dapat membaca sensor DHT11!");
  return;
}
```

Periksa jika pembacaan sensor gagal (*isnan*) akan diulangi.

```
if(isnan(t)) t=0;
if(isnan(h)) h=0;
```

Set nilai terakhir menjadi 0 jika nilai sensor *isnan*.

```
Serial.println("Temp:" + String(t) + " °C");
Serial.println("Humd:" + String(h) + " %");
Serial.println();
```

Mencetak data suhu dan kelembaban pada serial monitor.

```
Firestore.setFloat(fbdo, "/" + user + "/gauge1", t);
Firestore.setFloat(fbdo, "/" + user + "/gauge2", h);
```

Mengirim data suhu dan kelembaban (tipe data float) ke *database* dengan perintah `Firestore.setFoat` dengan alamat `*/user/gauge1` dan `*/user/gauge2`.

```
lcd.setCursor(0,0);
lcd.print("Temp: " + String(t) + char(223) + "C");
```

Mencetak data suhu pada LCD 16x2 I2C pada baris pertama.

```
lcd.setCursor(0,1);
lcd.print("Humd: " + String(h) + " %");
```

Mencetak data kelembaban pada LCD 16x2 I2C pada baris kedua.

## MODUL 16 | ANTARMUKA DENGAN SENSOR BMP180

Pada modul ini akan membahas mengenai antarmuka dengan sensor BMP180, proyek monitoring dengan sensor BMP180.

### A. Pengenalan Sensor BMP180

BMP180 adalah sensor tekanan barometrik (*digital barometric pressure sensor*) dan temperatur udara dari Bosch Sensortec yang berkinerja sangat tinggi yang dapat diaplikasikan pada berbagai perangkat. BMP180 lebih kecil (lebih hemat energi dengan konsumsi energi sangat rendah, kurang dari 3  $\mu$ A), BMP180 juga menjadi menonjol karena kinerjanya yang sangat stabil terlepas dari pasokan tegangan yang digunakan.



Gambar 16.1. Sensor BMP180

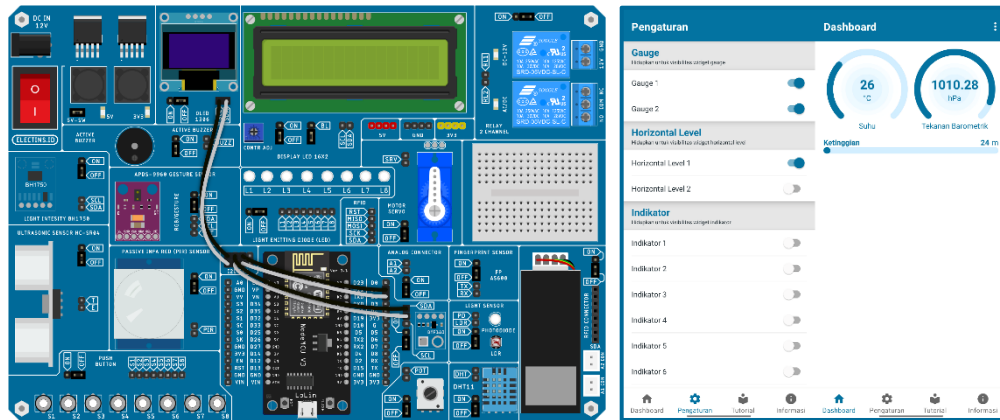
Spesifikasi Sensor BMP180:

- Rentang tekanan 300 - 1100 hPa
- Antarmuka I2C
- Resolusi tekanan 0.01 hPa
- Resolusi suhu 0.1 °C
- Penggunaan daya puncak 0.65 mA
- Tegangan kerja 1.8 - 3.6 VDC
- Suhu kerja -40 sampai +85 °C
- Waktu pendeteksian tekanan 5 mili-detik

### B. Perangkat yang Dibutuhkan

- |                               |        |
|-------------------------------|--------|
| 1. NodeMCU                    | 1 Buah |
| 2. Sensor BMP180              | 1 Buah |
| 3. OLED Display               | 1 Buah |
| 4. I2C Expanded               | 1 Buah |
| 5. Kabel Jumper Female-Female | 6 Buah |

### C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 16.2. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	I2C EXP	HEADER MCU
OLED Display - SDA	I2C Exp - SDA	PIN D2 - SDA
OLED Display - SCL	I2C Exp - SCL	PIN D1 - SCL
BMP180 - SDA		
BMP180 - SCL		

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Gauge 1	Suhu	80	°C
Gauge 2	Tekanan Barometrik	1100	hPa
Horizontal Level 1	Ketinggian	1000	m

### D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* OLED Display dan Sensor BMP180 ke I2C Expanded dan *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* OLED Display dan Sensor BMP180 ke posisi ON.
3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools* > *Board* > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module).
7. Pilih *port* yang terhubung pada *Tools* > *Port* > COM x.

## 8. *Verify (compile) dan Upload program.*

### Contoh Program:

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
// Library BMP180
#include <Adafruit_BMP085.h>

// Mengatur OLED Display dengan resolusi 128x64
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// BMP objek dengan nama bmp
Adafruit_BMP085 bmp;

// Variabel jarak menyimpan nilai sensor BMP180
int temperature;
int pressure;
int altitude;

// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Memulai komunikasi dengan sensor BMP180
  bmp.begin();
  // Memulai komunikasi dengan OLED Display
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
```

```
    delay(300);
}
// Menampilkan status koneksi dan alamat IP
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();

// Menampilkan versi client firebase
Serial.printf("Firebase Client v%s\n\n",
             FIREBASE_CLIENT_VERSION);
// Memulai koneksi dengan database
// Re-koneksi jika WiFi terputus
Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);

// Membersihkan tampilan display
display.clearDisplay();
}

void loop() {
// Membaca data suhu, tekanan udara dan ketinggian
temperature = bmp.readTemperature();
pressure = bmp.readPressure()/100;
altitude = bmp.readAltitude();

// Mencetak data sensor pada serial monitor
Serial.println("Temp : " + String(temperature) + " °C");
Serial.println("pres : " + String(pressure) + " hPa");
Serial.println("Alt : " + String(altitude) + " meter");
Serial.println();

// Mengirim data pembacaan sensor ke database
// Mengirim data suhu ke database *user/gauge1
// Mengirim data tekanan ke database *user/gauge2
// Mengirim data ketinggian ke database *user/hlevel1
Firebase.setInt(fbdo, "/" + user + "/gauge1", temperature);
Firebase.setInt(fbdo, "/" + user + "/gauge2", pressure);
Firebase.setInt(fbdo, "/" + user + "/hlevel1", altitude);

// Menampilkan data pada OLED Display
display.setTextSize(2); display.setTextColor(WHITE);
display.setCursor(1,1); display.print("BMP180");
display.setTextSize(1); display.setTextColor(WHITE);
display.setCursor(1,25); display.print("Temp: " +
    String(temperature) + " " + (char)247 + "C");
display.setCursor(1,35); display.print("Pres: " +
    String(pressure) + " hPa");
display.setCursor(1,45); display.print("Alt : " +
    String(altitude) + " m");
display.display(); display.clearDisplay();
}
```

**Keterangan Program:**

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
```

Library ESP8266 WiFi dan Firebase ESP8266.

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

Library OLED Display.

```
#include <Adafruit_BMP085.h>
```

Library BMP180.

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

Mengatur OLED Display dengan resolusi 128x64.

```
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan Password WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"
```

URL dan token firebase database.

```
FirebaseData fbdo;
Adafruit_BMP085 bmp;
```

Firestore dan BMP objek.

```
int temperature;
int pressure;
int altitude;
```

Variabel untuk menyimpan masing-masing data suhu, tekanan barometrik dan ketinggian.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
bmp.begin();
```

Memulai komunikasi dengan sensor BMP180.

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
```

Memulai komunikasi dengan OLED Display.

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
```



```
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
display.clearDisplay();
```

Membersihkan tampilan *display*.

```
temperature = bmp.readTemperature();  
pressure = bmp.readPressure()/100;  
altitude = bmp.readAltitude();
```

Membaca data suhu, tekanan dan ketinggian. Nilai tekanan per 100 untuk konversi nilai dari Pa menjadi hPa.

```
Serial.println("Temp : " + String(temperature) + " °C");  
Serial.println("pres : " + String(pressure) + " hPa");  
Serial.println("Alt : " + String(altitude) + " meter");  
Serial.println();
```

Mencetak nilai sensor pada serial monitor.

```
Firebase.setInt(fbdo, "/" + user + "/gauge1", temperature);  
Firebase.setInt(fbdo, "/" + user + "/gauge2", pressure);  
Firebase.setInt(fbdo, "/" + user + "/hlevel1", altitude);
```

Mengirim data suhu, tekanan dan ketinggian (tipe data integer) ke *database* dengan perintah `Firebase.setInt` dengan alamat `*/user/gauge1`, `*/user/gauge2` dan `*/user/hlevel1`.

```
display.setTextSize(2); display.setTextColor(WHITE);  
display.setCursor(1,1); display.print("BMP180");
```

```
...  
display.display(); display.clearDisplay();
```

Menampilkan data sensor pada OLED *display*.

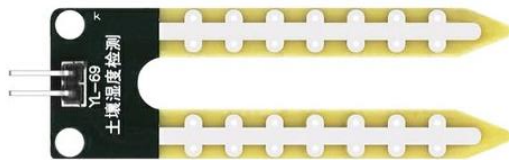
## MODUL 17 | ANTARMUKA DENGAN SOIL MOISTURE/RAIN SENSOR

Pada modul ini akan membahas mengenai antarmuka dengan sensor kelembaban tanah dan sensor kelembaban hujan, proyek monitoring dengan sensor kelembaban tanah dan hujan.

### A. Pengenalan Sensor Kelembaban Tanah dan Sensor Hujan

#### 1. Sensor Kelembaban Tanah

Sensor kelembaban tanah atau *soil moisture* adalah sensor yang dapat mengukur kadar air atau kelembaban pada tanah. Sensor ini diaplikasikan untuk membantu memantau kadar air pada tanaman.



Gambar 17.1. Sensor Kelembaban Tanah

Pada sensor terdapat dua konduktor terbuka yang bertindak sebagai resistor variabel resistansinya berubah sesuai dengan kadar air pada tanah. Semakin banyak air pada tanah akan menghasilkan konduktivitas yang tinggi sehingga nilai resistansinya rendah, sebaliknya jika air pada tanah semakin sedikit akan menghasilkan konduktivitas yang rendah sehingga nilai resistansinya lebih tinggi.

#### 2. Sensor Hujan

Sensor hujan atau *rain drop sensor* adalah sensor yang berfungsi untuk mendeteksi hujan turun atau tidak. Prinsip kerja sensor ini sama dengan sensor kelembaban tanah jika pada bagian papan sensor terkena air, resistansinya akan berubah. Semakin banyak air yang mengenai papan sensor maka nilai resistansinya semakin rendah.

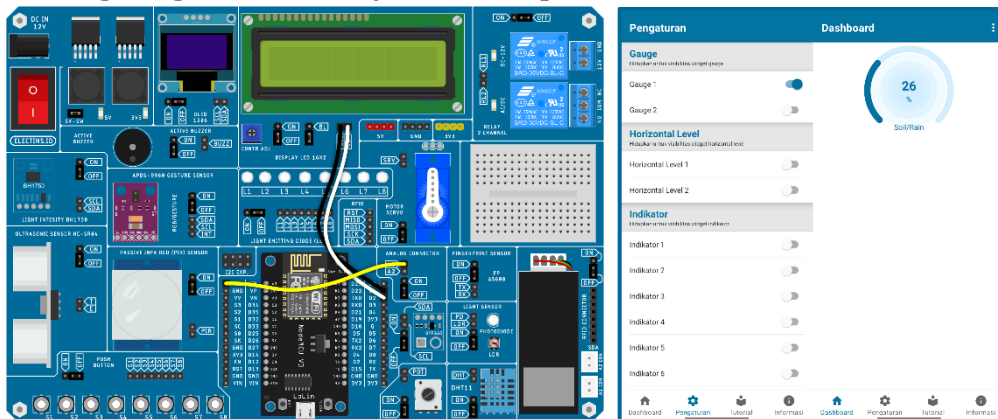


Gambar 17.2. Sensor Hujan

## B. Perangkat yang Dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. Sensor Kelembaban Tanah/Hujan     | 1 Buah |
| 3. LCD 16x2 I2C                      | 1 Buah |
| 4. Kabel <i>Jumper Female-Female</i> | 3 Buah |

## C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 17.3. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

### Rangkaian:

HEADER I/O	HEADER MCU
LCD 16x2 I2C - SDA	PIN D2 - SDA
LCD 16x2 I2C - SCL	PIN D1 - SCL
Sensor Kelembaban Tanah - A1	PIN A0
Sensor Hujan - A2	PIN A0

### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Gauge 1	Soil/Rain	100	%

## D. Petunjuk Praktikum:

1. Pasang sensor kelembaban tanah pada konektor A1 CON dan sensor hujan pada A2 CON.
2. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C dan Analog Connector A1/A2 ke *header* MCU sesuai dengan tabel rangkaian.
3. Pindahkan posisi *cap jumper* LCD 16x2 I2C dan Analog Connector ke posisi ON dan *Backlight* LCD ke posisi (BL).
4. Hidupkan trainer.

5. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
6. Buat *listing* program dengan mengikuti contoh program di bawah.
7. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools > Board > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module)*.
8. Pilih *port* yang terhubung pada *Tools > Port > COM x*.
9. *Verify (compile)* dan *Upload* program.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// PIN Analog Sensor (AS) terhubung ke PIN A0 NodeMCU
#define AS_PIN A0

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);

// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

// Variabel untuk menyimpan data ADC dan data konversi sensor
int as_adc, as_value;
// Variabel untuk menyimpan karakter data sensor
char as_data[3];

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  lcd.init();           // Inisialisasi LCD
  lcd.backlight();     // Menyalakan Backlight LCD
  lcd.print("Trainer Kit IoT"); // Menampilkan Teks pada LCD
  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
}
```

```
// Menampilkan status koneksi dan alamat IP
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();

// Menampilkan versi client firebase
Serial.printf("Firebase Client v%s\n\n",
             FIREBASE_CLIENT_VERSION);
// Memulai koneksi dengan database
// Re-koneksi jika WiFi terputus
Firebase.begin(DATABASE_URL, API_KEY);
Firebase.reconnectWiFi(true);

delay(2000); // Jeda selama 2000ms atau 2 detik
lcd.clear(); // Perintah untuk membersihkan tampilan LCD
}

void loop() {
  // Membaca data ADC sensor
  // ADC disimpan pada 'as_adc'
  // Data konversi disimpan pada 'as_value'
  as_adc = analogRead(AS_PIN);
  // Nilai 760 didapatkan dengan mencelupkan sensor ke air
  // Nilai *760 di ganti dengan nilai adc air
  // Ubah nilai *760 untuk sensor hujan
  as_value = map(as_adc, 0, 760, 0, 100);
  if(as_value > 100) as_value = 100;

  // Mencetak data pada serial monitor
  Serial.println("ADC : " + String(as_adc));
  Serial.println("Nilai : " + String(as_value) + " %");
  Serial.println();

  // Mengirim data sensor ke database
  // dengan alamat *user/gaugel
  Firebase.setInt(fbdo, "/" + user + "/" + "gaugel", as_value);

  // Mencetak data sensor pada LCD 16x2 I2C
  lcd.setCursor(0,0); lcd.print("Analog Sens (%)");
  sprintf(as_data, "AS: %3d", as_value);
  lcd.setCursor(0,1); lcd.print(as_data);

  // Jeda 150 ms agar perubahan nilai lebih halus
  delay(150);
}
```

### Keterangan Program:

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
```

Library ESP8266 WiFi dan Firebase ESP8266.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Library LCD 16x2 I2C.

```
#define WIFI_SSID "SSID_WIFI"  
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan *Password* WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"  
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
#define AS_PIN A0
```

Pin Analog Sensor (LS) terhubung ke Pin A0 NodeMCU

```
FirebaseData fbdo;  
LiquidCrystal_I2C lcd(0x27,16,2);
```

Firebase dan LCD objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
int as_adc, as_value;  
char as_data[3];
```

Variabel 'as\_adc' dan 'as\_value' untuk menyimpan data ADC dan data hasil konversi sensor. Variabel 'as\_data' untuk menyimpan data sensor dalam bentuk karakter.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
lcd.init();  
lcd.backlight();
```

Inisialisasi dan menyalakan *backlight* LCD.

```
lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");  
lcd.setCursor(0,1); lcd.print("Connecting-WiFi");
```

Menampilkan teks pada LCD

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");
```

```
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase *database* dan mengaktifkan koneksi ulang jika WiFi terputus.

```
delay(2000);  
lcd.clear();
```

Jeda tampilan teks selama 2000ms atau 2 detik kemudian tampilan LCD dibersihkan.

```
as_adc = analogRead(AS_PIN);  
as_value = map(as_adc, 0, 760, 0, 100);
```

Membaca data ADC sensor disimpan pada variabel 'as\_adc' kemudian dikonversi menjadi nilai persentase (0-100 %) yang disimpan pada variabel 'as\_value'. Nilai 760 didapatkan dari nilai ADC maksimal sensor ketika dicelupkan ke air sebagai nilai maksimal sensor.

```
if(as_value > 100) as_value = 100;
```

Membatasi nilai 'as\_value' jika melebihi 100 maka nilai diset menjadi 100.

```
Serial.println("ADC : " + String(as_adc));  
Serial.println("Nilai : " + String(as_value) + " %");  
Serial.println();
```

Mencetak data pada serial monitor.

```
Firebase.setInt(fbdo, "/" + user + "/gauge1", ls_value);
```

Mengirim data sensor (tipe data integer) ke database dengan perintah Firebase.setInt dengan alamat \*/user/gauge1.

```
lcd.setCursor(0,0); lcd.print("Analog Sens (%");  
sprintf(as_data, "AS: %3d", as_value);  
lcd.setCursor(0,1); lcd.print(as_data);
```

Mencetak teks "Analog Sens (%)" pada baris pertama dan mencetak nilai sensor pada baris kedua.

## MODUL 18 | ANTARMUKA DENGAN GESTURE SENSOR

Pada modul ini akan membahas mengenai antarmuka dengan sensor gesture APDS 9960, proyek menyalakan lampu LED dengan gesture.

### A. Pengenalan Sensor Gesture APDS 9960

Sensor *gesture* APDS 9960 adalah sebuah sensor dengan pengindraan gerakan tanpa sentuhan. Sensor ini memungkinkan pengontrolan berbagai peralatan hanya dengan sapuan tangan. Sensor ini digunakan pada beberapa *smartphone* dengan fitur *gesture* dan berbagai peralatan cerdas lainnya. Selain itu sensor ini dapat berfungsi sebagai proximity dan deteksi warna.



Gambar 18.1. Sensor Gesture APDS 9960

Spesifikasi Sensor Gesture APDS 9960:

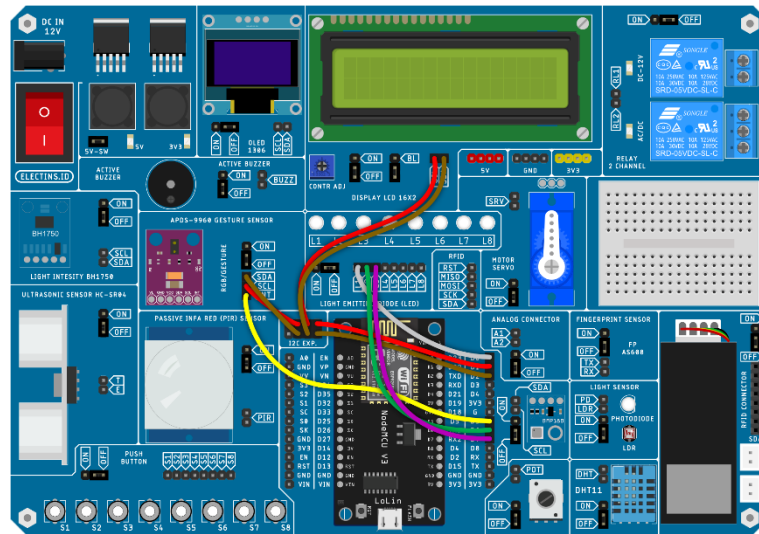
- Sensor Gesture Direction
- Sensor Proximity
- Deteksi Warna
- Chip APDS-9960
- Tegangan kerja 3.3V
- Antarmuka I2C

### B. Perangkat yang Dibutuhkan

- |                                      |         |
|--------------------------------------|---------|
| 1. NodeMCU                           | 1 Buah  |
| 2. Sensor Gesture APDS 9960          | 1 Buah  |
| 3. OLED Display                      | 1 Buah  |
| 4. I2C Expanded                      | 1 Buah  |
| 5. LED                               | 3 Buah  |
| 6. Kabel <i>Jumper Female-Female</i> | 10 Buah |



### C. Wiring Diagram



Gambar 18.2. Wiring Diagram

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	I2C EXP	HEADER MCU
OLED <i>Display</i> - SDA	I2C Exp - SDA	PIN D2 - SDA
OLED <i>Display</i> - SCL	I2C Exp - SCL	PIN D1 - SCL
<i>Gesture Sensor</i> - SDA		
<i>Gesture Sensor</i> - SCL		
<i>Gesture Sensor</i> - INT		PIN D5
LED - L1		PIN D0
LED - L2		PIN D6
LED - L3		PIN D7

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
-	-	-	-

### D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C, *Gesture Sensor* ke I2C *Expanded* dan LED *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C, *Gesture Sensor* dan LED ke posisi ON.
3. Hidupkan trainer.

4. Buat *listing* program dengan mengikuti contoh program di bawah.
5. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools > Board > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module)*.
6. Pilih *port* yang terhubung pada *Tools > Port > COM x*.
7. *Verify (compile)* dan *Upload* program.

**Contoh Program:**

```
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Library APDS9960
#include <SparkFun_APDS9960.h>

// Koneksi PIN APDS-9960 ke PIN NodeMCU
#define APDS9960_SDA D2
#define APDS9960_SCL D1
#define APDS9960_INT D5

// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);

// Variabel Global APDS9960
SparkFun_APDS9960 apds = SparkFun_APDS9960();
volatile bool isr_flag = 0;

// Fungsi interupsi pada IRAM
void ICACHE_RAM_ATTR interruptRoutine();

// Variabel data lampu dan pin lampu
int pos = 0, lamp_pin[3] = {D0, D6, D7};
// Variabel status lampu
boolean lamp_state[3];
String state;

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Memulai komunikasi I2C
  Wire.begin(APDS9960_SDA,APDS9960_SCL);
  // Mengatur pin interrupt sebagai input
  pinMode(digitalPinToInterrupt(APDS9960_INT), INPUT);
  // Inisialisasi Interupsi
  attachInterrupt(digitalPinToInterrupt(APDS9960_INT),
    interruptRoutine, FALLING);
  // Inisialisasi APDS-9960
  apds.init();
  // Mengaktifkan mode gesture
  apds.enableGestureSensor(true);

  // Mengatur pin lampu sebagai output
  // dengan kondisi awal LOW
  for(int i=0; i<=2; i++){
    Serial.println(i);
```

```
pinMode(lamp_pin[i], OUTPUT);
digitalWrite(lamp_pin[i], LOW);
}

lcd.init(); // Inisialisasi LCD
lcd.backlight(); // Menyalakan Backlight LCD
lcd.print("Trainer Kit IoT");
delay(2000); // Jeda tampilan teks pada LCD
lcd.clear(); // Membersihkan tampilan LCD
}

void loop(){
// Membaca status ISR (Interrupt Service Routine)
// Menjalankan fungsi handleGesture()
if(isr_flag == 1){
detachInterrupt(digitalPinToInterrupt(APDS9960_INT));
handleGesture();
isr_flag = 0;
attachInterrupt(digitalPinToInterrupt(APDS9960_INT),
interruptRoutine, FALLING);
}

// mengatur variabel pos
// Jika lebih dari 2 pos menjadi 0
// Jika kurang dari 0 pos menjadi 2
if(pos > 2) pos = 0;
if(pos < 0) pos = 2;
// Nyalakan dan matikan lampu
// Sesuai pin pada variabel lampu_pin dengan indeks pos
// Status pada variabel lamp_state dengan indeks pos
digitalWrite(lamp_pin[pos], lamp_state[pos]);

// Mencetak teks pada LCD
lcd.setCursor(0, 0); lcd.print("Lampu Gesture ");
lcd.setCursor(0, 1); lcd.print("> Lampu");
lcd.setCursor(8, 1); lcd.print(pos+1);
lcd.setCursor(9, 1); lcd.print(":");
lcd.setCursor(11, 1); lcd.print(state);

// Mengubah status lamp_state dengan teks
if(lamp_state[pos] == true) state = "ON ";
if(lamp_state[pos] == false) state = "OFF";
}

// Fungsi InterruptRoutine
void interruptRoutine(){
isr_flag = 1;
}

// Fungsi handleGesture
// Membaca arah gesture
// Mengatur variabel 'pos' dan 'lamp_state' dengan indeks pos
void handleGesture(){
if(apds.isGestureAvailable()){
switch(apds.readGesture()){
case DIR_UP:
```

```
    lamp_state[pos] = true;
    Serial.println("ATAS");
    break;
case DIR_DOWN:
    lamp_state[pos] = false;
    Serial.println("BAWAH");
    break;
case DIR_LEFT:
    pos--;
    Serial.println("KIRI");
    break;
case DIR_RIGHT:
    pos++;
    Serial.println("KANAN");
    break;
default:
    Serial.println("TIDAK ADA");
}
}
```

### Keterangan Program:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Library LCD 16x2 I2C.

```
#include <SparkFun_APDS9960.h>
```

Library APDS9960.

```
#define APDS9960_SDA D2
#define APDS9960_SCL D1
#define APDS9960_INT D5
```

Koneksi Pin APS9960.

```
LiquidCrystal_I2C lcd(0x27,16,2);
```

LCD objek.

```
SparkFun_APDS9960 apds = SparkFun_APDS9960();
volatile bool isr_flag = 0;
```

Variabel global APDS9960.

```
void ICACHE_RAM_ATTR interruptRoutine();
```

Fungsi interupsi pada IRAM.

```
int pos = 0, lamp_pin[3] = {D0, D6, D7};
boolean lamp_state[3];
String state;
```

Variabel data, pin dan status lampu.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
Wire.begin(APDS9960_SDA,APDS9960_SCL);
```

Memulai komunikasi I2C.

```
pinMode(digitalPinToInterrupt(APDS9960_INT), INPUT);
```

Mengatur Pin interupsi sebagai input.

```
attachInterrupt(digitalPinToInterrupt(APDS9960_INT),  
interruptRoutine, FALLING);
```

Inisialisasi interupsi.

```
apds.init();
```

Inisialisasi APDS9960.

```
apds.enableGestureSensor(true);
```

Mengaktifkan mode *gesture*.

```
for(int i=0; i<=2; i++){  
    ...  
}
```

Mengatur pin lampu sebagai output dengan kondisi awal LOW.

```
lcd.init(); // Inisialisasi LCD
```

```
...  
lcd.clear(); // Membersihkan tampilan LCD
```

Inisialisasi dan menyalakan *backlight* LCD, mencetak teks pada LCD, jeda selama 2000ms atau 2 detik kemudian LCD dibersihkan.

```
if(isr_flag == 1){  
    detachInterrupt(digitalPinToInterrupt(APDS9960_INT));  
    handleGesture();  
    isr_flag = 0;  
    attachInterrupt(digitalPinToInterrupt(APDS9960_INT),  
interruptRoutine, FALLING);  
}
```

Membaca status ISR (*Interrupt Service Routine*), menjalankan fungsi `handleGesture()`.

```
if(pos > 2) pos = 0;  
if(pos < 0) pos = 2;
```

Mengatur variabel `pos` jika lebih dari 2 `pos` menjadi 0, jika kurang dari 0 `pos` menjadi 2.

```
digitalWrite(lamp_pin[pos], lamp_state[pos]);
```

Menyalakan dan matikan lampu sesuai dengan pin pada variabel '`lamp_pin`' dengan indeks '`pos`'. Status lampu ON atau OFF pada variabel '`lamp_state`' dengan indeks '`pos`'.

```
lcd.setCursor(0, 0); lcd.print("Lampu Gesture ");  
lcd.setCursor(0, 1); lcd.print("> Lampu");  
lcd.setCursor(8, 1); lcd.print(pos+1);  
lcd.setCursor(9, 1); lcd.print(":");  
lcd.setCursor(11, 1); lcd.print(state);
```

Mencetak teks, posisi lampu (`pos`) dan status lampu.

```
if(lamp_state[pos] == true) state = "ON ";
```

```
if(lamp_state[pos] == false) state = "OFF";
```

Mengubah status lampu pada LCD dengan teks.

```
void interruptRoutine() {  
  isr_flag = 1;  
}
```

Fungsi InterruptRoutine.

```
void handleGesture() {  
  if(apds.isGestureAvailable()) {  
    switch(apds.readGesture()) {  
      case DIR_UP:  
        lamp_state[pos] = true;  
        Serial.println("ATAS");  
        break;  
      case DIR_DOWN:  
        lamp_state[pos] = false;  
        Serial.println("BAWAH");  
        break;  
      case DIR_LEFT:  
        pos--;  
        Serial.println("KIRI");  
        break;  
      case DIR_RIGHT:  
        pos++;  
        Serial.println("KANAN");  
        break;  
      default:  
        Serial.println("TIDAK ADA");  
    }  
  }  
}
```

Fungsi handleGesture untuk membaca arah *gesture*. Arah *gesture* mengatur variabel 'pos' dan 'lamp\_state'. *Gesture* ke kanan menambah nilai 'pos', *gesture* ke kiri mengurangi nilai 'pos'. *Gesture* ke atas mengatur nilai 'lamp\_state' dengan indeks 'pos' menjadi *true* (1), *gesture* ke bawah mengatur nilai 'lamp\_state' dengan indeks 'pos' menjadi *false* (0).

## MODUL 19 | ANTARMUKA DENGAN RFID MFRC522

Pada modul ini akan membahas mengenai antarmuka dengan RFID MFRC522 dengan proyek membaca NUID Kartu RFID dengan aplikasi IoT KIT.

### A. Pengenalan RFID

RFID atau *Radio Frequency Identification* merupakan salah satu sistem identifikasi yang memanfaatkan gelombang radio melalui medan elektromagnetik. RFID disebut juga sebagai salah satu metode identifikasi pengambilan data secara otomatis atau *Automatic Identification and Data Capture (AIDC)*. RFID mempunyai 2 bagian utama yaitu :

#### 1. RFID Tag

Merupakan sebuah perangkat yang akan diidentifikasi oleh RFID Reader yang dapat berupa perangkat pasif maupun aktif yang berisi suatu data atau informasi.

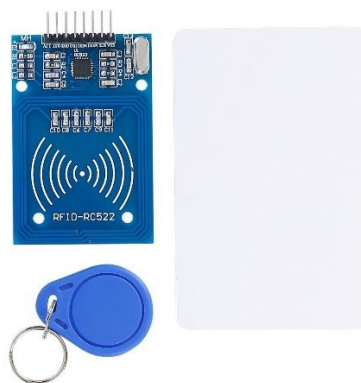
#### 2. RFID Reader

Merupakan alat yang berfungsi untuk membaca data dari RFID Tag.

Pada Trainer Kit Mikrokontroler Arduino menggunakan RFID RC522 dengan frekuensi 13.56 MHz yang memungkinkan pembacaan dan penulisan *chip* RFID dengan jarak yang dekat.

Spesifikasi RFID RC522 :

- Tegangan dan arus kerja : DC 3.3V/13-26mA
- Arus *Idle* : 10-13mA/DC 3.3V
- Arus Puncak : 30mA
- Kecepatan *transfer rate* data : *maximum* 10Mbit/s
- Frekuensi kerja : 13.56MHz
- Ukuran dari RFID Reader : 40 x 60mm
- Suhu kerja : -20 – 80 derajat Celsius
- Menggunakan Antarmuka SPI

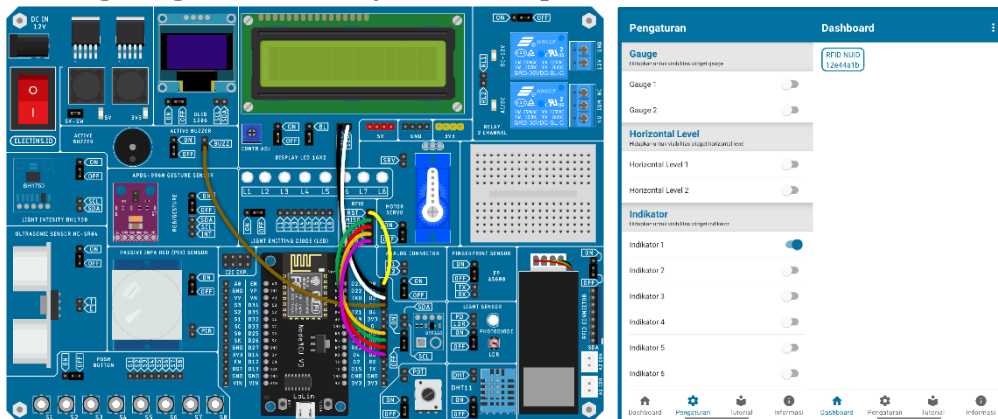


Gambar 19.1. RFID RC522

## B. Perangkat yang dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. RFID RC522                        | 1 Buah |
| 3. LCD 16x2 I2C                      | 1 Buah |
| 4. Kabel <i>Jumper Female-Female</i> | 8 Buah |

## C. Wiring Diagram dan Penyesuaian Aplikasi



Gambar 19.2. Wiring Diagram dan Penyesuaian Aplikasi

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

### Rangkaian:

HEADER I/O	HEADER MCU
LCD 16x2 I2C - SDA	PIN D2 - SDA
LCD 16x2 I2C - SCL	PIN D1 - SCL
Buzzer -BUZZ	PIN D3
RFID - RST	PIN D0
RFID - MISO	PIN D6
RFID - MOSI	PIN D7
RFID - SCK	PIN D5
RFID - SDA	PIN D8

### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
Indikator 1	RFID NUID	-	-

## D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* LCD 16x2 I2C dan RFID ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C dan RFID ke posisi ON.



3. Hidupkan trainer.
4. Buka aplikasi IoT KIT, atur sesuai dengan tabel pengaturan aplikasi.
5. Buat *listing* program dengan mengikuti contoh program di bawah.
6. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools* > *Board* > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module).
7. Pilih *port* yang terhubung pada *Tools* > *Port* > *COM x*.
8. *Verify (compile)* dan *Upload* program.

**Contoh Program:**

```
// Library ESP8266 WiFi dan Firebase ESP8266
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Library RFID
#include <SPI.h>
#include <MFRC522.h>

// Koneksi PIN RFID ke PIN NodeMCU
// RST/Reset      RST           D0
// SPI SS         SDA(SS)       D8
// SPI MOSI       MOSI          D7
// SPI MISO       MISO          D6
// SPI SCK        SCK           D5

#define RST_PIN D0
#define SS_PIN  D8

// SSID dan Password WiFi
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"

// URL Firebase dan Token Database
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"

// PIN Buzzer terhubung ke PIN D3 NodeMCU
#define BUZZ_PIN  D3

// Firebase objek dengan nama fbdo
FirebaseData fbdo;
// Nama pengguna pada Aplikasi IoT KIT
String user = "user_id";

// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);
// MFRC522 Objek dengan nama mrfc522
MFRC522 mrfc522(SS_PIN, RST_PIN);

// Variabel untuk menampung NUID kartu
String nuid;
String rfid_tag = "12e44a1b";
```

```
void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Mengatur BUZZ_PIN sebagai Output
  // Kondisi awal BUZZ_PIN LOW (Buzzer OFF)
  pinMode(BUZZ_PIN, OUTPUT);
  digitalWrite(BUZZ_PIN, LOW);

  SPI.begin();          // Memulai komunikasi SPI dengan RFID
  mfrc522.PCD_Init();  // Inisialisasi RFID

  lcd.init();          // Inisialisasi LCD
  lcd.backlight();    // Menyalakan Backlight LCD
  lcd.print("Trainer Kit IoT"); // Menampilkan Teks pada LCD
  // Memulai koneksi WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  // Menampilkan status koneksi dan alamat IP
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Menampilkan versi client firebase
  Serial.printf("Firebase Client v%s\n\n",
               FIREBASE_CLIENT_VERSION);
  // Memulai koneksi dengan database
  // Re-koneksi jika WiFi terputus
  Firebase.begin(DATABASE_URL, API_KEY);
  Firebase.reconnectWiFi(true);

  delay(2000);        // Jeda tampilan teks pada LCD
  lcd.clear();        // Membersihkan tampilan LCD
}

void loop() {
  // Mencetak teks pada LCD baris pertama dan kedua
  lcd.setCursor(0,0); lcd.print("RFID READER");
  lcd.setCursor(0,1); lcd.print("NUID:");

  // Membaca dan verifikasi kartu
  if(!mfrc522.PICC_IsNewCardPresent()) { return; }
  if(!mfrc522.PICC_ReadCardSerial()) { return; }
  Serial.print("UID tag : ");
  String tag_id;

  // NUID disatukan dan disimpan pada 'tag_id'
  for(byte i = 0; i < mfrc522.uid.size; i++) {
    tag_id.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
}
```

```
// Menyimpan NUID ke variabel global 'nuid'
nuid = tag_id;

// Pencocokan NUID yang tersimpan dengan yang terbaca
// Nada buzzer scan, kartu diterima dan ditolak
if(rfid_tag == nuid){
    buzz_auth();
    buzz_accepted();
}else{
    buzz_auth();
    buzz_denied();
}

if(nuid){
    // Mengirim data NUID ke database
    // dengan alamat user/indicator1
    Firebase.setString(fbdo, "/" + user + "/indicator1", nuid);
}

// Mencetak NUID pada LCD dan Serial Monitor
lcd.setCursor(5, 1); lcd.print(nuid);
Serial.println(nuid);
}

// Fungsi Nada Buzzer scan kartu
void buzz_auth() {
    for(int x = 0; x < 8 ; x++ ) {
        digitalWrite(BUZZ_PIN, HIGH);
        delay(50);
        digitalWrite(BUZZ_PIN, LOW);
        delay(30);
    }
    delay(500);
}

// Fungsi Nada Buzzer akses diterima
void buzz_accepted() {
    for(int x = 0; x < 2 ; x++ ) {
        digitalWrite(BUZZ_PIN, HIGH);
        delay(200);
        digitalWrite(BUZZ_PIN, LOW);
        delay(100);
    }
}

// Fungsi Nada Buzzer akses ditolak
void buzz_denied() {
    digitalWrite(BUZZ_PIN, HIGH);
    delay(1000);
    digitalWrite(BUZZ_PIN, LOW);
    delay(100);
}
```

**Keterangan Program:**

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
```

Library ESP8266 WiFi dan Firebase ESP8266.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Library LCD 16x2 I2C.

```
#include <SPI.h>
#include <MFRC522.h>
```

Library RFID MFRC522

```
#define RST_PIN D0
#define SS_PIN D8
```

Koneksi Pin RFID ke NodeMCU.

```
#define WIFI_SSID "SSID_WIFI"
#define WIFI_PASSWORD "PASS_WIFI"
```

SSID dan Password WiFi.

```
#define DATABASE_URL "project_id.firebaseio.com"
#define API_KEY "database_secret"
```

URL dan *token* firebase *database*.

```
#define BUZZ_PIN D3
```

Pin Buzzer terhubung ke Pin D3 NodeMCU.

```
FirebaseData fbdo;
LiquidCrystal_I2C lcd(0x27,16,2);
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

Firestore, LCD MFRC522 objek.

```
String user = "user_id";
```

Variabel *user*. Gunakan sesuai dengan *user* ID yang didaftarkan pada aplikasi IoT KIT.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
pinMode(BUZZ_PIN, OUTPUT);
digitalWrite(BUZZ_PIN, LOW);
```

Mengatur Pin Buzzer sebagai Output dengan kondisi awal LOW (Buzzer OFF).

```
SPI.begin(); // Memulai komunikasi SPI dengan RFID
mfrc522.PCD_Init(); // Inisialisasi RFID
```

Memulai komunikasi SPI dan inisialisasi RFID.

```
lcd.init();
lcd.backlight();
```

Inisialisasi dan menyalakan *backlight* LCD.

```
lcd.setCursor(0,0); lcd.print("Trainer Kit IoT");
lcd.setCursor(0,1); lcd.print("Connecting-WiFi");
```

Menampilkan teks pada LCD.

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
...  
Serial.println(WiFi.localIP());  
Serial.println();
```

Koneksi dengan Wi-Fi, status koneksi dan IP ditampilkan pada serial monitor.

```
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);  
Firebase.begin(DATABASE_URL, API_KEY);  
Firebase.reconnectWiFi(true);
```

Menampilkan versi klien firebase, memulai koneksi dengan firebase database dan mengaktifkan koneksi ulang jika WiFi terputus.

```
delay(2000);  
lcd.clear();
```

Jeda tampilan teks selama 2000ms atau 2 detik kemudian tampilan LCD dibersihkan.

```
lcd.setCursor(0,0); lcd.print("RFID READER");  
lcd.setCursor(0,1); lcd.print("NUID:");
```

Mencetak teks pada baris pertama dan baris kedua.

```
// Membaca dan verifikasi kartu  
if(!mfr522.PICC_IsNewCardPresent()) { return; }  
if(!mfr522.PICC_ReadCardSerial()) { return; }  
Serial.print("UID tag : ");
```

Membaca dan verifikasi kartu.

```
// NUID disatukan dan disimpan pada 'tag_id'  
for(byte i = 0; i < mfr522.uid.size; i++) {  
  tag_id.concat(String(mfr522.uid.uidByte[i], HEX));  
}  
// Menyimpan NUID ke variabel global 'nuid'  
nuid = tag_id;
```

NUID di satukan dan di simpan pada 'tag\_id'. NUID pada 'tag\_id' disimpan pada variabel global 'nuid'.

```
if(rfid_tag == nuid){  
  ...  
}else{  
  ...  
}
```

Pencocokan NUID yang tersimpan dengan yang terbaca dengan nada Buzzer scan, akses diterima dan ditolak.

```
if(nuid){  
  Firebase.setString(fbdo, "/" + user + "/indicator1", nuid);  
}
```

Mengirim NUID ke *database* dengan alamat *\*/user/indicator1*.

```
lcd.setCursor(5, 1); lcd.print(nuid);  
Serial.println(nuid);
```

Mencetak NUID pada LCD dan serial monitor.

```
void buzz_auth() {  
  for(int x = 0; x < 8 ; x++ ) {  
    ...  
  }  
  delay(500);  
}
```

Fungsi nada Buzzer scan kartu.

```
void buzz_accepted() {  
  for(int x = 0; x < 2 ; x++ ) {  
    ...  
  }  
}
```

Fungsi nada Buzzer akses diterima.

```
void buzz_denied() {  
  ...  
}
```

Fungsi nada Buzzer akses ditolak.

## MODUL 20 | ANTARMUKA DENGAN SENSOR SIDIK JARI

Pada modul ini akan membahas mengenai antarmuka dengan sensor sidik jari dengan proyek daftar, baca dan hapus data sidik jari.

### A. Pengenalan Sensor Sidik Jari

Sensor sidik jari atau *fingerprint* sensor adalah sebuah alat elektronik yang menggunakan sensor *scanning* untuk mengetahui sidik jari seseorang sebagai verifikasi identitas. Setiap manusia dilahirkan dengan sidik jari yang unik sehingga mempunyai sidik jari yang berbeda-beda, sehingga keunikan ini dapat digunakan sebagai identitas yang bisa di gunakan untuk verifikasi.



Gambar 20.1. Sensor Sidik Jari

Prinsip kerja sensor *fingerprint* secara umum:

- **Enroll** yaitu pendaftaran atau pengenalan sidik jari sebagai ID
- **Storage** yaitu penyimpanan data IDE
- **Matching and comparing** yaitu pencocokan dan perbandingan

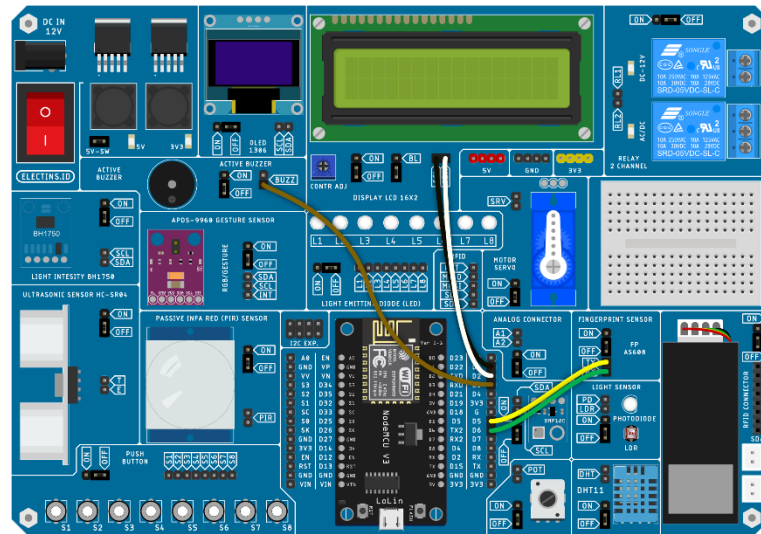
Pada trainer menggunakan *fingerprint* dengan tipe AS608 dengan spesifikasi sebagai berikut:

- Resolusi 500 dpi
- Tegangan kerja 3.3V
- Arus kerja <60mA
- Pembacaan sidik jari <1.0 detik
- Antarmuka USB/UART
- Suhu kerja -20 sampai +50 °C

### B. Perangkat yang dibutuhkan

- |                                      |        |
|--------------------------------------|--------|
| 1. NodeMCU                           | 1 Buah |
| 2. Sensor Sidik Jari                 | 1 Buah |
| 3. Buzzer                            | 1 Buah |
| 4. Kabel <i>Jumper Female-Female</i> | 5 Buah |

### C. Wiring Diagram



Gambar 20.2. Wiring Diagram

Rangkaian pada gambar dapat diimplementasikan pada trainer dan aplikasi IoT KIT dengan mengikuti tabel berikut:

#### Rangkaian:

HEADER I/O	HEADER MCU
LCD 16x2 I2C - SDA	PIN D2 - SDA
LCD 16x2 I2C - SCL	PIN D1 - SCL
Sensor Sidik Jari - TX	PIN D5
Sensor Sidik Jari - RX	PIN D6
Buzzer - BUZZ	PIN D3

#### Pengaturan Aplikasi:

VISIBILITAS	LABEL	NILAI MAKS	SATUAN
-	-	-	-

### D. Petunjuk Praktikum:

1. Hubungkan kabel *jumper* dari *header* Sensor Sidik Jari dan Relay ke *header* MCU sesuai dengan tabel rangkaian.
2. Pindahkan posisi *cap jumper* LCD 16x2 I2C dan RFID ke posisi ON.
3. Hidupkan trainer.
4. Buat *listing* program dengan mengikuti contoh program di bawah.  
Terdapat 3 *listing* program yang dapat digunakan untuk sensor sidik jari dengan masing-masing fungsi:
  - *Listing* program *enroll*, berfungsi untuk mendaftarkan sidik jari yang disimpan pada IC/*chip* sensor sidik jari.



- *Listing* program *read*, berfungsi untuk membaca data sidik jari sebagai IDE yang sebelumnya telah di daftarkan.
  - *Listing* program *delete*, berfungsi untuk menghapus data sidik jari yang tersimpan.
5. Pilih *board* NodeMCU 1.0 (ESP-12E Module) pada *Tools* > *Board* > ESP8266 Boards > NodeMCU 1.0 (ESP-12E Module).
  6. Pilih *port* yang terhubung pada *Tools* > *Port* > *COM x*.
  7. *Verify (compile)* dan *Upload* program.

### Contoh Program Enroll:

```
// Library Sidik Jari
#include <Adafruit_Fingerprint.h>
// Library LCD 16x2 I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Library Software Serial
#include <SoftwareSerial.h>

// PIN Buzzer terhubung ke PIN D3 NodeMCU
#define BUZZ_PIN D3

// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2
LiquidCrystal_I2C lcd(0x27,16,2);

// Koneksi PIN Fingerprint ke PIN NodeMCU
// TX D5
// RX D6
// SoftwareSerial objek dengan nama FP_Serial
SoftwareSerial FP_Serial(D5, D6);

// Fingerprint Objek dengan nama finger
// dengan koneksi serial FP_Serial
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&FP_Serial);
// Variabel untuk menyimpan ID
int id;

void setup() {
  // Serial Monitor pada baudrate 115200
  Serial.begin(115200);

  // Mengatur BUZZ_PIN sebagai Output
  // Kondisi awal BUZZ_PIN LOW (Buzzer OFF)
  pinMode(BUZZ_PIN, OUTPUT);
  digitalWrite(BUZZ_PIN, LOW);

  // Komunikasi Serial FP dengan baudrate 57600
  finger.begin(57600);

  lcd.init(); // Inialisasi LCD
  lcd.backlight(); // Menyalakan Backlight LCD
  lcd.print("Trainer Kit IoT");
  delay(2000); // Jeda tampilan teks pada LCD
  lcd.clear(); // Membersihkan tampilan LCD
```

```
// Cek koneksi fingerprint
if(finger.verifyPassword()) {
  lcd.setCursor(0,0); lcd.print("Sensor ditemukan");
  Serial.println("Sensor ditemukan");
  delay(1000);
  lcd.clear();
}else{
  lcd.setCursor(0,0); lcd.print("Tidak Ada Sensor");
  Serial.println("Tidak Ada Sensor");
  delay(1000);
  lcd.clear();
}
}

// Membaca angka (ID) dari serial monitor
int baca_id() {
  uint8_t num = 0;
  while (num == 0) {
    while (!Serial.available());
    num = Serial.parseInt();
  }
  buzz_notif();
  return num;
}

void loop() {
  // Mencetak teks ke serial monitor dan LCD 16x2 I2C
  Serial.println("Ketik ID (1-127) yang akan didaftarkan!");
  lcd.setCursor(0,0); lcd.print("Ketik ID (1-127)");
  lcd.setCursor(0,1); lcd.print(" Serial Monitor ");

  // Memabaca ID
  id = baca_id();
  if(id == 0) {
    return;
  }
  Serial.print("Mendaftar ID#");
  Serial.println(id);

  while(!getFingerprintEnroll());
}

// Fungsi pendaftaran sidik jari
uint8_t getFingerprintEnroll() {
  int p = -1;
  // Memindai Sidik jari (1)
  lcd.setCursor(0,0); lcd.print("Tempel jari anda");
  lcd.setCursor(0,1); lcd.print(".....");
  while(p != FINGERPRINT_OK) {
    p = finger.getImage();
    if(p == FINGERPRINT_OK) {
      buzz_scan();
      lcd.setCursor(0,0); lcd.print("Scanning sensor ");
      lcd.setCursor(0,1); lcd.print(".....");
      delay(2000);
    }
  }
}
```

```
        Serial.println("Memindai sidik jari (1)");
        break;
    }
}

// Konversi sidik jari (1)
p = finger.image2Tz(1);
if(p == FINGERPRINT_OK) {
    Serial.println("Konversi sidik jari (1)");
} else {
    return p;
}

Serial.println("Lepas jari");
buzz_notif();
lcd.setCursor(0,0); lcd.print("Lepas jari      ");
lcd.setCursor(0,1); lcd.print(".....");
p = 0;
while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
}

// Memindai sidik jari (2)
p = -1;
Serial.println("Letakkan lagi jari yang sama");
lcd.setCursor(0,0); lcd.print("Letakkan kembali");
lcd.setCursor(0,1); lcd.print(".....");
while (p != FINGERPRINT_OK) {
    p = finger.getImage();

    if(p == FINGERPRINT_OK) {
        buzz_scan();
        Serial.println("Memindai sidik jari (2)");
        break;
    }
}

// Konversi sidik jari (2)
p = finger.image2Tz(2);
if(p == FINGERPRINT_OK) {
    Serial.println("Konversi sidik jari (2)");
} else {
    return p;
}

// Pencocokan sidik jari
p = finger.createModel();
if(p == FINGERPRINT_OK) {
    Serial.println("Sidik jari cocok");
    buzz_match();
    lcd.setCursor(0,0); lcd.print("Sidik jari cocok");
    lcd.setCursor(0,1); lcd.print(".....");
    delay(2000);
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
    Serial.println("Sidik jari tidak cocok");
    buzz_not_match();
}
```

```
    lcd.setCursor(0,0); lcd.print("Tidak cocok   ");
    lcd.setCursor(0,1); lcd.print("Ulangi proses ");
    delay(2000);
    return p;
}

// Menyimpan data sidik jari
p = finger.storeModel(id);
if(p == FINGERPRINT_OK) {
    Serial.println("Sidik jari disimpan!");
    buzz_store_id();
    lcd.setCursor(0,0); lcd.print("Menyimpan data..");
    lcd.setCursor(0,1); lcd.print("ID " + String(id) + "
                                Tersimpan ");

    delay(3000);
}else{
    Serial.println("Sidik jari disimpan!");
    buzz_not_match();
    lcd.setCursor(0,0); lcd.print("Gagal Menyimpan ");
    lcd.setCursor(0,1); lcd.print("Ulangi proses ");
    delay(3000);
    return p;
}
return true;
}

// Fungsi Nada Buzzer scan sidik jari
void buzz_scan() {
    for(int x = 0; x < 8 ; x++ ) {
        digitalWrite(BUZZ_PIN, HIGH);
        delay(50);
        digitalWrite(BUZZ_PIN, LOW);
        delay(30);
    }
}

// Fungsi Nada Buzzer notifikasi
void buzz_notif(){
    digitalWrite(BUZZ_PIN, HIGH);
    delay(200);
    digitalWrite(BUZZ_PIN, LOW);
    delay(100);
}

// Fungsi Nada Buzzer sidik jari cocok
void buzz_match() {
    for(int x = 0; x < 2 ; x++ ) {
        digitalWrite(BUZZ_PIN, HIGH);
        delay(200);
        digitalWrite(BUZZ_PIN, LOW);
        delay(100);
    }
}

// Fungsi Nada Buzzer sidik jari tidak cocok
void buzz_not_match() {
```

```
digitalWrite(BUZZ_PIN, HIGH);
delay(1000);
digitalWrite(BUZZ_PIN, LOW);
delay(100);
}

// Fungsi Nada Buzzer menyimpan ID sidik jari
void buzz_store_id(){
  for(int x = 0; x < 2 ; x++ ) {
    digitalWrite(BUZZ_PIN, HIGH);
    delay(200);
    digitalWrite(BUZZ_PIN, LOW);
    delay(100);
  }
  digitalWrite(BUZZ_PIN, HIGH);
  delay(500);
  digitalWrite(BUZZ_PIN, LOW);
  delay(100);
}
```

**Keterangan Program:**

```
#include <Adafruit_Fingerprint.h>
```

*Library* sensor sidik jari.

```
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>
```

*Library* LCD 16x2 I2C.

```
#include <SoftwareSerial.h>
```

*Library* software serial.

```
#define BUZZ_PIN D3
```

Pin buzzer terhubung ke pin D3 NodeMCU.

```
LiquidCrystal_I2C lcd(0x27,16,2);
```

LCD objek.

```
SoftwareSerial FP_Serial(D5, D6);
```

Software serial objek dengan nama FP\_serial dengan pin D5 dan D6.

```
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&FP_Serial);
```

*Fingerprint* objek dengan nama *finger* dengan koneksi serial FP\_Serial.

```
int id;
```

Variabel untuk menyimpan ID.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
pinMode(BUZZ_PIN, OUTPUT);
```

```
digitalWrite(BUZZ_PIN, LOW);
```

Mengatur buzzer sebagai output dengan kondisi awal LOW (Buzzer OFF).

```
finger.begin(57600);
```

komunikasi serial dengan sensor sidik jari pada *baudrate* 57600.

```
lcd.init();  
lcd.backlight();  
lcd.print("Trainer Kit IoT");  
delay(2000);  
lcd.clear();
```

Inisialisasi LCD, menyalakan *backlight* dan mencetak teks dengan jeda 2000ms atau 2 detik kemudian LCD dibersihkan.

```
if(finger.verifyPassword()) {  
    ...  
}else{  
    ...  
}
```

Cek koneksi sensor *fingerprint*, data ditampilkan pada LCD dan serial monitor.

```
int baca_id() {  
    uint8_t num = 0;  
    while (num == 0) {  
        while (!Serial.available());  
        num = Serial.parseInt();  
    }  
    buzz_done();  
    return num;  
}
```

Fungsi membaca angka dari serial monitor dengan nilai balik angka sebagai ID.

```
Serial.println("Ketik ID (1-127) yang akan didaftarkan!");  
lcd.setCursor(0,0); lcd.print("Ketik ID (1-127)");  
lcd.setCursor(0,1); lcd.print(" Serial Monitor ");
```

Mencetak teks ke serial monitor dan LCD

```
id = baca_id();  
if(id == 0) {  
    return;  
}  
Serial.print("Mendaftar ID#");  
Serial.println(id);  
while(!getFingerprintEnroll());
```

Membaca ID, jika ID bernilai 0 akan mengulang proses sampai 'id' tidak bernilai 0. Nilai pada 'id' dicetak pada serial monitor dan melakukan proses *looping* pada fungsi *getFingerprintEnroll*.

```
int p = -1;  
...  
while(p != FINGERPRINT_OK) {  
    ...  
}
```

Proses memindai sidik jari (1).

```
p = finger.image2Tz(1);  
...  
...  
while (p != FINGERPRINT_NOFINGER) {  
    ...  
}
```

Proses konversi sidik jari (1).

```
int p = -1;  
...  
while(p != FINGERPRINT_OK) {  
    ...  
}
```

Proses memindai sidik jari (2).

```
p = finger.image2Tz(2);  
...  
...  
while (p != FINGERPRINT_NOFINGER) {  
    ...  
}
```

Proses konversi sidik jari (2).

```
p = finger.createModel();  
if(p == FINGERPRINT_OK) {  
    ...  
}else if (p == FINGERPRINT_ENROLLMISMATCH){  
    ...  
    return p;  
}
```

Proses pencocokan sidik jari.

```
p = finger.storeModel(id);  
if(p == FINGERPRINT_OK) {  
    ...  
}else{  
    ...  
    return p;  
}
```

Proses menyimpan data sidik jari.

```
void buzz_scan() {  
    for(int x = 0; x < 8 ; x++ ) {  
        ...  
    }  
}
```

Fungsi nada buzzer scan sidik jari.

```
void buzz_notif(){  
    ...  
}
```

Fungsi nada buzzer notifikasi.

```
void buzz_match() {  
    for(int x = 0; x < 2 ; x++ ) {
```

```
    ...  
  }  
}
```

Fungsi nada buzzer sidik jari cocok.

```
void buzz_not_match() {  
    ...  
}
```

Fungsi nada buzzer sidik jari tidak cocok.

```
// Fungsi Nada Buzzer menyimpan ID sidik jari  
void buzz_store_id(){  
    for(int x = 0; x < 2 ; x++ ) {  
        ...  
    }  
    ...  
}
```

Fungsi nada buzzer menyimpan IDE sidik jari.

### Contoh Program Read:

```
// Library Sidik Jari  
#include <Adafruit_Fingerprint.h>  
// Library LCD 16x2 I2C  
#include <Wire.h>  
#include <LiquidCrystal_I2C.h>  
// Library Software Serial  
#include <SoftwareSerial.h>  
  
// PIN Buzzer terhubung ke PIN D3 NodeMCU  
#define BUZZ_PIN D3  
  
// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2  
LiquidCrystal_I2C lcd(0x27,16,2);  
  
// Koneksi PIN Fingerprint ke PIN NodeMCU  
// TX D5  
// RX D6  
// SoftwareSerial objek dengan nama FP_Serial  
SoftwareSerial FP_Serial(D5, D6);  
  
// Fingerprint Objek dengan nama finger  
// dengan koneksi serial FP_Serial  
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&FP_Serial);  
// Variabel untuk menyimpan ID  
int id = 0;  
// Variabel untuk nama ID yang tersimpan  
String id_name = "IoT KIT";  
  
void setup() {  
    // Serial Monitor pada baudrate 115200  
    Serial.begin(115200);  
  
    // Mengatur BUZZ_PIN sebagai Output  
    // Kondisi awal BUZZ_PIN LOW (Buzzer OFF)  
    pinMode(BUZZ_PIN, OUTPUT);
```



```
digitalWrite(BUZZ_PIN, LOW);

// Komunikasi Serial FP dengan baudrate 57600
finger.begin(57600);

lcd.init();           // Inisialisasi LCD
lcd.backlight();     // Menyalakan Backlight LCD
lcd.print("Trainer Kit IoT");
delay(2000);         // Jeda tampilan teks pada LCD
lcd.clear();         // Membersihkan tampilan LCD

// Cek koneksi fingerprint
if(finger.verifyPassword()) {
  lcd.setCursor(0,0); lcd.print("Sensor ditemukan");
  Serial.println("Sensor ditemukan");
  delay(1000);
  lcd.clear();
}else{
  lcd.setCursor(0,0); lcd.print("Tidak Ada Sensor");
  Serial.println("Tidak Ada Sensor");
  delay(1000);
  lcd.clear();
}
}

void loop() {
  // Menampilkan teks pada LCD 16x2 I2C
  lcd.setCursor(0,0); lcd.print(" SELAMAT DATANG ");
  lcd.setCursor(0,1); lcd.print(" SCAN JARI ANDA ");

  // Membaca ID yang tersimpan
  id = getFingerprintIDez();
  delay(50);

  // ID 1 terdaftar menampilkan teks dan notif buzzer
  // dan jika ID tidak terdaftar
  if(id == 1){
    lcd.clear();
    buzz_match();
    lcd.setCursor(0,0); lcd.print("ID TERDAFTAR   ");
    lcd.setCursor(0,1); lcd.print("Hai, " + id_name);
    delay(2000);
    id = 0; lcd.clear();
  }else{
    buzz_not_match();
    lcd.setCursor(0,0); lcd.print("TIDAK TERDAFTAR!");
    lcd.setCursor(0,1); lcd.print("Silahkan Reg. fp");
    delay(2000);
    id = 0; lcd.clear();
  }
}

// Fungsi untuk mencari ID yang tersimpan
int getFingerprintIDez() {
  int p = -1;
```

```
// Memindai sidik jari
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  if(p == FINGERPRINT_OK) {
    Serial.println("Memindai sidik jari");
    buzz_scan();
    break;
  }
}

// Mengonversi sidik jari
p = finger.image2Tz();
if(p == FINGERPRINT_OK) {
  Serial.println("Konversi sidik jari");
} else {
  Serial.println("Konversi bermasalah");
  return p;
}

// Mencari sidik jari yang cocok
p = finger.fingerFastSearch();
if(p == FINGERPRINT_OK) {
  Serial.println("Sidik jari ditemukan");
} else {
  Serial.println("Sidik jari tidak ditemukan");
  return p;
}

// Mencetak ID dengan konfidensi pada serial monitor
// Mengembalikan nilai ID pada finger.fingerID
Serial.print("ID Ditemukan #");
Serial.print(finger.fingerID);
Serial.print(" dengan konfidensi ");
Serial.println(finger.confidence);

return finger.fingerID;
}

// Fungsi Nada Buzzer scan sidik jari
void buzz_scan() {
  for (int x = 0; x < 8 ; x++ ) {
    digitalWrite(BUZZ_PIN, HIGH);
    delay(50);
    digitalWrite(BUZZ_PIN, LOW);
    delay(30);
  }
}

// Fungsi Nada Buzzer sidik jari cocok
void buzz_match() {
  for (int x = 0; x < 2 ; x++ ) {
    digitalWrite(BUZZ_PIN, HIGH);
    delay(200);
    digitalWrite(BUZZ_PIN, LOW);
    delay(100);
  }
}
```

```
}  
  
// Fungsi Nada Buzzer sidik jari tidak cocok  
void buzz_not_match() {  
    digitalWrite(BUZZ_PIN, HIGH);  
    delay(1000);  
    digitalWrite(BUZZ_PIN, LOW);  
    delay(100);  
}
```

### Keterangan Program:

```
#include <Adafruit_Fingerprint.h>
```

*Library* sensor sidik jari.

```
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>
```

*Library* LCD 16x2 I2C.

```
#include <SoftwareSerial.h>
```

*Library* software serial.

```
#define BUZZ_PIN D3
```

Pin buzzer terhubung ke pin D3 NodeMCU.

```
LiquidCrystal_I2C lcd(0x27,16,2);
```

LCD objek.

```
SoftwareSerial FP_Serial(D5, D6);
```

*Software* serial objek dengan nama FP\_serial dengan pin D5 dan D5.

```
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&FP_Serial);
```

*Fingerprint* objek dengan nama *finger* dengan koneksi serial FP\_Serial.

```
int id = 0;
```

Variabel untuk ID yang tersimpan.

```
String id_name = "IoT KIT";
```

Variabel untuk nama ID yang tersimpan.

```
Serial.begin(115200);
```

Serial monitor pada baudrate 115200.

```
pinMode(BUZZ_PIN, OUTPUT);
```

```
digitalWrite(BUZZ_PIN, LOW);
```

Mengatur buzzer sebagai output dengan kondisi awal LOW (Buzzer OFF).

```
finger.begin(57600);
```

komunikasi serial dengan sensor sidik jari pada *baudrate* 57600.

```
lcd.init();
```

```
lcd.backlight();
```

```
lcd.print("Trainer Kit IoT");
```

```
delay(2000);
```

```
lcd.clear();
```

Inisialisasi LCD, menyalakan *backlight* dan mencetak teks dengan jeda 2000ms atau 2 detik kemudian LCD dibersihkan.

```
if(finger.verifyPassword()) {  
    ...  
}else{  
    ...  
}
```

Cek koneksi sensor fingerprint, data ditampilkan pada LCD dan serial monitor.

```
lcd.setCursor(0,0); lcd.print(" SELAMAT DATANG ");  
lcd.setCursor(0,1); lcd.print(" SCAN JARI ANDA ");
```

Menampilkan teks pada LCD 16x2 I2C.

```
id = getFingerprintIDez();  
delay(50);
```

Membaca ID yang tersimpan, menjalankan fungsi getFingerprintDez.

```
if(id == 1){  
    ...  
}else{  
    ...  
}
```

Jika yang terdaftar ID 1 akan menampilkan teks pada LCD dan bunyi buzzer.

```
int p = -1;  
while (p != FINGERPRINT_OK) {  
    ...  
    if(p == FINGERPRINT_OK) {  
        ...  
    }  
}
```

Memindai sidik jari.

```
p = finger.image2Tz();  
if(p == FINGERPRINT_OK) {  
    ...  
}else{  
    ...  
}
```

Mengonversi data sidik jari.

```
p = finger.fingerFastSearch();  
if(p == FINGERPRINT_OK) {  
    ...  
}else{  
    ...  
}
```

Mencari sidik jari yang cocok.

```
Serial.print("ID Ditemukan #");  
...  
Serial.println(finger.confidence);
```

Mencetak ID dengan konfidensi pada serial monitor.

```
void buzz_scan() {  
  for (int x = 0; x < 8 ; x++ ) {  
    ...  
  }  
}
```

Fungsi nada buzzer *scan* sidik jari.

```
void buzz_match() {  
  for (int x = 0; x < 2 ; x++ ) {  
    ...  
  }  
}
```

Fungsi nada buzzer sidik jari cocok.

```
void buzz_not_match() {  
  digitalWrite(BUZZ_PIN, HIGH);  
  delay(1000);  
  digitalWrite(BUZZ_PIN, LOW);  
  delay(100);  
}
```

Fungsi nada buzzer sidik jari tidak cocok.

### Contoh Program Delete:

```
// Library Sidik Jari  
#include <Adafruit_Fingerprint.h>  
// Library LCD 16x2 I2C  
#include <Wire.h>  
#include <LiquidCrystal_I2C.h>  
// Library Software Serial  
#include <SoftwareSerial.h>  
  
// PIN Buzzer terhubung ke PIN D3 NodeMCU  
#define BUZZ_PIN D3  
  
// LCD Objek dengan nama lcd, alamat I2C 0x27, karakter 16x2  
LiquidCrystal_I2C lcd(0x27,16,2);  
  
// Koneksi PIN Fingerprint ke PIN NodeMCU  
// TX D5  
// RX D6  
// SoftwareSerial objek dengan nama FP_Serial  
SoftwareSerial FP_Serial(D5, D6);  
  
// Fingerprint Objek dengan nama finger  
// dengan koneksi serial FP_Serial  
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&FP_Serial);  
// Variabel untuk menyimpan ID  
int id;  
  
void setup() {
```

```
Serial.begin(115200);
// Serial Monitor pada baudrate 115200

// Mengatur BUZZ_PIN sebagai Output
// Kondisi awal BUZZ_PIN LOW (Buzzer OFF)
pinMode(BUZZ_PIN, OUTPUT);
digitalWrite(BUZZ_PIN, LOW);

// Komunikasi Serial FP dengan baudrate 57600
finger.begin(57600);

lcd.init(); // Inisialisasi LCD
lcd.backlight(); // Menyalakan Backlight LCD
lcd.print("Trainer Kit IoT");
delay(2000); // Jeda tampilan teks pada LCD
lcd.clear(); // Perintah untuk membersihkan tampilan LCD

// Cek koneksi fingerprint
if(finger.verifyPassword()) {
  lcd.setCursor(0,0); lcd.print("Sensor ditemukan");
  Serial.println("Sensor ditemukan");
  delay(1000);
  lcd.clear();
}else{
  lcd.setCursor(0,0); lcd.print("Tidak Ada Sensor");
  Serial.println("Tidak Ada Sensor");
  delay(1000);
  lcd.clear();
}
}

// Membaca angka (ID) dari serial monitor
int baca_id() {
  uint8_t num = 0;
  while (num == 0) {
    while(!Serial.available());
    num = Serial.parseInt();
  }
  buzz_notif();
  return num;
}

void loop() {
  // Mencetak teks ke serial monitor dan LCD 16x2 I2C
  Serial.println("Ketik ID (1-127) yang akan dihapus!");
  lcd.setCursor(0,0); lcd.print("Ketik ID (1-127)");
  lcd.setCursor(0,1); lcd.print(" Serial Monitor ");

  // Memabaca ID
  id = baca_id();
  if (id == 0) {
    return;
  }
  Serial.print("Menghapus ID#");
  Serial.println(id);
}
```

```
// Menghapus data fingerprint
// Susuai ID yang dimasukkan
deleteFingerprint(id);
}

// Fungsi menghapus data sidik jari
uint8_t deleteFingerprint(int id){
  int p = -1;
  p = finger.deleteModel(id);
  if(p == FINGERPRINT_OK) {
    Serial.println("Sidik jari dihapus!");
    buzz_deleted();
    lcd.setCursor(0,0); lcd.print("Berhasil dihapus");
    lcd.setCursor(0,1); lcd.print(".....");
    delay(2000);
  }else{
    Serial.println("Sidik jari gagal dihapus");
    buzz_failed();
    lcd.setCursor(0,0); lcd.print("Gagal dihapus  ");
    lcd.setCursor(0,1); lcd.print(".....");
    delay(2000);
  }
  return p;
}

// Fungsi Nada Buzzer notif
void buzz_notif(){
  digitalWrite(BUZZ_PIN, HIGH);
  delay(200);
  digitalWrite(BUZZ_PIN, LOW);
  delay(100);
}

// Fungsi Nada Buzzer menghapus data sidik jari
void buzz_deleted() {
  for(int x = 0; x < 2 ; x++ ) {
    digitalWrite(BUZZ_PIN, HIGH);
    delay(200);
    digitalWrite(BUZZ_PIN, LOW);
    delay(100);
  }
}

// Fungsi Nada Buzzer gagal menghapus data sidik jari
void buzz_failed() {
  digitalWrite(BUZZ_PIN, HIGH);
  delay(1000);
  digitalWrite(BUZZ_PIN, LOW);
  delay(100);
}
```

**Keterangan Program:**

```
#include <Adafruit_Fingerprint.h>
```

*Library* sensor sidik jari.

```
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>
```

Library LCD 16x2 I2C.

```
#include <SoftwareSerial.h>
```

Library software serial.

```
#define BUZZ_PIN D3
```

Pin buzzer terhubung ke pin D3 NodeMCU.

```
LiquidCrystal_I2C lcd(0x27,16,2);
```

LCD objek.

```
SoftwareSerial FP_Serial(D5, D6);
```

Software serial objek dengan nama FP\_serial dengan pin D5 dan D5.

```
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&FP_Serial);
```

Fingerprint objek dengan nama *finger* dengan koneksi serial FP\_Serial.

```
int id;
```

Variabel untuk menyimpan ID.

```
Serial.begin(115200);
```

Serial monitor pada *baudrate* 115200.

```
pinMode(BUZZ_PIN, OUTPUT);
```

```
digitalWrite(BUZZ_PIN, LOW);
```

Mengatur buzzer sebagai output dengan kondisi awal LOW (Buzzer OFF).

```
finger.begin(57600);
```

komunikasi serial dengan sensor sidik jari pada *baudrate* 57600.

```
lcd.init();  
lcd.backlight();  
lcd.print("Trainer Kit IoT");  
delay(2000);  
lcd.clear();
```

Inisialisasi LCD, menyalakan *backlight* dan mencetak teks dengan jeda 2000ms atau 2 detik kemudian LCD dibersihkan.

```
if(finger.verifyPassword()) {  
    ...  
}else{  
    ...  
}
```

Cek koneksi sensor *fingerprint*, data ditampilkan pada LCD dan serial monitor.

```
int baca_id() {  
    uint8_t num = 0;  
    while (num == 0) {  
        while (!Serial.available());  
        num = Serial.parseInt();  
    }  
    buzz_notif();  
    return num;
```



```
}
```

Fungsi membaca angka dari serial monitor dengan nilai balik angka sebagai ID.

```
Serial.println("Ketik ID (1-127) yang akan dihapus!");  
lcd.setCursor(0,0); lcd.print("Ketik ID (1-127)");  
lcd.setCursor(0,1); lcd.print(" Serial Monitor ");
```

Mencetak teks ke serial monitor dan LCD

```
id = baca_id();  
if(id == 0) {  
    return;  
}  
Serial.print("Menghapus ID#");  
Serial.println(id);  
deleteFingerprint(id);
```

Membaca ID, jika ID bernilai 0 akan mengulang proses sampai 'id' tidak bernilai 0. Nilai pada 'id' dicetak pada serial monitor dan menjalankan proses pada fungsi deleteFingerprint.

```
uint8_t deleteFingerprint(int id){  
    int p = -1;  
    p = finger.deleteModel(id);  
    if(p == FINGERPRINT_OK) {  
        ...  
    }else{  
        ...  
    }  
    return p;  
}
```

Fungsi menghapus data sidik jari yang tersimpan.

```
void buzz_done(){  
    ...  
}
```

Fungsi nada buzzer notifikasi.

```
void buzz_deleted() {  
    for(int x = 0; x < 2 ; x++ ) {  
        ...  
    }  
}
```

Fungsi nada buzzer menghapus data sidik jari.

```
void buzz_failed() {  
    ...  
}
```

Fungsi nada buzzer gagal menghapus data sidik jari.



Makassar, Sulawesi Selatan

 ELECTINS | [www.electins.id](http://www.electins.id)

**#Belajar**LebihMudah

