

MODUL AJAR LANJUTAN MIKROKONTROLLER



Disusun oleh:
Nitasya Rahma, Alia Dewi
K, dan Tim CV. Kreasi
Industri Nusantara
2025

Kata Pengantar

Puji syukur ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga penyusunan modul praktikum ini dapat diselesaikan dengan baik. Modul ini disusun sebagai panduan bagi mahasiswa dalam memahami konsep dasar dan penerapan mikrokontroler khususnya dengan menggunakan board ESP32.

Dalam modul ini disajikan berbagai topik praktikum, mulai dari sistem kontrol sederhana, pengolahan sinyal sensor, hingga komunikasi data berbasis Internet of Things (IoT). Harapannya, modul ini dapat membantu mahasiswa dalam menguasai keterampilan pemrograman mikrokontroler sekaligus memperluas wawasan mengenai implementasi teknologi di bidang elektronika dan sistem tertanam.

Penyusun menyadari bahwa modul ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk perbaikan di masa mendatang.

Pada kesempatan ini, penyusun ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan, khususnya kepada Nitasya Rahma, Alia Dewi K. dan Tim Kreasi Industries atas bantuan, semangat, dan kontribusinya dalam penyusunan modul ini.

Semoga modul praktikum ini dapat memberikan manfaat dan menjadi bekal yang berguna bagi mahasiswa dalam memahami serta mengaplikasikan konsep mikrokontroler di berbagai bidang.

Surabaya, 17 September 2025

Penulis

Daftar Isi

Kata Pengantar.....	1
Daftar Isi.....	2
MODUL I.....	4
TRAFFIC LIGHT DENGAN TIMER DAN INTERRUPT DARI PUSH BUTTON.....	4
A. Latar Belakang.....	4
B. Tujuan.....	5
C. Alat dan Bahan.....	5
D. Prosedur.....	5
E. Tugas.....	14
MODUL 2.....	15
KONTROL KECEPATAN MOTOR DENGAN POTENSIOMETER.....	15
A. Latar Belakang.....	15
B. Tujuan.....	16
C. Alat dan Bahan.....	16
D. Prosedur.....	16
E. Tugas.....	19
A. Latar Belakang.....	20
B. Tujuan.....	20
C. Alat dan Bahan.....	20
D. Prosedur.....	21
E. Tugas.....	24
MODUL 4.....	25
MENAMPILKAN PEMBACAAN SENSOR HEART RATE DENGAN OLED.....	25
A. Latar Belakang.....	25
B. Tujuan.....	26
C. Alat dan Bahan.....	26
D. Prosedur.....	26
E. Tugas.....	30
MODUL 5.....	31
MENAMPILKAN HASIL PEMBACAAN SENSOR ULTRASONIC DI LCD 16X2.....	31
A. Latar Belakang.....	31
B. Tujuan.....	32
C. Alat dan Bahan.....	32
D. Prosedur.....	32
E. Tugas.....	35
MODUL 6.....	36
MONITOR SUHU MENGGUNAKAN LCD 16X2 DAN BUZZER SEBAGAI TANDA PERINGATAN.....	36

A. Latar Belakang.....	36
B. Tujuan.....	36
C. Alat dan Bahan.....	37
D. Prosedur.....	37
E. Tugas.....	39
MODUL 7.....	40
KOMUNIKASI CAN BUS ANTAR ESP.....	40
A. Latar Belakang.....	40
B. Tujuan.....	40
C. Alat dan Bahan.....	41
D. Prosedur.....	41
E. Tugas.....	43
MODUL 8.....	44
KOMUNIKASI BLUETOOTH ANTAR ESP.....	44
A. Latar Belakang.....	44
B. Tujuan.....	44
C. Alat dan Bahan.....	45
D. Prosedur.....	45
E. Tugas.....	48
MODUL 9.....	48
PEMBACAAN SUHU DITAMPILKAN KE WEB SERVER ESP32.....	48
A. Latar Belakang.....	48
B. Tujuan.....	49
C. Alat dan Bahan.....	49
D. Prosedur.....	49
E. Tugas.....	53
MODUL 10.....	54
PEMBACAAN SUHU DITAMPILKAN MENGGUNAKAN THINGSPEAK.....	54
A. Latar Belakang.....	54
B. Tujuan.....	54
C. Alat dan Bahan.....	54
D. Prosedur.....	55
E. Tugas.....	57

MODUL I

TRAFFIC LIGHT DENGAN TIMER DAN INTERRUPT DARI PUSH BUTTON

A. Latar Belakang

Sistem lalu lintas merupakan bagian penting dalam kehidupan masyarakat modern. Salah satu komponen utama dari sistem lalu lintas adalah lampu lalu lintas (traffic light) yang berfungsi mengatur pergerakan kendaraan secara teratur dan aman. Dalam implementasinya, traffic light membutuhkan sistem kendali otomatis berbasis waktu, serta sistem input manual seperti tombol pejalan kaki (push button) agar mampu beradaptasi dengan kondisi nyata di lapangan. Oleh karena itu, sistem pengendali mikrokontroler sangat ideal digunakan untuk mengatur logika kerja traffic light yang presisi dan fleksibel.

Dalam praktiknya, mikrokontroler dapat mengatur durasi setiap lampu (merah, kuning, dan hijau) menggunakan timer internal, serta merespons input dari pengguna menggunakan interrupt eksternal. Timer digunakan untuk mengatur perubahan lampu secara otomatis dalam siklus tertentu, sementara interrupt digunakan untuk menangani prioritas input seperti tombol dari pengguna yang ingin menyeberang. Kemampuan mikrokontroler dalam menangani kedua mekanisme ini secara bersamaan menjadikannya sangat relevan dalam sistem tertanam yang memerlukan kendali real-time.

Melalui praktikum ini, mahasiswa akan memahami bagaimana cara merancang dan memprogram sistem traffic light menggunakan mikrokontroler berbasis ESP32 atau sejenisnya, dengan kombinasi logika timer dan interrupt. Mahasiswa juga akan belajar bagaimana suatu sistem dapat menginterupsi siklus utama program ketika suatu kondisi eksternal terpenuhi, serta bagaimana memastikan bahwa sistem tetap berjalan secara efisien tanpa konflik logika. Praktikum ini bertujuan untuk membekali mahasiswa dengan dasar pengendalian otomatis dan responsif yang banyak digunakan dalam dunia industri dan transportasi.

B. Tujuan

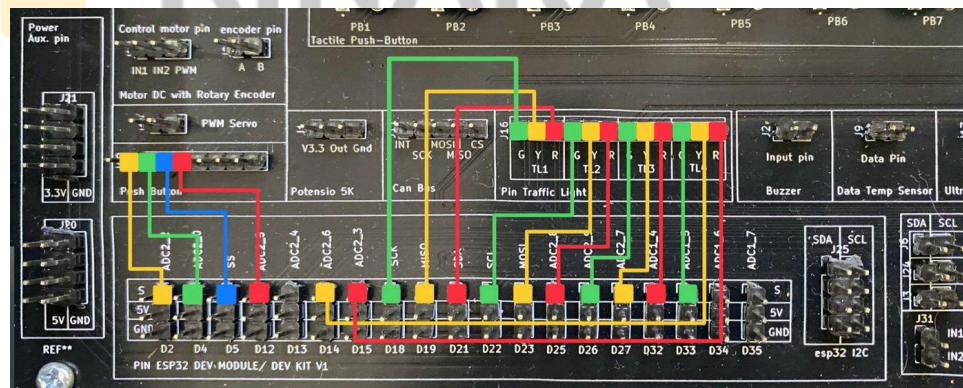
Praktikum ini bertujuan untuk memberikan pemahaman kepada mahasiswa tentang cara mengimplementasikan sistem traffic light berbasis mikrokontroler dengan memanfaatkan fitur timer dan interrupt. Mahasiswa diharapkan mampu merancang logika pengaturan nyala lampu merah, kuning, dan hijau secara berurutan dalam durasi tertentu, serta menambahkan mekanisme interrupt menggunakan push button untuk mengubah alur lampu secara manual saat tombol ditekan.

C. Alat dan Bahan

1. Laptop dengan Arduino IDE
2. Kabel Jumper
3. Kabel USB mikro
4. Modul Ajar Mikrokontroller

D. Prosedur

1. Penyusunan Rangkaian



- Jalur TL1 (Traffic Light 1)
 - a. Hubungkan LED hijau jalur TL1 pada kotak Traffic Light ke pin D18 (GPIO 18) pada baris S di kotak ESP32.
 - b. Hubungkan LED kuning jalur TL1 pada kotak Traffic Light ke pin D19 (GPIO 19) pada baris S di kotak ESP32.
 - c. Hubungkan LED merah jalur TL1 pada kotak Traffic Light ke pin D21 (GPIO 21) pada baris S di kotak ESP32.

- Jalur TL2
 - a. Hubungkan LED hijau jalur TL2 pada kotak Traffic Light ke pin D22 (GPIO 22) pada baris S di kotak ESP32.
 - b. Hubungkan LED kuning jalur TL2 pada kotak Traffic Light ke pin D23 (GPIO 23) pada baris S di kotak ESP32.
 - c. Hubungkan LED merah jalur TL2 pada kotak Traffic Light ke pin D25 (GPIO 25) pada baris S di kotak ESP32.
 - Jalur TL3
 - a. Hubungkan LED hijau jalur TL3 pada kotak Traffic Light ke pin D26 (GPIO 26) pada baris S di kotak ESP32.
 - b. Hubungkan LED kuning jalur TL3 pada kotak Traffic Light ke pin D27 (GPIO 27) pada baris S di kotak ESP32.
 - c. Hubungkan LED merah jalur TL3 pada kotak Traffic Light ke pin D32 (GPIO 32) pada baris S di kotak ESP32.
 - Jalur TL4
 - a. Hubungkan LED hijau jalur TL4 pada kotak Traffic Light ke pin D33 (GPIO 33) pada baris S di kotak ESP32.
 - b. Hubungkan LED kuning jalur TL4 pada kotak Traffic Light ke pin D15 (GPIO 15) pada baris S di kotak ESP32.
 - c. Hubungkan LED merah jalur TL4 pada kotak Traffic Light ke pin D14 (GPIO 14) pada baris S di kotak ESP32.
 - Push Button
 - a. Hubungkan pada kotak push button TL1 ke pin D2 (GPIO 2) pada baris S di kotak ESP32.
 - b. Hubungkan salah satu kaki push button TL2 ke pin D4 (GPIO 4) pada baris S di kotak ESP32.
 - c. Hubungkan salah satu kaki push button TL3 ke pin D5 (GPIO 5) pada baris S di kotak ESP32.
 - d. Hubungkan salah satu kaki push button TL4 ke pin D12 (GPIO 12) pada baris S di kotak ESP32.
2. Persiapan Arduino IDE

- Buka Arduino IDE pada laptop.
- Pastikan board ESP32 sudah terinstal pada Arduino IDE.
- Install library LiquidCrystal_I2C melalui Library Manager.
- Hubungkan ESP32 ke laptop menggunakan kabel USB mikro.

3. Program ESP32

```

1  /* PROGRAM TRAFFIC LIGHT DENGAN INTERRUPT */
2
3  #define GTL1 18
4  #define YTL1 19
5  #define RTL1 21
6  #define GTL2 22
7  #define YTL2 23
8  #define RTL2 25
9  #define GTL3 26
10 #define YTL3 27
11 #define RTL3 32
12 #define GTL4 33
13 #define YTL4 15
14 #define RTL4 14
15 #define PBTL1 2
16 #define PBTL2 4
17 #define PBTL3 5
18 #define PBTL4 12
19 #define PBTL5 13
20 int i;
21 int con;
22 int PB1, PB2, PB3, PB4, PB5;
23
24 volatile bool buttonPressedTL1 = false;
25 volatile bool buttonPressedTL2 = false;
26 volatile bool buttonPressedTL3 = false;
27 volatile bool buttonPressedTL4 = false;
```



```
29 // simpan jalur yang sedang hijau
30 int currentGreen = 1;
31
32 // TRAFFIC LIGHT SETUP
33 void setTL1(bool G, bool Y, bool R) {
34     digitalWrite(GTL1, G);
35     digitalWrite(YTL1, Y);
36     digitalWrite(RTL1, R);
37 }
38 void setTL2(bool G, bool Y, bool R) {
39     digitalWrite(GTL2, G);
40     digitalWrite(YTL2, Y);
41     digitalWrite(RTL2, R);
42 }
43 void setTL3(bool G, bool Y, bool R) {
44     digitalWrite(GTL3, G);
45     digitalWrite(YTL3, Y);
46     digitalWrite(RTL3, R);
47 }
48 void setTL4(bool G, bool Y, bool R) {
49     digitalWrite(GTL4, G);
50     digitalWrite(YTL4, Y);
51     digitalWrite(RTL4, R);
52 }
```



```

54 // helper all-red dan kedip kuning dua jalur (untuk target dan current)
55 void allRed(){
56     setTL1(LOW,LOW,HIGH);
57     setTL2(LOW,LOW,HIGH);
58     setTL3(LOW,LOW,HIGH);
59     setTL4(LOW,LOW,HIGH);
60 }
61 void blinkYellow(int a, int b, int times=3){
62     for(int k=0;k<times;k++){
63         if(a==1) setTL1(LOW,HIGH,LOW);
64         if(a==2) setTL2(LOW,HIGH,LOW);
65         if(a==3) setTL3(LOW,HIGH,LOW);
66         if(a==4) setTL4(LOW,HIGH,LOW);
67
68         if(b==1) setTL1(LOW,HIGH,LOW);
69         if(b==2) setTL2(LOW,HIGH,LOW);
70         if(b==3) setTL3(LOW,HIGH,LOW);
71         if(b==4) setTL4(LOW,HIGH,LOW);
72
73         delay(500);
74         setTL1(LOW,HIGH,LOW);
75         setTL2(LOW,HIGH,LOW);
76         setTL3(LOW,HIGH,LOW);
77         setTL4(LOW,HIGH,LOW);
78         delay(500);
79     }
80 }

```

IMPLEMENTASI

```

82 void onButtonPressTL1() { buttonPressedTL1 = true; }
83 void onButtonPressTL2() { buttonPressedTL2 = true; }
84 void onButtonPressTL3() { buttonPressedTL3 = true; }
85 void onButtonPressTL4() { buttonPressedTL4 = true; }
86
87 // TL1 Run. Kedipkan kuning TL1 + currentGreen, lalu set hijau TL1
88 void TL1Run(){
89     blinkYellow(1, currentGreen);
90     allRed();
91     setTL1(HIGH,LOW,LOW);
92     setTL2(LOW,LOW,HIGH);
93     setTL3(LOW,LOW,HIGH);
94     setTL4(LOW,LOW,HIGH);
95     currentGreen = 1;
96     buttonPressedTL1 = false;
97     delay(3000);
98 }

```

```
100 // TL2Run
101 void TL2Run(){
102     blinkYellow(2, currentGreen);
103     allRed();
104     setTL1(LOW,LOW,HIGH);
105     setTL2(HIGH,LOW,LOW);
106     setTL3(LOW,LOW,HIGH);
107     setTL4(LOW,LOW,HIGH);
108     currentGreen = 2;
109     buttonPressedTL2 = false;
110     delay(3000);
111 }
112
113 // TL3Run
114 void TL3Run(){
115     blinkYellow(3, currentGreen);
116     allRed();
117     setTL1(LOW,LOW,HIGH);
118     setTL2(LOW,LOW,HIGH);
119     setTL3(HIGH,LOW,LOW);
120     setTL4(LOW,LOW,HIGH);
121     currentGreen = 3;
122     buttonPressedTL3 = false;
123     delay(3000);
124 }
```

```
126 // TL4Run
127 void TL4Run(){
128     blinkYellow(4, currentGreen);
129     allRed();
130     setTL1(LOW,LOW,HIGH);
131     setTL2(LOW,LOW,HIGH);
132     setTL3(LOW,LOW,HIGH);
133     setTL4(HIGH,LOW,LOW);
134     currentGreen = 4;
135     buttonPressedTL4 = false;
136     delay(3000);
137 }
```

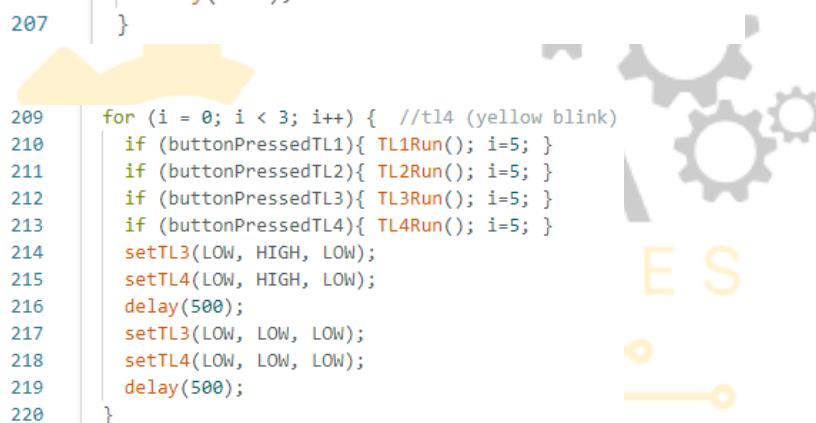
```
139 void normalRun(){
140     for (i = 0; i < 3; i++) { //tl1 (yellow blink)
141         if (buttonPressedTL1){ TL1Run(); i=5; }
142         if (buttonPressedTL2){ TL2Run(); i=5; }
143         if (buttonPressedTL3){ TL3Run(); i=5; }
144         if (buttonPressedTL4){ TL4Run(); i=5; }
145         setTL1(LOW, HIGH, LOW);
146         setTL4(LOW, HIGH, LOW);
147         delay(500);
148         setTL1(LOW, LOW, LOW);
149         setTL4(LOW, LOW, LOW);
150         delay(500);
151     }
152
153     // fase TL1 hijau
154     currentGreen = 1;
155     for (i = 0; i < 5; i++) { //tl1 green
156         setTL1(HIGH, LOW, LOW);
157         setTL2(LOW, LOW, HIGH);
158         setTL3(LOW, LOW, HIGH);
159         setTL4(LOW, LOW, HIGH);
160         delay(1000);
161     }
162
163     for (i = 0; i < 3; i++) { //tl2 (yellow blink)
164         if (buttonPressedTL1){ TL1Run(); i=5; }
165         if (buttonPressedTL2){ TL2Run(); i=5; }
166         if (buttonPressedTL3){ TL3Run(); i=5; }
167         if (buttonPressedTL4){ TL4Run(); i=5; }
168         setTL1(LOW, HIGH, LOW);
169         setTL2(LOW, HIGH, LOW);
170         delay(500);
171         setTL1(LOW, LOW, LOW);
172         setTL2(LOW, LOW, LOW);
173         delay(500);
174     }
175
176     // fase TL2 hijau
177     currentGreen = 2;
178     for (i = 0; i < 5; i++) { //tl2 green
179         setTL1(LOW, LOW, HIGH);
180         setTL2(HIGH, LOW, LOW);
181         setTL3(LOW, LOW, HIGH);
182         setTL4(LOW, LOW, HIGH);
183         delay(1000);
184     }
```

```

186   for (i = 0; i < 3; i++) { //tl3 (yellow blink)
187     if (buttonPressedTL1){ TL1Run(); i=5; }
188     if (buttonPressedTL2){ TL2Run(); i=5; }
189     if (buttonPressedTL3){ TL3Run(); i=5; }
190     if (buttonPressedTL4){ TL4Run(); i=5; }
191     setTL2(LOW, HIGH, LOW);
192     setTL3(LOW, HIGH, LOW);
193     delay(500);
194     setTL2(LOW, LOW, LOW);
195     setTL3(LOW, LOW, LOW);
196     delay(500);
197   }
198
199 // fase TL3 hijau
200 currentGreen = 3;
201 for (i = 0; i < 5; i++) { //tl3 green
202   setTL1(LOW, LOW, HIGH);
203   setTL2(LOW, LOW, HIGH);
204   setTL3(HIGH, LOW, LOW);
205   setTL4(LOW, LOW, HIGH);
206   delay(1000);
207 }

209 for (i = 0; i < 3; i++) { //tl4 (yellow blink)
210   if (buttonPressedTL1){ TL1Run(); i=5; }
211   if (buttonPressedTL2){ TL2Run(); i=5; }
212   if (buttonPressedTL3){ TL3Run(); i=5; }
213   if (buttonPressedTL4){ TL4Run(); i=5; }
214   setTL3(LOW, HIGH, LOW);
215   setTL4(LOW, HIGH, LOW);
216   delay(500);
217   setTL3(LOW, LOW, LOW);
218   setTL4(LOW, LOW, LOW);
219   delay(500);
220 }
221
222 // fase TL4 hijau
223 currentGreen = 4;
224 for (i = 0; i < 5; i++) { //tl4 green
225   setTL1(LOW, LOW, HIGH);
226   setTL2(LOW, LOW, HIGH);
227   setTL3(LOW, LOW, HIGH);
228   setTL4(HIGH, LOW, LOW);
229   delay(1000);
230 }
231
232 }

```



```

234 void setup() {
235     pinMode(GTL1, OUTPUT);
236     pinMode(YTL1, OUTPUT);
237     pinMode(RTL1, OUTPUT);
238     pinMode(GTL2, OUTPUT);
239     pinMode(YTL2, OUTPUT);
240     pinMode(RTL2, OUTPUT);
241     pinMode(GTL3, OUTPUT);
242     pinMode(YTL3, OUTPUT);
243     pinMode(RTL3, OUTPUT);
244     pinMode(GTL4, OUTPUT);
245     pinMode(YTL4, OUTPUT);
246     pinMode(RTL4, OUTPUT);
247
248     pinMode(PBTL1, INPUT_PULLUP);
249     pinMode(PBTL2, INPUT_PULLUP);
250     pinMode(PBTL3, INPUT_PULLUP);
251     pinMode(PBTL4, INPUT_PULLUP);
252     pinMode(PBTL5, INPUT_PULLUP);
253
254     attachInterrupt(digitalPinToInterrupt(PBTL1), onButtonPressTL1, FALLING);
255     attachInterrupt(digitalPinToInterrupt(PBTL2), onButtonPressTL2, FALLING);
256     attachInterrupt(digitalPinToInterrupt(PBTL3), onButtonPressTL3, FALLING);
257     attachInterrupt(digitalPinToInterrupt(PBTL4), onButtonPressTL4, FALLING);
258     con = 0;
259 }

```



```

261 void loop() {
262     // tambahkan return agar tidak jatuh ke normalRun
263     if (buttonPressedTL1){ TL1Run(); return; }
264     if (buttonPressedTL2){ TL2Run(); return; }
265     if (buttonPressedTL3){ TL3Run(); return; }
266     if (buttonPressedTL4){ TL4Run(); return; }
267     else {
268         | normalRun();
269     }
270 }

```

4. Pengujian Hasil

- Mulai dan amati siklus normal traffic light berjalan dari TL1 ke TL4 secara otomatis.
- Tekan salah satu push button (misalnya TL3), dan perhatikan apakah sistem segera memberi prioritas ke jalur yang ditekan.
- Ulangi pengujian dengan menekan tombol lain untuk memastikan seluruh interupsi berjalan dengan baik.

- Jika hasil tidak sesuai, cek ulang koneksi kabel dan pastikan program tidak mengalami kesalahan logika.

E. Tugas

Buatlah sistem penyeberangan pejalan kaki. Lampu merah menyala ketika pejalan menekan tombol, lalu kembali ke siklus normal setelah beberapa detik.



MODUL 2

KONTROL KECEPATAN MOTOR DENGAN POTENSIOMETER

A. Latar Belakang

Motor DC merupakan salah satu aktuator yang paling banyak digunakan dalam berbagai aplikasi elektronika dan sistem kendali. Mulai dari kipas, mainan anak, robotika, conveyor, hingga perangkat rumah tangga modern, motor DC memegang peranan penting sebagai penggerak. Namun, dalam implementasi nyata, tidak semua aplikasi membutuhkan kecepatan motor yang konstan. Ada kalanya motor harus berputar dengan kecepatan rendah, sedang, maupun tinggi sesuai dengan kebutuhan sistem. Oleh karena itu, diperlukan sebuah metode untuk mengontrol kecepatan motor secara fleksibel dan efisien.

Salah satu teknik yang banyak digunakan untuk mengatur kecepatan motor DC adalah Pulse Width Modulation (PWM). PWM bekerja dengan cara mengatur lebar pulsa sinyal digital yang dikirimkan ke motor. Meskipun motor menerima tegangan penuh pada setiap pulsa, efek rata-rata tegangan yang diterima akan bervariasi sesuai dengan duty cycle PWM. Duty cycle yang kecil menghasilkan putaran lambat, sedangkan duty cycle yang besar menghasilkan putaran lebih cepat. Teknik ini terbukti efektif karena selain sederhana, juga efisien dalam penggunaan energi.

Dalam praktikum ini, digunakan potensiometer sebagai input analog yang mewakili perintah dari pengguna. Dengan memutar potensiometer, nilai resistansi berubah sehingga menghasilkan variasi tegangan yang dibaca oleh pin ADC (Analog to Digital Converter) pada ESP32. Nilai tegangan tersebut kemudian dikonversi menjadi nilai digital (0–4095) yang dapat dipetakan ke dalam rentang PWM (0–255). Dengan demikian, potensiometer bertindak sebagai “kendali manual” untuk mengatur kecepatan motor.

Selain itu, sistem juga dilengkapi dengan LCD berbasis I2C sebagai media tampilan. LCD digunakan untuk menampilkan nilai pembacaan potensiometer sekaligus nilai PWM yang dikirim ke motor. Hal ini bertujuan agar pengguna dapat memantau

secara langsung keterkaitan antara input analog, duty cycle PWM, dan respon kecepatan motor.

Penggunaan ESP32 dalam praktikum ini dipilih karena mikrokontroler tersebut memiliki keunggulan dalam menghasilkan sinyal PWM dengan frekuensi dan resolusi yang dapat diatur secara fleksibel. Hal ini memudahkan dalam melakukan kontrol motor yang presisi serta membuka peluang pengembangan ke aplikasi yang lebih kompleks, seperti kontrol berbasis sensor atau sistem kendali otomatis.

Dengan memahami konsep dasar ini, mahasiswa tidak hanya berlatih dalam merangkai perangkat keras dan menulis program, tetapi juga mendapatkan wawasan tentang prinsip kerja sistem kendali motor. Pemahaman ini sangat penting karena menjadi dasar bagi pengembangan teknologi lebih lanjut di bidang robotika, otomasi industri, hingga sistem embedded cerdas.

B. Tujuan

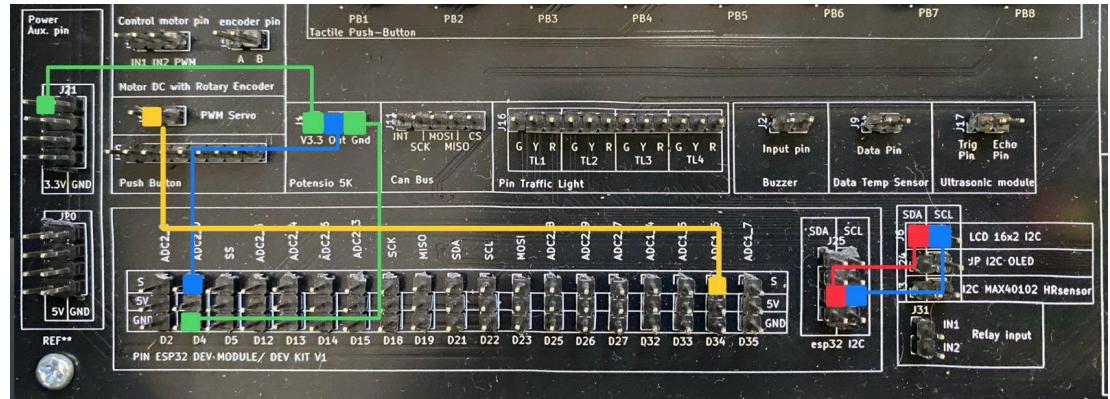
Tujuan praktikum ini adalah agar mahasiswa memahami cara mengontrol kecepatan motor DC menggunakan potensiometer sebagai input analog dan sinyal PWM dari ESP32. Melalui percobaan ini, mahasiswa dapat melihat hubungan antara perubahan nilai potensiometer, duty cycle PWM, dan kecepatan motor yang ditampilkan pada LCD.

C. Alat dan Bahan

1. Laptop dengan Arduino IDE
2. Motor DC
3. Potensiometer
4. Driver Motor
5. LCD 16x2 I2C
6. Kabel Jumper
7. Kabel USB mikro

D. Prosedur

1. Penyusunan Rangkaian



- Hubungkan kaki tengah potensiometer ke pin analog ESP32 (misalnya D4).
 - Hubungkan dua kaki samping potensiometer ke VCC (3.3V) dan GND.
 - Hubungkan motor DC ke pin output ESP32 melalui driver motor.
 - Hubungkan LCD 16x2 ke jalur I2C ESP32 (SDA dan SCL).
 - Pastikan semua kabel jumper terpasang dengan benar.
2. Persiapan Arduino IDE
- Buka Arduino IDE pada laptop.
 - Pastikan board ESP32 sudah terinstal pada Arduino IDE.
 - Install library LiquidCrystal_I2C melalui Library Manager.
 - Hubungkan ESP32 ke laptop menggunakan kabel USB mikro.
3. Program ESP32

```
1 #include <Wire.h>
2 #include <LiquidCrystal_I2C.h>
3
4 // inisialisasi LCD (alamat I2C bisa 0x27 atau 0x3F)
5 LiquidCrystal_I2C lcd(0x27, 16, 2);
6 int pos;
7 const int potPin = 4;          // pin analog ESP32 (potensio tengah)
8 const int motorPin = 15;       // pin PWM ke motor driver
9 const int PWM_CH = 0;          // channel PWM (0 - 7)
10
11 int potValue = 0;
12 int Servalue = 0;
13
14 void setup() {
15     // setup LCD
16     lcd.init();
17     lcd.backlight();
18
19     // setup PWM motor (core 3.x pakai ini, bukan ledcSetup)
20     ledcAttachChannel(motorPin, 100, 8, PWM_CH);
21     // parameternya: pin, frekuensi (Hz), resolusi (bit), channel
22
23     // tampilan awal
24     lcd.setCursor(0,0);
25     lcd.print("ESP32 Motor PWM");
26     delay(1500);
27     lcd.clear();
28 }
```



```

29
30 void loop() {
31     //baca potensio (0 - 4095)
32     potValue = analogRead(potPin);
33
34     // ubah ke skala PWM (0 - 255)
35     Servalue = map(potValue, 0, 4095, 0, 255);
36
37     // tulis PWM ke motor
38     ledcWriteChannel(PWM_CH, Servalue);
39
40     // tampilkan di LCD
41     lcd.setCursor(0,0);
42     lcd.print("Pot: ");
43     lcd.print(potValue);
44     lcd.print("    ");
45
46     lcd.setCursor(0,1);
47     lcd.print("PWM: ");
48     lcd.print(Servalue);
49     lcd.print("    ");
50

```

4. Pengujian Hasil

- Putar potensiometer secara perlahan.
- Amati perubahan kecepatan motor sesuai dengan nilai potensiometer.
- Perhatikan tampilan nilai potensiometer dan PWM yang muncul di LCD 16x2.

E. Tugas

Gunakan potensiometer untuk mengatur intensitas kecerahan LED dengan PWM, bukan motor. Tambahkan batas minimal & maksimal intensitas.

MODUL 3

KONTROL SUDUT SERVO DENGAN IN-OUT DARI ROTARY ENCODER

A. Latar Belakang

Servo merupakan aktuator yang banyak digunakan dalam bidang robotika, otomasi, dan sistem kendali karena mampu bergerak dengan presisi pada sudut tertentu. Untuk mengontrol sudut servo, dibutuhkan sebuah input yang dapat mengatur posisi secara bertahap dan akurat. Salah satu komponen yang umum digunakan adalah rotary encoder. Rotary encoder berfungsi sebagai sensor putaran yang mengubah gerakan mekanik (putar kanan/kiri) menjadi sinyal digital. Dengan memanfaatkan encoder, sudut servo dapat dikendalikan secara interaktif sesuai arah dan jumlah putaran yang dilakukan.

ESP32 dipilih sebagai pengendali utama karena mendukung pembacaan input digital berkecepatan tinggi sekaligus mampu mengontrol servo melalui sinyal PWM. Ditambah dengan penggunaan LCD I2C, mahasiswa dapat memantau sudut servo secara real time, baik dalam bentuk angka derajat maupun tampilan grafis progress bar. Praktikum ini melatih pemahaman mahasiswa mengenai hubungan input–proses–output pada sistem kendali sederhana serta menjadi dasar bagi implementasi sistem kontrol yang lebih kompleks.

B. Tujuan

Praktikum ini bertujuan agar mahasiswa memahami prinsip kerja rotary encoder sebagai input digital serta penerapannya untuk mengontrol sudut servo menggunakan sinyal PWM dari ESP32. Selain itu, mahasiswa diharapkan mampu menampilkan informasi sudut servo pada LCD I2C sehingga dapat memantau hubungan antara putaran encoder dan pergerakan servo secara langsung.

C. Alat dan Bahannya

1. Laptop dengan Arduino IDE
2. ESP32
3. Servo motor
4. Rotary encoder

5. LCD 16x2 I2C
6. Kabel jumper
7. Kabel micro USB

D. Prosedur

1. Penyusunan Rangkaian
 - Hubungkan pin CLK rotary encoder ke pin digital mikrokontroler (misalnya D2).
 - Hubungkan pin DT rotary encoder ke pin digital mikrokontroler (misalnya D3).
 - Hubungkan pin SW rotary encoder ke pin digital mikrokontroler (opsional, jika digunakan).
 - Hubungkan pin VCC dan GND rotary encoder ke 5V dan GND mikrokontroler.
 - Hubungkan pin PWM servo ke pin digital mikrokontroler (misalnya D9).
 - Hubungkan pin VCC servo ke 5V (atau sumber eksternal sesuai kebutuhan servo).
 - Hubungkan pin GND servo ke GND yang sama dengan mikrokontroler.
 - Pastikan semua koneksi jumper terpasang dengan benar.
2. Persiapan Arduino IDE
 - Buka Arduino IDE pada laptop.
 - Pilih board sesuai mikrokontroler yang digunakan (ESP32 atau Arduino Uno).
 - Hubungkan mikrokontroler dengan laptop menggunakan kabel USB.
 - Pastikan library Servo.h sudah terinstal pada Arduino IDE.
3. Program ESP32

```

1 #include <ESP32Servo.h>
2 #include <LiquidCrystal_I2C.h>
3
4 // Pin Encoder
5 #define ENCODER_A 19
6 #define ENCODER_B 18
7
8 // Pin Servo
9 #define SERVO_PIN 13
10
11 // LCD 16x2 (alamat bisa 0x27 atau 0x3F)
12 LiquidCrystal_I2C lcd(0x27, 16, 2);
13
14 // Servo object
15 Servo myServo;
16
17 // Variabel kontrol
18 int angle = 90;           // posisi awal servo
19 int lastStateA;          // simpan state terakhir
20 unsigned long lastLCDUpdate = 0;
21
22 void setup() {
23     // Setup encoder
24     pinMode(ENCODER_A, INPUT);
25     pinMode(ENCODER_B, INPUT);
26
27     // Setup servo
28     myServo.attach(SERVO_PIN);
29     myServo.write(angle);
30
31     // Setup LCD
32     lcd.init();
33     lcd.backlight();
34     lcd.clear();
35
36     // Serial Monitor
37     Serial.begin(115200);
38
39     lastStateA = digitalRead(ENCODER_A);
40
41     // Tampilan awal
42     lcd.setCursor(0, 0);
43     lcd.print("SERVO CONTROL");
44     updateLCD();
45 }
46

```



```

47 void loop() {
48     int currentStateA = digitalRead(ENCODER_A);
49
50     // Deteksi perubahan encoder
51     if (currentStateA != lastStateA) {
52         if (digitalRead(ENCODER_B) != currentStateA) {
53             angle += 1; // putar kanan
54             Serial.println("Putar kanan");
55         } else {
56             angle -= 1; // putar kiri
57             Serial.println("Putar kiri");
58         }
59
60         // Batasi sudut servo
61         angle = constrain(angle, 0, 180);
62
63         // Gerakkan servo
64         myServo.write(angle);
65
66         // Update LCD
67         updateLCD();
68     }
69
70     lastStateA = currentStateA;
71
72     // Update LCD tiap 200 ms (opsional)
73     if (millis() - lastLCDUpdate > 200) {
74         updateLCD();
75         lastLCDUpdate = millis();
76     }
77 }

```

4. Pengujian Hasil

- Pastikan rangkaian telah tersusun dengan benar sesuai skema.
- Upload program ke ESP32 melalui Arduino IDE.
- Amati LCD 16x2, pada baris pertama akan tampil nilai sudut servo dalam derajat.
- Putar rotary encoder
- Amati gerakan fisik servo apakah sesuai dengan sudut yang tampil di LCD.
- Pastikan progress bar pada baris kedua LCD ikut bergerak mengikuti perubahan sudut.
- Dokumentasikan hasil pengujian berupa foto atau video gerakan servo dan tampilan LCD.
-

E. Tugas

Buat menu pemilihan sudut preset (0° , 90° , 180°) dengan encoder sebagai navigator. Tekan tombol encoder untuk memilih sudut, servo bergerak otomatis ke sudut tersebut.



MODUL 4

MENAMPILKAN PEMBACAAN SENSOR HEART RATE DENGAN OLED

A. Latar Belakang

Di era modern saat ini, pemantauan kesehatan secara real-time menjadi kebutuhan penting dalam dunia medis dan gaya hidup. Salah satu indikator penting dalam pemantauan kesehatan adalah detak jantung (heart rate), yang menunjukkan seberapa cepat jantung memompa darah ke seluruh tubuh. Data ini sangat krusial karena bisa mencerminkan kondisi stres, kelelahan, bahkan potensi gangguan jantung pada seseorang. Perangkat wearable seperti smartwatch dan fitness tracker telah memanfaatkan teknologi sensor optik untuk mengukur heart rate secara terus-menerus, sehingga teknologi ini perlu dipahami sejak dini oleh mahasiswa teknik, khususnya dalam bidang biomedis dan IoT.

Sensor detak jantung seperti MAX30102 bekerja berdasarkan prinsip fotopletismografi (PPG), yaitu dengan memancarkan cahaya ke kulit dan mengukur perubahan intensitas pantulan cahaya akibat aliran darah. Sinyal ini kemudian diolah menjadi nilai denyut per menit (BPM). Untuk bisa mengakses dan memahami data ini secara langsung, diperlukan sistem mikrokontroler seperti ESP32 atau Arduino, serta media output berupa layar OLED kecil yang dapat menampilkan nilai detak jantung secara real-time. Kombinasi antara sensor, mikrokontroler, dan tampilan visual membentuk sistem monitoring sederhana yang sangat berguna untuk pembelajaran dan prototipe perangkat medis portabel.

Melalui praktikum ini, mahasiswa diharapkan tidak hanya memahami cara kerja sensor dan rangkaian elektronika, tetapi juga mampu merancang sistem monitoring biomedical sederhana yang dapat diintegrasikan dalam aplikasi nyata. Pengalaman langsung dalam menghubungkan sensor dengan mikrokontroler dan menampilkan data ke OLED akan memberikan pemahaman menyeluruh dari sisi perangkat keras, komunikasi data (I2C), hingga pemrograman dasar. Praktikum ini sekaligus menjadi langkah awal

mahasiswa untuk mengembangkan sistem IoT berbasis kesehatan yang terukur, portabel, dan berpotensi dikembangkan lebih lanjut menjadi alat medis komersial atau riset klinis.

B. Tujuan

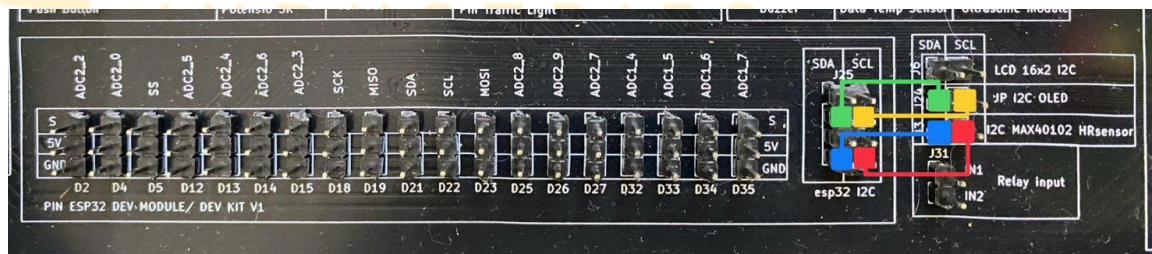
Mahasiswa diharapkan mampu menghubungkan sensor dan layar OLED ke mikrokontroler dengan benar, serta membuat program untuk membaca dan menampilkan data detak jantung (BPM) secara real-time.

C. Alat dan Bahan

1. Laptop dengan Arduino IDE
2. Kabel jumper
3. Sensor Heart Rate and Oxymeter Sensor MAX30102
4. Layar OLED 0.96 inch (I2C, 128x64)
5. Modul Ajar Mikrocontroller

D. Prosedur

1. Penyusunan Rangkaian



- Hubungkan pin SDA pada kotak I2C OLED ke pin SDA kotak I2C pada ESP32.
- Hubungkan pin SCL pada kotak I2C OLED ke pin SCL kotak I2C pada ESP32.
- Hubungkan pin SDA pada sensor MAX30102 ke pin SDA kotak I2C pada ESP32.
- Hubungkan pin SCL pada sensor MAX30102 ke pin SCL kotak I2C pada ESP32.
- Pastikan koneksi jumper terpasang dengan benar.

2. Persiapan Arduino IDE

- Buka Arduino IDE pada laptop.
- Pastikan board ESP32 sudah terpasang di Arduino IDE.
- Hubungkan ESP32 dengan laptop menggunakan kabel USB mikro.
- Unduh library Adafruit_SSD1306, Adafruit_GFX, dan MAX30105 sudah terinstal.

3. Program ESP32

```
1 #include <Wire.h>
2 #include "MAX30105.h"
3 #include "heartRate.h"
4 #include <Adafruit_GFX.h>
5 #include <Adafruit_SSD1306.h>
6
7 // Konfigurasi OLED
8 #define SCREEN_WIDTH 128
9 #define SCREEN_HEIGHT 64
10 #define OLED_RESET -1
11 #define SCREEN_ADDRESS 0x3C
12 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
13
14 MAX30105 particleSensor;
15
16 const byte RATE_SIZE = 4;
17 byte rates[RATE_SIZE];
18 byte rateSpot = 0;
19 long lastBeat = 0;
20
21 float beatsPerMinute;
22 int beatAvg;
```

```

24   void setup()
25  {
26     Serial.begin(115200);
27     Serial.println("Initializing...");
28
29     // Inisialisasi OLED
30     if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
31       Serial.println(F("SSD1306 allocation failed"));
32       for(;;);
33     }
34
35     // Tampilkan pesan awal di OLED
36     display.display();
37     delay(2000); // Pause for 2 seconds
38     display.clearDisplay();
39
40     // Initialize sensor
41     if (!particleSensor.begin(Wire, I2C_SPEED_FAST))
42     {
43       Serial.println("MAX30105 was not found. Please check wiring/power. ");
44       display.clearDisplay();
45       display.setTextSize(1);
46       display.setTextColor(SSD1306_WHITE);
47       display.setCursor(0,0);
48       display.println("Sensor not found!");
49       display.setCursor(0,20);
50       display.println("Check wiring");
51       display.display();
52       while (1);
53     }
54     particleSensor.setup(); //Configure sensor with default settings
55     particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate sensor is running
56     particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
57   }
58
59   void loop()
60  {
61   long irValue = particleSensor.getIR();
62
63   if (checkForBeat(irValue) == true)
64   {
65     //We sensed a beat!
66     long delta = millis() - lastBeat;
67     lastBeat = millis();
68
69     beatsPerMinute = 60 / (delta / 1000.0);
70
71     if (beatsPerMinute < 255 && beatsPerMinute > 20)
72     {
73       rates[rateSpot++] = (byte)beatsPerMinute;
74       rateSpot %= RATE_SIZE;
75
76       //Take average of readings
77       beatAvg = 0;
78       for (byte x = 0 ; x < RATE_SIZE ; x++)
79       {
80         beatAvg += rates[x];
81       }
82     }
83   }

```

```

84 // Tampilkan di Serial Monitor
85 Serial.print("IR=");
86 Serial.print(irValue);
87 Serial.print(", BPM=");
88 Serial.print(beatsPerMinute);
89 Serial.print(", Avg BPM=");
90 Serial.print(beatAvg);
91
92 if (irValue < 50000)
| Serial.print(" No finger?");
93
94 Serial.println();
95
96 // Tampilkan di OLED
97 display.clearDisplay();
98 display.setTextSize(1);
99 display.setTextColor(SSD1306_WHITE);
100
101 // Header
102 display.setCursor(0, 0);
103 display.println("Heart Rate Monitor");
104
105
106 // Garis pemisah
107 display.drawLine(0, 10, 128, 10, SSD1306_WHITE);
108
109 // Nilai BPM saat ini
110 display.setTextSize(2);
111 display.setCursor(0, 15);
112 display.print("BPM: ");
113 if (beatsPerMinute > 0) {
114 | display.print(beatsPerMinute);
115 } else {
116 | display.print("--");
117 }
118
119 // Nilai rata-rata BPM
120 display.setTextSize(1);
121 display.setCursor(0, 40);
122 display.print("Avg BPM: ");
123 if (beatAvg > 0) {
124 | display.print(beatAvg);
125 } else {
126 | display.print("--");
127 }
128
129 // Status jari
130 display.setTextSize(1);
131 display.setCursor(0, 55);
132 if (irValue < 50000) {
133 | display.print("No Finger");
134 } else {
135 | display.print("Finger Detected");
136 }
137
138 display.display();
139 delay(100); // Delay untuk stabilitas pembacaan
140 }

```



E. Tugas

Buat alarm buzzer jika nilai BPM < 60 (bradikardi) atau > 120 (takikardi). Data tetap ditampilkan di OLED.



MODUL 5

MENAMPILKAN HASIL PEMBACAAN SENSOR ULTRASONIC DI LCD 16X2

A. Latar Belakang

Sensor ultrasonik adalah perangkat yang bekerja dengan prinsip pantulan gelombang suara berfrekuensi tinggi, yaitu di atas 20 kHz, yang tidak dapat didengar oleh telinga manusia. Prinsip kerja sensor ini sederhana: transmitter memancarkan gelombang ultrasonik ke arah objek, lalu receiver menangkap pantulan gelombang tersebut. Waktu tempuh gelombang digunakan untuk menghitung jarak objek terhadap sensor. Dengan metode ini, jarak dapat diukur tanpa kontak fisik, sehingga sensor ultrasonik banyak dimanfaatkan dalam bidang robotika, otomasi industri, hingga perangkat keamanan.

Salah satu jenis sensor ultrasonik yang populer adalah HC-SR04, yang memiliki dua pin utama, yaitu Trigger dan Echo. Pin Trigger berfungsi mengirimkan pulsa ultrasonik, sedangkan pin Echo digunakan untuk menerima pantulannya. Dengan mengukur durasi pantulan, jarak objek dapat dihitung menggunakan persamaan:

$$\text{Jarak (cm)} = \frac{\text{waktu} \times 0.034}{2}$$

Pada praktikum ini, sensor ultrasonik dihubungkan ke mikrokontroler ESP32 sebagai pengolah data. ESP32 dipilih karena memiliki kecepatan pemrosesan yang tinggi, koneksi lengkap, serta mendukung berbagai jenis komunikasi, termasuk dengan perangkat display. Hasil pembacaan jarak dari sensor kemudian ditampilkan pada LCD 16x2 yang terhubung melalui komunikasi I2C. LCD ini dipilih karena mampu menampilkan informasi secara langsung dalam bentuk teks, praktis digunakan, dan hanya membutuhkan dua pin (SDA dan SCL) sehingga lebih efisien dibandingkan LCD paralel.

Implementasi menampilkan hasil pembacaan sensor ke LCD penting karena mahasiswa dapat melihat secara nyata proses aliran data dari sensor → mikrokontroler → media output. Hal ini melatih pemahaman mengenai integrasi perangkat keras dalam sistem embedded, yang merupakan salah satu kompetensi utama dalam bidang teknik

elektro dan teknik biomedis. Selain itu, mahasiswa juga dapat menganalisis keterbatasan sensor ultrasonik, misalnya pada objek yang permukaannya menyerap suara, posisi miring, atau adanya gangguan dari lingkungan sekitar.

B. Tujuan

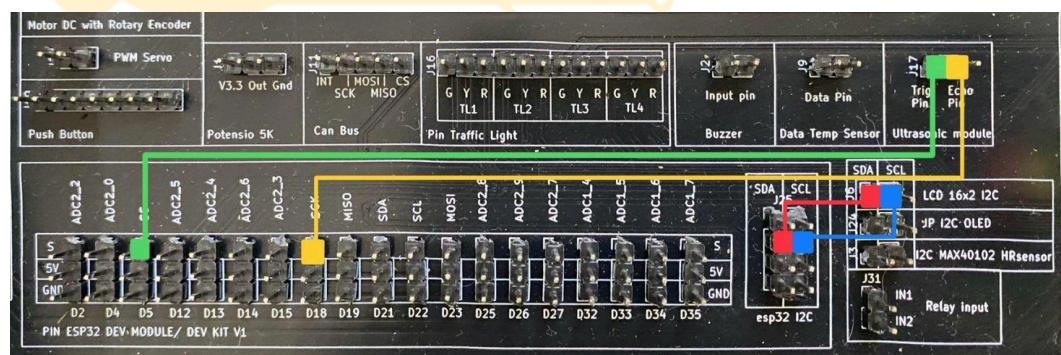
Tujuan dari praktikum ini adalah agar mahasiswa memahami prinsip dasar kerja sensor ultrasonik HC-SR04 dan penerapannya dalam sistem berbasis ESP32. Mahasiswa diharapkan mampu menghubungkan sensor ultrasonik dengan mikrokontroler, membaca hasil pengukuran jarak, serta menampilkannya pada LCD 16x2 dengan komunikasi I2C.

C. Alat dan Bahan

1. Laptop dengan Arduino IDE
2. Sensor ultrasonik HC-SR04 (sudah terpasang dalam kit)
3. LCD 16x2 I2C (sudah terpasang dalam kit)
4. Kabel jumper
5. Kabel USB Mikro

D. Prosedur

1. Penyusunan Rangkaian



- Hubungkan pin Trig pada sensor Ultrasonik ke pin D5 pada ESP32.
- Hubungkan pin Echo pada sensor Ultrasonik ke pin D18 pada ESP32.
- Hubungkan SDA dan SCL LCD 16x2 I2C ke pin SDA dan SCL ESP32.
- Pastikan koneksi jumper terpasang dengan benar.

2. Persiapan Arduino IDE

- Buka Arduino IDE pada laptop.
- Pastikan board ESP32 sudah terpasang di Arduino IDE.
- Hubungkan ESP32 dengan laptop menggunakan kabel USB mikro.

3. Program ESP32

```

1 // Library untuk komunikasi I2C dan LCD
2 #include <Wire.h>
3 #include <LiquidCrystal_I2C.h>
4
5 // Pin sensor ultrasonik
6 #define TRIG_PIN 5 // pin trigger ke ESP
7 #define ECHO_PIN 18 // pin echo ke ESP
8
9 // Inisialisasi LCD (alamat 0x27, ukuran 16x2)
10 LiquidCrystal_I2C lcd(0x27, 16, 2);
11
12 // Konfigurasi batas jarak
13 const int MIN_CM = 2; // jarak minimum valid (cm)
14 const int MAX_CM = 400; // jarak maksimum valid (cm)
15
16 void setup() {
17     Serial.begin(115200); // komunikasi serial (debug)
18     pinMode(TRIG_PIN, OUTPUT); // trigger sebagai output
19     pinMode(ECHO_PIN, INPUT); // echo sebagai input
20
21     // Inisialisasi LCD
22     Wire.begin(21, 22);
23     lcd.init();
24     lcd.backlight();
25     lcd.clear();
26     lcd.setCursor(0,0);
27     lcd.print("Ultrasonic Init");
28     delay(1000);
29     lcd.clear();
30 }
```

```

32 // Fungsi untuk membaca jarak dari sensor
33 float readDistanceCM() {
34     // Kirim pulsa trigger 10 microsecond
35     digitalWrite(TRIG_PIN, LOW);
36     delayMicroseconds(2);
37     digitalWrite(TRIG_PIN, HIGH);
38     delayMicroseconds(10);
39     digitalWrite(TRIG_PIN, LOW);
40
41     // Hitung durasi pulsa ECHO
42     long duration = pulseIn(ECHO_PIN, HIGH, 30000UL); // timeout 30ms
43
44     // Jika tidak ada pantulan (timeout)
45     if (duration == 0) return -1;
46
47     // Konversi durasi ke cm
48     float distance = (duration * 0.0343) / 2.0;
49     return distance;
50 }
51

52 void loop() {
53     float dist = readDistanceCM();
54
55     if (dist < 0 || dist < MIN_CM || dist > MAX_CM) {
56         // Jika tidak terdeteksi atau di luar jangkauan
57         Serial.println("Out of range");
58         lcd.setCursor(0,0);
59         lcd.print("Jarak: -- cm    ");
60         lcd.setCursor(0,1);
61         lcd.print("Out of range    ");
62     } else {
63         // Jika jarak valid
64         Serial.print("Jarak: ");
65         Serial.print(dist);
66         Serial.println(" cm");
67
68         lcd.setCursor(0,0);
69         lcd.print("Jarak: ");
70         lcd.print((int)dist); // tampilkan dalam cm (dibulatkan)
71         lcd.print(" cm    ");
72
73         lcd.setCursor(0,1);
74         lcd.print("Objek terdeteksi ");
75     }
76
77     delay(200);
78 }
79

```



4. Pengujian Hasil

- Amati tampilan pada LCD 16x2.

- Letakkan sebuah objek di depan sensor Ultrasonik dengan jarak yang berbeda-beda.
- Perhatikan nilai jarak yang ditampilkan pada LCD, apakah sesuai dengan perubahan posisi objek.

E. Tugas

Simpan data pembacaan ECG ke **SD card** selain ditampilkan di OLED. File hasil berupa data waktu dan nilai tegangan.



MODUL 6

MONITOR SUHU MENGGUNAKAN LCD 16X2 DAN BUZZER SEBAGAI TANDA PERINGATAN

A. Latar Belakang

Sensor suhu DS18B20 adalah sensor suhu digital yang menggunakan protokol komunikasi One-Wire, sehingga hanya membutuhkan satu pin data untuk bertukar informasi dengan mikrokontroler. Hal ini membuatnya sangat efisien dan mudah diintegrasikan dengan berbagai platform mikrokontroler, termasuk ESP32.

Kelebihan utama DS18B20 dibandingkan sensor suhu analog adalah tingkat akurasinya yang lebih tinggi ($\pm 0,5$ °C dalam rentang -10 °C hingga +85 °C) serta kemampuannya untuk memberikan data langsung dalam format digital, sehingga tidak memerlukan konversi analog-ke-digital tambahan.

Versi waterproof dari DS18B20 dibungkus dengan tabung stainless steel dan kabel pelindung, menjadikannya cocok untuk aplikasi di lingkungan yang basah atau ekstrem. Sensor ini sering digunakan pada bidang biomedis (misalnya monitoring suhu cairan tubuh atau inkubator), otomasi industri, hingga Internet of Things (IoT) untuk monitoring suhu lingkungan.

Dengan menggunakan protokol One-Wire, beberapa sensor DS18B20 dapat dipasang secara paralel pada satu jalur data, yang memungkinkan pembuatan sistem monitoring suhu multipoint hanya dengan sedikit pin mikrokontroler.

B. Tujuan

Tujuan dari praktikum ini adalah memahami cara menghubungkan dan menggunakan sensor suhu digital DS18B20 yang memiliki kemampuan tahan air (waterproof) dengan modul mikrokontroler ESP32. Melalui praktikum ini, mahasiswa diharapkan dapat memahami prinsip komunikasi One-Wire dalam membaca data suhu, menampilkan hasil

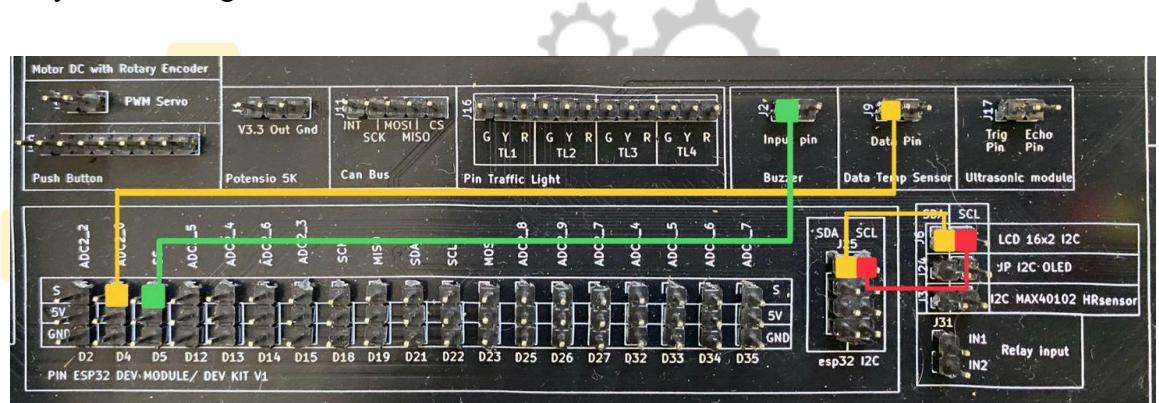
pembacaan pada LCD 16x2 berbasis I2C, serta menerapkan sistem peringatan dini dengan buzzer ketika suhu yang terdeteksi melewati batas tertentu.

C. Alat dan Bahan

1. Laptop dengan Arduino IDE
2. Sensor suhu waterproof DS18B20 + modul adaptor (white housing)
3. LCD 16x2 I2C
4. Buzzer
5. Kabel jumper
6. Kabel micro USB

D. Prosedur

1. Penyusunan Rangkaian



- Hubungkan white housing sensor DS18B20 ke modul adaptor sensor.
- Sambungkan pin data modul DS18B20 ke pin D4 ESP32.
- Hubungkan buzzer ke pin D5 ESP32.
- Hubungkan ESP32 ke laptop menggunakan kabel micro USB.

2. Persiapan Arduino IDE

- Buka Arduino IDE di laptop.
- Buka menu Library Manager dan install library OneWire serta DallasTemperature.
- Pastikan juga library LiquidCrystal_I2C sudah terpasang untuk mendukung tampilan LCD.

3. Program ESP32

```

1 #include <OneWire.h>
2 #include <DallasTemperature.h>
3 #include <LiquidCrystal_I2C.h>
4
5 #define ONE_WIRE_BUS 4 // pin data DS18B20 ke D4
6 #define BUZZER_PIN 5 // buzzer ke D5
7
8 OneWire oneWire(ONE_WIRE_BUS);
9 DallasTemperature sensors(&oneWire);
10 LiquidCrystal_I2C lcd(0x27, 16, 2); // alamat I2C LCD (0x27 bisa diganti 0x3F kalau beda)
11
12 void setup() {
13     pinMode(BUZZER_PIN, OUTPUT);
14     digitalWrite(BUZZER_PIN, LOW);
15
16     sensors.begin();
17     lcd.init();
18     lcd.backlight();
19 }
20
21
22 void loop() {
23     sensors.requestTemperatures();
24     float suhu = sensors.getTempCByIndex(0);
25
26     lcd.clear();
27     lcd.setCursor(0, 0);
28     lcd.print("Suhu: ");
29     lcd.print(suhu);
30     lcd.print(" °C");
31
32     if (suhu > 32) {
33         digitalWrite(BUZZER_PIN, HIGH); // nyalakan buzzer
34         lcd.setCursor(0, 1);
35         lcd.print("BAHAYA! >32°C");
36     } else {
37         digitalWrite(BUZZER_PIN, LOW); // matikan buzzer
38         lcd.setCursor(0, 1);
39         lcd.print("Suhu Normal");
40     }
41     delay(1000);
42 }
--
```



4. Pengujian Hasil

- Buka Serial Monitor pada Arduino IDE, ubah baudrate menjadi 115200.
- Celupkan sensor ke dalam air dingin, lalu amati perubahan nilai suhu pada LCD dan Serial Monitor.
- Ulangi langkah sebelumnya dengan air hangat, perhatikan apakah buzzer aktif jika suhu melebihi 32 °C.

- Catat hasil perubahan nilai suhu sesuai kondisi pengujian.

E. Tugas

Selain alarm buzzer, nyalakan motor DC ketika suhu melewati batas yang ditentukan.



MODUL 7

KOMUNIKASI CAN BUS ANTAR ESP

A. Latar Belakang

Komunikasi data merupakan salah satu aspek penting dalam sistem embedded, terutama ketika melibatkan lebih dari satu mikrokontroler yang harus saling bertukar informasi secara real-time. Salah satu protokol komunikasi yang banyak digunakan dalam aplikasi industri dan otomotif adalah Controller Area Network (CAN Bus).

CAN Bus dirancang untuk memungkinkan beberapa perangkat (node) berkomunikasi pada jalur komunikasi yang sama secara efisien, tahan terhadap gangguan, dan tanpa memerlukan host komputer sebagai pengendali utama. Protokol ini awalnya dikembangkan untuk otomotif, tetapi kini juga banyak dipakai di bidang robotika, sistem sensor terdistribusi, hingga perangkat medis.

ESP32 mendukung komunikasi CAN secara native melalui SJA1000-compatible CAN controller yang sudah tertanam di dalam chip, namun tetap memerlukan transceiver eksternal (misalnya SN65HVD230, MCP2551, atau TJA1050) agar dapat terhubung ke jalur fisik CAN (CAN_H dan CAN_L). Dengan adanya dukungan ini, mahasiswa dapat mempelajari secara langsung bagaimana dua ESP32 saling mengirim dan menerima data menggunakan protokol CAN.

Melalui praktikum ini, mahasiswa akan memahami dasar-dasar komunikasi CAN, cara konfigurasi bit rate, ID pesan, serta proses transmisi dan penerimaan data antar node. Pemahaman ini sangat penting karena protokol CAN digunakan luas dalam sistem kritis yang membutuhkan komunikasi cepat, handal, dan tahan terhadap gangguan.

B. Tujuan

Tujuan praktikum ini adalah agar mahasiswa memahami konsep dasar komunikasi CAN Bus dan mampu mengimplementasikannya pada ESP32. Melalui percobaan ini, mahasiswa diharapkan dapat merangkai perangkat keras dengan transceiver CAN,

mengkonfigurasi ESP32 untuk mengirim dan menerima data, serta mengamati hasil komunikasi melalui Serial Monitor sebagai bukti pertukaran data antar node.

C. Alat dan Bahan

1. Laptop dengan Arduino IDE
2. 2 unit ESP32
3. 2 modul transceiver CAN (misalnya MCP2515 atau SN65HVD230)
4. Kabel jumper
5. Kabel micro USB

D. Prosedur

1. Persiapan Arduino IDE
 - a. Buka Arduino IDE dan install esp32 by Espressif Systems dan install mcp2515.h
 - b. Hubungkan ESP 32 pertama dan CANbus pertama ke laptop pertama (Master)
 - c. Hubungkan ESP 32 kedua dan CANbus kedua ke laptop kedua (Slave)
2. Rangkaian CANbus dan ESP32
 - a. Hubungkan pin CS CANbus ke GPIO 5 pada ESP32,
 - b. Hubungkan pin SCK CANbus ke GPIO 18 ESP32,
 - c. Hubungkan SO (MISO) CANbus ke GPIO 19 ESP32,
 - d. Hubungkan SI (MOSI) CANbus ke GPIO 23 ESP32.
 - e. Hubungkan GND CANbus ke GND ESP32,
 - f. Hubungkan VCC CANbus ke VCC 5V (Vin) ESP32,
 - g. Hubungkan pin INT CANbus ke GPIO 4 pada ESP32,
 - h. Hubungkan pin CAN H CANbus 1 ke CAN H CANbus 2,
 - i. Hubungkan pin CAN L CANbus 1 ke CAN L CANbus 2,
3. Program ESP32 Server

```

1 #include <mcp2515.h>
2 #include <SPI.h>
3
4 MCP2515 mcp2515(5);
5 struct can_frame canMsg;
6
7 unsigned long last = 0;
8
9 void setup() {
10   Serial.begin(115200);
11   SPI.begin(18,19,23,5);
12   mcp2515.reset();
13   if (mcp2515.setBitrate(CAN_100KBPS, MCP_16MHZ) != MCP2515::ERROR_OK) {
14     Serial.println("Gagal set bitrate!");
15   } else Serial.println("Bitrate set OK");
16   mcp2515.setNormalMode();
17   Serial.println("Receiver siap");
18 }
19
20 void loop() {
21   int r = mcp2515.readMessage(&canMsg);
22   if (r == MCP2515::ERROR_OK) {
23     Serial.print("Diterima ID 0x"); Serial.print(canMsg.can_id, HEX);
24     Serial.print(" data: ");
25     for (int i=0;i<canMsg.can_dlc;i++) Serial.print((char)canMsg.data[i]);
26     Serial.println();
27   } else {
28     // // tampil setiap 2 detik kalau kosong -> membantu 'lihat aktif/tidak'
29     // if (millis() - last > 2000) {
30     //   Serial.println("Belum terima frame...");
31     //   last = millis();
32     // }
33   }
34 }
```

Setelah berhasil di-upload, buka Serial Monitor (115200 bps) dan masukan pesan yang ingin dikirim

4. Program ESP32 Client

```

1 #include <mcp2515.h>
2 #include <SPI.h>
3
4 MCP2515 mcp2515(5); // pastikan ini CS pin yang benar
5 struct can_frame canMsg;
6
7 void setup() {
8   Serial.begin(115200);
9   SPI.begin(18,19,23,5);
10  mcp2515.reset();
11  // ganti MCP_8MHZ ke MCP_16MHZ kalau modulmu punya kristal 16MHz
12  if (mcp2515.setBitrate(CAN_100KBPS, MCP_16MHZ) != MCP2515::ERROR_OK) {
13    Serial.println("Gagal set bitrate!");
14  } else Serial.println("Bitrate set OK");
15  mcp2515.setNormalMode();
16  Serial.println("Ketik pesan dan tekan ENTER:");
17 }
```

```

19 void loop() {
20   if (Serial.available()) {
21     String s = Serial.readStringUntil('\n');
22     s.trim();
23     if (s.length() == 0) return;
24
25     // kirim max 8 byte satu frame
26     int len = min(8, (int)s.length());
27     canMsg.can_id = 0x100;
28     canMsg.can_dlc = len;
29     memcpy(canMsg.data, s.c_str(), len);
30
31     // kirim
32     mcp2515.sendMessage(&canMsg);
33     Serial.print("Terkirim frame ID 0x");
34     Serial.print(canMsg.can_id, HEX);
35     Serial.print(" data: ");
36     for (int i=0;i<len;i++) Serial.print((char)canMsg.data[i]);
37     Serial.println();
38   }
39 }
```

Nanti akan muncul pada Serial Monitor hasil pesan yang dikirim dari Laptop pertama.

E. Tugas

Buat komunikasi dua arah: ESP A mengirim data suhu, ESP B mengirim data kelembaban. Keduanya ditampilkan di LCD masing-masing.

MODUL 8

KOMUNIKASI BLUETOOTH ANTAR ESP

A. Latar Belakang

Perkembangan teknologi mikrokontroler semakin pesat, terutama dengan hadirnya ESP32 yang memiliki fitur komunikasi nirkabel terintegrasi, seperti Wi-Fi dan Bluetooth. Kemampuan ini memungkinkan ESP32 digunakan dalam berbagai sistem tertanam dan proyek Internet of Things (IoT) yang memerlukan komunikasi antar perangkat tanpa menggunakan kabel. Salah satu bentuk komunikasi nirkabel yang praktis dan efisien adalah Bluetooth, yang umum digunakan karena konsumsi dayanya rendah serta tidak memerlukan infrastruktur jaringan yang kompleks.

Bluetooth Classic, khususnya dengan profil Serial Port Profile (SPP), memungkinkan dua perangkat ESP32 untuk bertukar data secara langsung menggunakan komunikasi serial berbasis protokol Bluetooth. Dalam konteks pembelajaran mikrokontroler, komunikasi Bluetooth antar ESP32 dapat digunakan untuk memahami dasar interkoneksi antar sistem, pengiriman dan penerimaan data sederhana, serta proses sinkronisasi perangkat. Penerapan ini sangat relevan dalam dunia nyata, seperti dalam sistem monitoring, pengendali jarak jauh, maupun komunikasi antar sensor-node.

Melalui praktikum ini, mahasiswa akan mendapatkan pemahaman teknis mengenai bagaimana dua buah ESP32 dapat saling terhubung dan berkomunikasi menggunakan Bluetooth tanpa memerlukan perangkat tambahan seperti router atau komputer sebagai perantara. Dengan memahami dasar komunikasi Bluetooth, mahasiswa dapat mengembangkan keterampilan untuk membangun sistem nirkabel yang lebih kompleks, yang menjadi fondasi penting dalam pengembangan teknologi IoT masa kini.

B. Tujuan

Praktikum ini bertujuan agar mahasiswa memahami konsep dasar komunikasi nirkabel menggunakan Bluetooth Classic dan mampu menerapkannya pada dua buah mikrokontroler ESP32. Mahasiswa diharapkan dapat memprogram satu ESP32 sebagai perangkat pengirim (master) dan satu ESP32 sebagai perangkat penerima (slave), serta

memastikan proses pengiriman dan penerimaan data berjalan secara real-time melalui komunikasi Bluetooth Serial Port Profile (SPP).

C. Alat dan Bahan

1. Laptop dengan Arduino IDE
2. Dua buah ESP32 Dev Module
3. Kabel jumper
4. 2 Kabel micro USB
5. Modul Ajar Mikrokontroller

D. Prosedur

1. Persiapan Arduino IDE
 - d. Buka Arduino IDE dan install esp32 by Espressif Systems
 - e. Hubungkan ESP 32 pertama ke laptop pertama (Master)
 - f. Hubungkan ESP 32 kedua ke laptop kedua (Slave)
2. Program ESP32 Server

```
1 #include "BluetoothSerial.h"
2 #include "esp_bt_device.h"
3
4 BluetoothSerial SerialBT;
5
6 // Ganti dengan MAC ESP32 server
7 uint8_t serverAddress[6] = {0x78, 0x1C, 0x3C, 0x2C, 0x1D, 0xA6};
8
9 void setup() {
10     Serial.begin(115200);
11     delay(300);
12     Serial.println("\n==== ESP32 SPP CLIENT (NO PIN) ===");
13
14     // Mulai sebagai Slave
15     if (!SerialBT.begin("ESP32_CLIENT", true)) {
16         Serial.println("Gagal start Bluetooth client");
17         while (1) delay(1000);
18     }
19
20     const uint8_t* mac = esp_bt_dev_get_address();
21     Serial.printf("BT MAC (client): %02X:%02X:%02X:%02X:%02X:%02X\n",
22                 mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
23
24     Serial.print("Coba connect ke server...");
25     if (SerialBT.connect(serverAddress)) {
26         Serial.println(" TERHUBUNG!");
27     } else {
28         Serial.println(" GAGAL connect. Pastikan server aktif dan dekat.");
29     }
30 }
```

```

27 void loop() {
28   if (SerialBT.available()) {
29     char c = SerialBT.read();
30     Serial.write(c);
31     if (c == '1') digitalWrite(LED, HIGH);
32     if (c == '0') digitalWrite(LED, LOW);
33   }
34   if (Serial.available()) {
35     SerialBT.write(Serial.read());
36   }
37 }
```

Setelah berhasil di-upload, buka Serial Monitor (115200 bps) dan catat MAC Address yang muncul.

3. Program ESP32 Client

Ganti isi *serverAddress[]* dengan MAC yang telah dicatat dari server.

```

1 #include "BluetoothSerial.h"
2 #include "esp_bt_device.h"
3
4 BluetoothSerial SerialBT;
5
6 // Ganti dengan MAC ESP32 server
7 uint8_t serverAddress[6] = {0x78, 0x1C, 0x3C, 0x2C, 0x1D, 0xA6};
8
9 void setup() {
10   Serial.begin(115200);
11   delay(300);
12   Serial.println("\n==> ESP32 SPP CLIENT (NO PIN) ==>");
13
14   // Mulai sebagai client (tanpa setPin)
15   if (!SerialBT.begin("ESP32_CLIENT", true)) {
16     Serial.println("Gagal start Bluetooth client");
17     while (1) delay(1000);
18   }
19
20   const uint8_t* mac = esp_bt_dev_get_address();
21   Serial.printf("BT MAC (client): %02X:%02X:%02X:%02X:%02X:%02X\n",
22   | | | | | mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
23
24   Serial.print("Coba connect ke server...");
25   if (SerialBT.connect(serverAddress)) {
26     Serial.println(" TERHUBUNG!");
27   } else {
28     Serial.println(" GAGAL connect. Pastikan server aktif dan dekat.");
29   }
30 }
```

```
32 void loop() {  
33     if (!SerialBT.connected()) return;  
34  
35     if (Serial.available()) SerialBT.write(Serial.read());  
36     if (SerialBT.available()) Serial.write(SerialBT.read());  
37 }
```

4. Pengujian Hasil

Setelah kedua ESP32 sudah menyala:

- ESP32 Server akan mencetak “Menunggu koneksi dari client...” di Serial Monitor.
- ESP32 Client akan mencoba menyambung, dan menampilkan “BERHASIL TERHUBUNG!” jika berhasil.

Pastikan kedua board saling dekat (< 2 meter), tidak terhubung ke perangkat lain seperti HP. Cobalah:

- Ketik 1 di Serial Monitor client → LED di server akan menyala.
- Ketik 0 → LED di server akan mati.
- Kirim pesan dari client ke server dan sebaliknya → muncul di masing-masing Serial Monitor.

E. Tugas

Buat aplikasi **chat sederhana**: ESP A mengirim teks ke ESP B (via Serial Monitor), lalu ditampilkan, begitu juga sebaliknya..

MODUL 9

PEMBACAAN SUHU DITAMPILKAN KE WEB SERVER ESP32

A. Latar Belakang

Internet of Things (IoT) merupakan konsep yang memungkinkan perangkat elektronik saling terhubung dan dapat diakses melalui jaringan internet. Salah satu aplikasi dasar IoT adalah menampilkan data sensor ke web server lokal yang dapat diakses melalui perangkat seperti laptop atau smartphone.

ESP32 menjadi pilihan yang tepat untuk implementasi ini karena memiliki Wi-Fi bawaan serta performa pemrosesan yang tinggi. Dengan menghubungkan sensor suhu DS18B20 ke ESP32, data suhu dapat dibaca, ditampilkan pada LCD 16x2, dan juga disajikan melalui halaman web sederhana. Hal ini memberikan pengalaman nyata bagi mahasiswa dalam memahami alur kerja sistem monitoring berbasis IoT, mulai dari akuisisi data sensor, pengolahan data di mikrokontroler, hingga publikasi data ke antarmuka pengguna berbasis web.

Penerapan monitoring suhu menggunakan web server lokal relevan pada banyak bidang, misalnya sistem biomedis untuk memantau suhu cairan, industri makanan untuk menjaga suhu air atau bahan, hingga aplikasi rumah pintar. Praktikum ini diharapkan dapat memperkenalkan mahasiswa pada konsep dasar web server ESP32 sebagai salah satu fondasi menuju aplikasi IoT yang lebih kompleks.

B. Tujuan

Tujuan praktikum ini adalah agar mahasiswa memahami bagaimana menghubungkan sensor suhu DS18B20 ke ESP32 dan menampilkan hasil pembacaan suhu pada LCD 16x2 serta web server lokal. Dengan praktikum ini mahasiswa dapat mempelajari integrasi sensor dengan ESP32, pengolahan data, serta cara mengakses data melalui jaringan Wi-Fi.

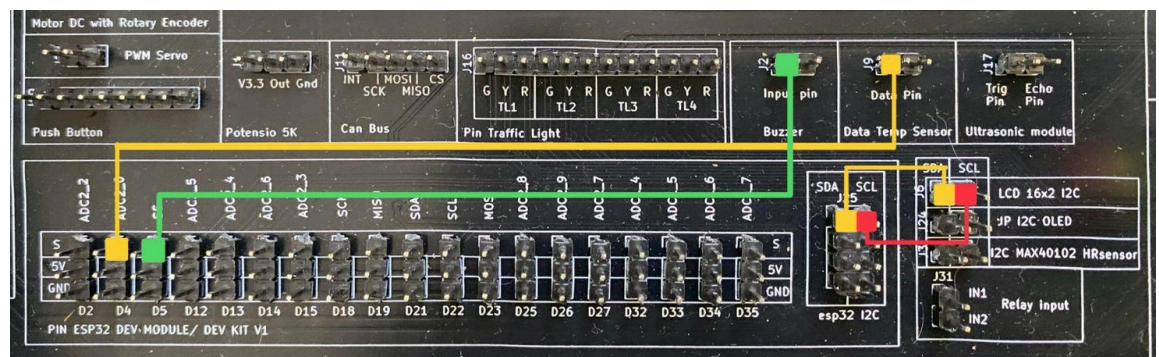
C. Alat dan Bahan

1. Laptop dengan Arduino IDE

2. Sensor suhu DS18B20 (waterproof)
 3. LCD 16x2 dengan antarmuka I2C (sudah terpasang dalam kit)
 4. Kabel jumper
 5. Kabel USB Mikro

D. Prosedur

- ## 1. Penyusunan Rangkaian



- Hubungkan pin data sensor DS18B20 ke pin D4 ESP32.
 - Hubungkan pin buzzer ke pin D5 ESP32

2. Persiapan Arduino IDE

 - Buka Arduino IDE di laptop.
 - Pastikan Board ESP32 sudah terinstall pada Arduino IDE.
 - Install library OneWire, DallasTemperature, dan LiquidCrystal_I2C dari Library Manager.
 - Hubungkan ESP32 ke laptop menggunakan kabel USB Mikro.

3. Program ESP32

```

1 #include <WiFi.h>
2 #include <OneWire.h>
3 #include <DallasTemperature.h>
4 #include <Wire.h>
5 #include <LiquidCrystal_I2C.h>
6
7 // Konfigurasi WiFi
8 const char* ssid = "KINARA";      // Ganti dengan SSID WiFi kamu
9 const char* password = "mautauajasih"; // Ganti dengan password WiFi kamu
10
11 WiFiServer server(80);
12
13 // DS18B20
14 #define ONE_WIRE_BUS 4
15 Onewire onewire(ONE_WIRE_BUS);
16 DallasTemperature sensors(&oneWire);
17
18 // LCD I2C
19 LiquidCrystal_I2C lcd(0x27, 16, 2);
20
21 float suhu = 0.0;
22

```

```

23 void setup() {
24   Serial.begin(115200);
25
26   // Start DS18B20
27   sensors.begin();
28
29   // Start LCD
30   lcd.init();
31   lcd.backlight();
32   lcd.setCursor(0,0);
33   lcd.print("ESP32 Web + LCD");
34
35   // Connect ke WiFi
36   WiFi.begin(ssid, password);
37   Serial.print("Menghubungkan ke WiFi...");
38   lcd.setCursor(0,1);
39   lcd.print("WiFi Connecting");
40
41   while (WiFi.status() != WL_CONNECTED) {
42     delay(500);
43     Serial.print(".");
44   }
45   Serial.println("");
46   Serial.println("WiFi Terhubung!");
47   Serial.print("IP Address: ");
48   Serial.println(WiFi.localIP());
49
50   lcd.clear();
51   lcd.setCursor(0,0);
52   lcd.print("WiFi OK");
53   lcd.setCursor(0,1);
54   lcd.print(WiFi.localIP().toString());
55

```



```

56   // Start server
57   server.begin();
58 }
59
60 void loop() {
61   // Ambil suhu
62   sensors.requestTemperatures();
63   suhu = sensors.getTempCByIndex(0);
64
65   // Tampilkan di Serial
66   Serial.print("Suhu: ");
67   Serial.print(suhu);
68   Serial.println(" *C");
69
70   // Tampilkan di LCD
71   lcd.clear();
72   lcd.setCursor(0,0);
73   lcd.print("Suhu Air:");
74   lcd.setCursor(0,1);
75   lcd.print(suhu);
76   lcd.print((char)223); // simbol derajat
77   lcd.print("C");
78
79   // Tangani client web
80   WiFiClient client = server.available();
81   if (client) {
82     String request = client.readStringUntil('\r');
83     client.flush();
84
85     // Kirim respon HTML
86     client.println("HTTP/1.1 200 OK");
87     client.println("Content-type:text/html");
88     client.println();
89     client.println("<!DOCTYPE html><html>");
90     client.println("<head><meta charset='UTF-8'><meta name='viewport' content='width=device-width, initial-scale=1'>");
91     client.println("<title>ESP32 Web Server</title></head>");
92     client.println("<body style='font-family:Arial;text-align:center;'>");
93     client.println("<h2>Monitoring Suhu Air</h2>");
94     client.print("<h1>");
95     client.print(suhu);
96     client.print(" &deg;C</h1>");
97     client.println("</body></html>");
98     client.println();
99     delay(1);
100   }
101
102   delay(1000); // update setiap 1 detik
103 }

```



4. Pengujian Hasil

- Buka Serial Monitor pada baudrate 115200 untuk melihat status koneksi Wi-Fi dan IP Address ESP32.
- Amati tampilan suhu pada LCD 16x2.
- Buka browser pada perangkat yang terhubung ke jaringan Wi-Fi yang sama, lalu masukkan alamat IP ESP32.

- Pastikan data suhu muncul pada halaman web server dengan format teks sederhana.

E. Tugas

Tambahkan tombol **ON/OFF kipas** pada halaman web server untuk mengontrol pin output ESP32 secara remote.



MODUL 10

PEMBACAAN SUHU DITAMPILKAN MENGGUNAKAN THINGSPEAK

A. Latar Belakang

Perkembangan teknologi Internet of Things (IoT) telah membawa kemajuan signifikan dalam bidang pemantauan data secara real-time, termasuk dalam hal pemantauan suhu lingkungan. Salah satu teknologi yang mendukung sistem IoT adalah mikrokontroler berbasis ESP32 yang memiliki konektivitas Wi-Fi, serta kemampuan untuk mengirimkan data sensor ke cloud secara langsung.

Dalam konteks praktikum ini akan mempelajari bagaimana mikrokontroler ESP32 dapat digunakan untuk membaca data suhu dari sensor digital DS18B20, kemudian mengirimkan data tersebut ke platform cloud ThingSpeak. ThingSpeak merupakan layanan berbasis web yang memungkinkan visualisasi data secara real-time, sehingga sangat cocok digunakan untuk keperluan monitoring jarak jauh.

Dengan mengintegrasikan pembacaan sensor suhu dan pengiriman data ke cloud, nantinya pengajaran tidak hanya mempelajari dasar-dasar pemrograman mikrokontroler, tetapi juga mendapatkan pemahaman praktis tentang bagaimana sistem IoT bekerja secara keseluruhan.

B. Tujuan

Praktikum bertujuan untuk dapat menyusun rangkaian antara ESP32 dan sensor DS18B20, memprogram pembacaan suhu, serta menghubungkannya dengan jaringan Wi-Fi dan platform ThingSpeak untuk menampilkan data suhu secara real-time dalam bentuk grafik.

C. Alat dan Bahan

1. Laptop dengan Arduino IDE
2. Akun ThingSpeak (thingspeak.com)
3. Kabel jumper
4. Sensor suhu DS18B20

5. Modul Ajar Mikrocontroller

D. Prosedur

1. Penyusunan Rangkaian



- a. Hubungkan pin data (DQ) dari "Data Temp Sensor" ke pin D4 (GPIO 4) pada baris S
- b. Pastikan koneksi jumper terpasang dengan benar.
2. Persiapan ThingSpeak
 - a. Buka browser dan kunjungi situs resmi: <https://thingspeak.com>
 - b. Klik tombol "Sign Up" di pojok kanan atas. Klik "Create Account". Setelah selesai, klik Create.
 - c. Klik "Sign In" dan login menggunakan akun MathWorks yang telah dibuat.
 - d. Setelah login, klik tab "Channels" di bagian atas.
 - e. Klik tombol hijau "New Channel"
 - f. Isi form pembuatan channel: Name: "Monitoring Suhu ESP32" Description: Bebas (opsional). Centang "Field 1" dan isi labelnya "Suhu (°C)". Public / Private: Biarkan default (Private). Klik tombol Save Channel di bagian bawah.
 - g. Perhatikan bagian Channel Info di sidebar kanan:
 - Channel ID → contoh: 1973821 (angka ini penting untuk dipakai di kode ESP32). Klik tab "API Keys", Salin Write API Key (biasanya berupa 16 karakter huruf/angka, misalnya A1B2C3D4E5F6G7H8). Gunakan key ini di dalam program Arduino untuk mengirim data.

h. Mengatur Tampilan Grafik dengan Klik “Private View”. Di bagian “Field 1 Chart”, klik ikon gerigi atau pengaturan lalu ubah dan klik Save.

- Title: "Grafik Suhu"
- Y-axis Label: "Suhu (°C)"
- Results: 100
- Days: 1

3. Program ESP32

```
1 #include <OneWire.h>
2 #include <DallasTemperature.h>
3 #include <WiFi.h>
4 #include "ThingSpeak.h"
5
6 // WiFi & ThingSpeak settings
7 const char* ssid          = "KINARA";           // Ganti dengan SSID kamu
8 const char* password      = "---";             // Ganti dengan password WiFi
9 unsigned long channelID   = 3057813;            // Ganti dengan Channel ID ThingSpeak
10 const char* writeAPIKey  = "3EU3QPX88KS72GT8"; // Ganti dengan Write API Key
11
12 WiFiClient client;
13
14 // Sensor DS18B20
15 #define ONE_WIRE_BUS 4    // pakai GPIO4
16 OneWire oneWire(ONE_WIRE_BUS);
17 DallasTemperature sensors(&oneWire);
18
19 // Interval kirim ke ThingSpeak
20 const unsigned long interval = 20000; // 20 detik
21 unsigned long lastSend = 0;
22
23 void setup() {
24     Serial.begin(115200);
25
26     // Inisialisasi sensor suhu
27     sensors.begin();
28
29     // Koneksi WiFi
30     WiFi.begin(ssid, password);
31     Serial.print("Menghubungkan ke WiFi");
32     while (WiFi.status() != WL_CONNECTED) {
33         delay(500);
34         Serial.print(".");
35     }
36     Serial.println("\nWiFi terhubung.");
37     Serial.print("IP Address: ");
38     Serial.println(WiFi.localIP());
39
40     // Mulai ThingSpeak
41     ThingSpeak.begin(client);
42 }
```

```

44 void loop() {
45     // Periksa koneksi WiFi
46     if (WiFi.status() != WL_CONNECTED) {
47         Serial.println("Koneksi WiFi terputus. Mencoba menghubungkan kembali...");
48         WiFi.begin(ssid, password);
49         while (WiFi.status() != WL_CONNECTED) {
50             delay(500);
51             Serial.print(".");
52         }
53         Serial.println("\nWiFi terhubung kembali.");
54     }
55
56     // Baca suhu dari sensor
57     sensors.requestTemperatures();
58     float tempC = sensors.getTempCByIndex(0);
59
60     if (tempC == DEVICE_DISCONNECTED_C) {
61         Serial.println("Gagal membaca suhu. Periksa koneksi sensor DS18B20.");
62     } else {
63         Serial.print("Suhu: ");
64         Serial.print(tempC);
65         Serial.println(" °C");
66     }
67
68     // Kirim ke ThingSpeak setiap interval
69     if (millis() - lastSend >= interval && WiFi.status() == WL_CONNECTED && tempC != DEVICE_DISCONNECTED_C) {
70         lastSend = millis();
71
72         // Debug info
73         Serial.printf("[TS] Mengirim data ke channel %lu\n", channelID);
74
75         // Set field dan kirim ke ThingSpeak
76         ThingSpeak.setField(1, tempC);
77         int httpCode = ThingSpeak.writeFields(channelID, writeAPIKey);
78
79         if (httpCode == 200) {
80             Serial.println("Data berhasil dikirim ke ThingSpeak");
81         } else {
82             Serial.printf("Gagal mengirim data. Kode HTTP: %d\n", httpCode);
83         }
84     }
85
86     delay(1000); // Tunggu 1 detik sebelum membaca lagi
87 }

```



4. Pengujian Hasil

- Buka **Channel ThingSpeak** yang sudah dibuat.
- Amati apakah data suhu muncul di **grafik Field 1** sesuai waktu kirim.

E. Tugas

Kirim **dua data sekaligus** (suhu & kelembaban) ke Thingspeak, lalu analisis grafik perbandingan keduanya.

