PERCOBAAN II
Finite State Machine

2.1 TUJUAN PERCOBAAN

Percobaan ini bertujuan untuk memberikan pengetahuan kepada mahasiswa tentang konsep *Finite State Machine* (FSM) dan penerapannya dalam pemrograman *embedded system*

2.2 PERANGKAT YANG DIGUNAKAN

Perangkat yang digunakan dalam praktikum ini adalah sebagai berikut:

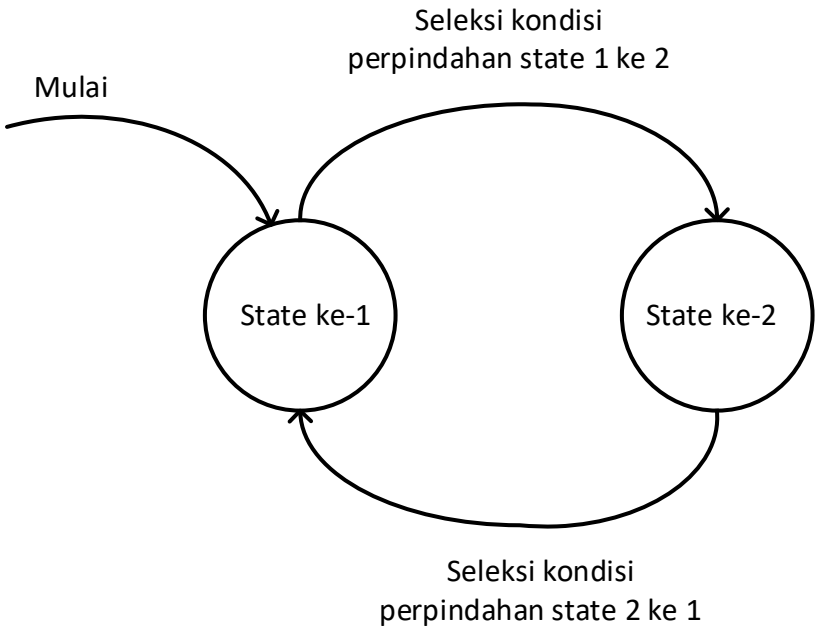
- Arduino UNO
- 3x LED
- 1x Push button
- 1x Resistor 1K
- Kabel jumper
- Project board
- Komputer yang terpasang Arduino IDE

2.3 DASAR TEORI

Finite State Machine (FSM)

Finite State Machine (FSM) merupakan model komputasional yang digunakan untuk merepresentasikan dan mengendalikan alur eksekusi sebuah program. Sebuah FSM terdiri dari beberapa kondisi/*state*, masukan/*input* dan keluaran/*output*. Pemrograman berbasis FSM digunakan jika sistem tersebut bekerja berdasarkan perubahan input atau pemicu tertentu. Sistem akan bekerja dalam beberapa kondisi/*state*. Pada FSM, *state* yang dapat aktif atau dieksekusi dalam satu waktu berjumlah satu *state* saja. Sebuah *embedded system* harus berpindah dari satu *state* ke *state* yang lain jika akan melakukan aksi yang berbeda. Perpindahan *state* tersebut memerlukan pemicuan dari sebuah masukan atau kejadian.

FSM biasa digambarkan dalam bentuk diagram *state*. Penggambaran FSM dalam bentuk diagram memudahkan programmer untuk memahami keseluruhan alur kerja sistem terlebih dahulu sebelum membaca kode program. Sebuah diagram *state* terdiri dari *state*, alur perpindahan dan kondisi yang menyebabkan berpindah. *State* direpresentasikan dalam bentuk lingkaran/oval. Alur perpindahan digambarkan dalam anak panah. Kondisi yang menyebabkan sebuah state berpindah ke state lainnya dituliskan di dekat anak panah alur. Sebuah diagram *state* sederhana ditunjukkan dalam Gambar 2.1.



Gambar 2.1 Diagram *state*

Konsep dasar pemrograman dalam FSM terdiri dari tiga bagian utama. Bagian pertama adalah deklarasi dari setiap *state*. Bagian deklarasi menjelaskan detail tahapan yang harus dikerjakan saat sistem ada di *state* tertentu. Bagian kedua adalah seleksi kondisi untuk perpindahan state. Perpindahan akan terjadi jika *input*/kejadian sesuai dengan kondisi perpindahan yang dirancang pada diagram *state*. Bagian ketiga adalah penentuan state selanjutnya jika kondisi perpindahan terpenuhi. *Pseudocode* FSM sederhana ditunjukkan dalam Tabel 2.1. Pada *pseudocode* tersebut, bagian deklarasi ditunjukkan dalam baris ke-18 dan seterusnya. Kode program menggunakan seleksi kondisi "*switch-case*" untuk menentukan program berada pada *state* tertentu seperti pada baris ke-7. Jumlah *case* yang digunakan sebanyak jumlah *state* yang telah ditetapkan. Pada setiap *state*, fungsi deklarasi dipanggil seperti yang terdapat dalam baris ke-9. Bagian kedua atau seleksi perpindahan *state* diletakkan setelah fungsi deklarasi dipanggil seperti yang terdapat dalam baris ke-10. Apabila seleksi kondisi terpenuhi, bagian ketiga atau penentuan *state* berikutnya diletakkan di dalam setiap *state* seperti yang ditunjukkan dalam baris ke-11.

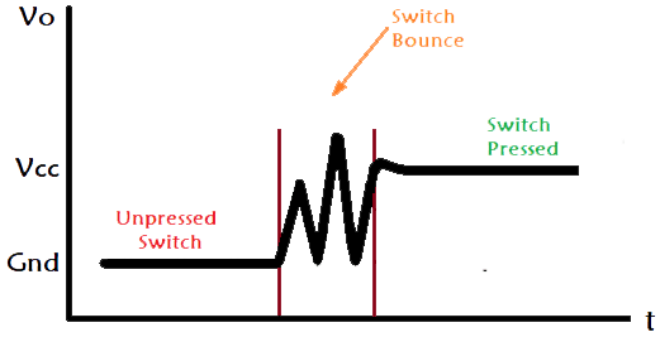
Tabel 2.1 Pseudocode FSM

1	inisialisasi fungsi_state_1();
2	
3	...
4	void loop(){
5	…
6	membaca_input/kejadian
7	switch(state){
8	case 1:
9	memanggil fungsi_state_1();

```
10   if (input/kejadian sesuai) {
11      state = state_selanjutnya;}
12
13   case 2:
14    ...
15    ...
16  }
17
18  detail fungsi_state_1(){
19    ...
20  }
```

Debouncing

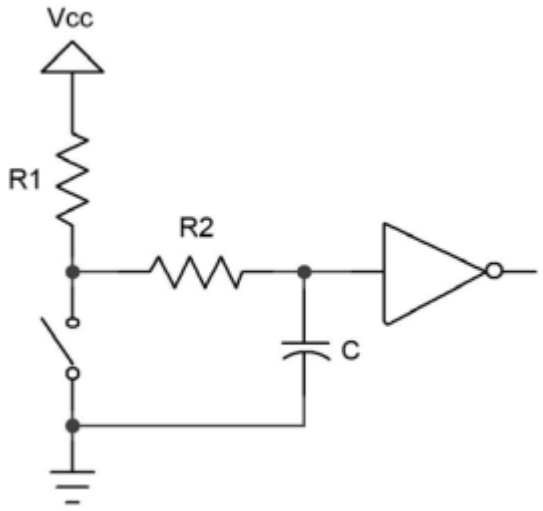
FSM berpindah dari satu *state* ke *state* lain bergantung dari sinyal masukan yang diterima oleh sistem. Beberapa sistem terkadang hanya menggunakan satu tombol masukan saja untuk berpindah dari state ke state lain. Kode program umumnya dirancang saat tombol ditekan maka sistem akan berpindah ke suatu *state*, kemudian jika tombol ditekan lagi maka sistem akan berpindah ke *state* selanjutnya. Penggunaan tombol sering menghasilkan kondisi *bouncing* pada saat tombol tersebut ditekan yang mengakibatkan sistem berpindah *state* tidak sesuai dengan alur FSM. Gambar 2.2 menunjukkan sinyal masukan yang disertai dengan *bouncing* atau derau. Kondisi *bouncing* yang sangat cepat ini menyebabkan sistem berpindah ke state lain dalam waktu singkat sehingga seolah-olah kode program berjalan tidak sesuai dengan diagram FSM yang dirancang. Berdasarkan Gambar 2.2, sebuah sinyal masukan diasumsikan berlogika rendah saat tombol masukan tidak ditekan. Pada saat tombol ditekan maka terjadi perubahan logika dari rendah ke logika tinggi. Perpindahan tersebut dapat disertai dengan derau yang menimbulkan perubahan logika dalam waktu singkat (*bounce*) sehingga menjadi pemicu untuk berpindah state dalam waktu singkat pula.



Gambar 2.2 Kondisi bouncing saat perpindahan input

Kondisi *bouncing* tersebut perlu dipertimbangkan dalam implementasi sistem atau penyusunan kode program. Cara menghilangkan *bouncing* atau *debouncing* dapat dilakukan melalui modifikasi perangkat keras atau perangkat lunak. *Debouncing* melalui perangkat keras dilakukan dengan menambahkan flip-flop atau resistor dan kapasitor pada jalur tombol seperti

yang ditunjukkan dalam Gambar 2.3. Nilai resistansi dan kapasitor perlu dihitung dan disesuaikan dengan karakteristik frekuensi *bouncing*.



Gambar 2.3 Perangkat keras *debouncing*

Debouncing dapat dilakukan melalui kode program dengan cara menambahkan mekanisme deteksi *bouncing* dan menjadikan sistem tidak merespon masukan selama *bouncing* tersebut terdeteksi. *Debouncing* dilakukan dengan tidak menganggap masukan selama selang waktu tertentu. Durasi waktu sistem tidak aktif tersebut umumnya ditentukan lebih lama dari durasi waktu *bouncing*. Setelah sistem melebihi durasi waktu tersebut, hasil pembacaan dari tombol akan disimpan sebagai logika masukan dan sistem menjalankan program sesuai kondisi masukan tersebut. Contoh *pseudocode* program *debouncing* ditunjukkan dalam Tabel 2.2.

Tabel 2.2 Pseudocode *debouncing*

1	inisialisasi variabel untuk menyimpan *logika_tombol*
2	inisialisasi variabel untuk menyimpan *logika_tombol_sebelumnya*
3	*batas_waktu_debouncing* = 10 ms
4	….
5	
6	void loop() {
7	baca tombol dan disimpan pada *variabel_sementara*
8	if (*variabel_sementara* berbeda dengan *kondisi_tombol_sebelumnya*) {
9	catat *waktu_awal*;
10	}
11	
12	if ((*waktu_sekarang* - *waktu_awal*) > *batas_waktu_debouncing*) {
13	perbaharui dan simpan *logika_tombol*
14	}
15	
16	…
17	kerjakan aksi sesuai *logika_tombol*
18	
19	*kondisi_tombol_sementara* = *variable_sementara*

20	}

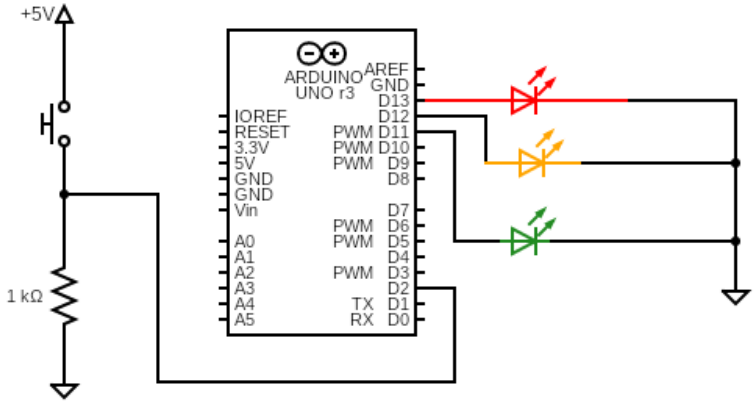
Pada *pseudocode* baris ke-3, desainer sistem akan menetapkan durasi waktu sistem tidak aktif. Saat terdeteksi perbedaan kondisi logika masukan saat ini dengan logika masukan sebelumnya sistem akan mencatat waktu seperti yang ditunjukkan dalam baris ke-8. Waktu tercatat tersebut digunakan sebagai waktu awal dalam menghitung durasi sistem tidak aktif atau masukan tidak dianggap. Jika waktu tunggu telah melebihi batas waktu seperti yang ditunjukkan pada baris ke-12 maka sistem akan menggunakan hasil pembacaan tombol pada baris ke-7 sebagai masukan dari sistem seperti yang ditunjukkan pada baris ke-13. Sistem akan mengerjakan sesuai dengan masukan yang tersimpan seperti yang ditunjukkan pada baris ke-17.

2.4 PROSEDUR PERCOBAAN

Percobaan FSM terdiri dari tiga percobaan. Percobaan pertama adalah pemrograman menggunakan FSM dengan satu masukan. Percobaan kedua adalah penggunaan debouncing pada FSM percobaan pertama. Percobaan ketiga adalah pemrograman FSM dengan lebih dari satu masukan atau pemicu untuk pergantian *state*.

A. Persiapan

1. Rangkai Arduino Uno, push button, LED dan resistor dengan menggunakan kabel jumper seperti pada rangkaian berikut



Gambar 2.4 Rangkaian percobaan FSM

2. Setelah rangkaian tersusun, mintalah kepada asisten praktikum untuk memeriksa kebenaran rangkaianya dahulu.
3. Hubungkan Arduino Uno dan komputer dengan menggunakan kabel USB.
4. Buka Arduino IDE, tuliskan kode program sesuai percobaan dan *Upload* kode program tersebut ke mikrokontroler Arduino Uno

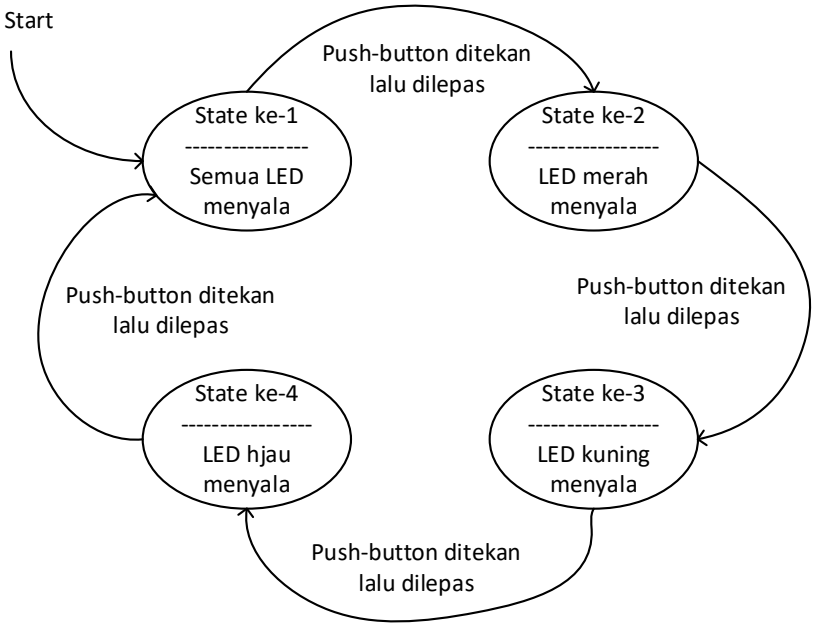
B. Percobaan 1: Percobaan FSM Satu Masukan

B.1 Kode Program

Percobaan ini akan menjalankan program yang terdiri dari empat *state* dengan satu masukan. Keluaran dari program ini akan menggunakan tiga LED dan masukan sistem menggunakan satu *push-button*. Setiap state dibedakan dengan nyala LED sebagai berikut:

- *State* ke-1: semua LED menyala
- *State* ke-2: LED Merah menyala, LED Kuning mati, LED Hijau mati
- *State* ke-3: LED Kuning menyala, LED Merah mati, LED Hijau mati
- *State* ke-4: LED Hijau menyala, LED Merah mati, LED Kuning mati

Program akan berpindah dari satu state ke state berikutnya pada saat push button ditekan. Diagram state yang digunakan pada percobaan ini ditunjukkan dalam Gambar 2.5.



Gambar 2.5 Diagram state percobaan 1

Kode program yang digunakan pada percobaan ke-1 ini ditunjukkan dalam Tabel 2.3.

Tabel 2.3 Kode Program FSM tanpa debouncing

```
1  #define LED_MERAH   13
2  #define LED_KUNING  12
3  #define LED_HIJAU   11
4  #define TOMBOL      2
5
6  int state=1;
7  int input_sblm = HIGH;
8  int tepi_naik = HIGH;
9
10 void state_1();
11 void state_2();
12 void state_3();
13 void state_4();
14
15 void setup() {
16   pinMode(LED_MERAH, OUTPUT);
17   pinMode(LED_KUNING, OUTPUT);
18   pinMode(LED_HIJAU, OUTPUT);
19   pinMode(TOMBOL, INPUT_PULLUP);
```

```
}

void loop() {
 int input = digitalRead(TOMBOL);

 if(input == LOW && input_sblm == HIGH){
  input_sblm = input;
 }
 else if(input == HIGH && input_sblm == LOW){
  input_sblm = input;
  tepi_naik = LOW;
 }

 switch(state){
  case 1:
   state_1();
   if(tepi_naik == LOW){
    state = 2;
   }
   break;
  case 2:
   state_2();
   if(tepi_naik == LOW){
    state = 3;
   }
   break;
  case 3:
   state_3();
   if(tepi_naik == LOW){
    state = 4;
   }
   break;
  case 4:
   state_4();
   if(tepi_naik == LOW){
    state = 1;
   }
   break;
 }

 tepi_naik = HIGH;
}

void state_1() {
 digitalWrite(LED_MERAH, HIGH);
 digitalWrite(LED_KUNING, HIGH);
```

```
66   digitalWrite(LED_HIJAU, HIGH);
67  }
68  void state_2(){
69   digitalWrite(LED_MERAH, HIGH);
70   digitalWrite(LED_KUNING, LOW);
71   digitalWrite(LED_HIJAU, LOW);
72  }
73  void state_3(){
74   digitalWrite(LED_MERAH, LOW);
75   digitalWrite(LED_KUNING, HIGH);
76   digitalWrite(LED_HIJAU, LOW);
77  }
78  void state_4(){
79   digitalWrite(LED_MERAH, LOW);
80   digitalWrite(LED_KUNING, LOW);
81   digitalWrite(LED_HIJAU, HIGH);
82  }
```

B.2 Analisis

1. Tekan tombol RESET pada Arduino Uno. Amati LED dan serial monitor. *State* manakah yang dikerjakan oleh mikrokontroler saat pertama kali beroperasi?
2. Tekan tombol push button secara berurutan dan amati state yang dikerjakan oleh mikrokontroler. Lengkapi tabel berikut.

Tabel 2.4 Data percobaan FSM tanpa *debouncer*

Tombol ditekan ke-	**State sesuai FSM**	**Percobaan ke-1**	**Percobaan ke-2**	**Percobaan ke-3**	**Percobaan ke-4**	**Percobaan ke-5**
0	State ke-1	State ke-1	State ke-1	State ke-1	State ke-1	State ke-1
1	State ke-2	….	….	….	….	….
2	State ke-3	….	….	….	….	….
3	State ke-4	….	….	…..	…..	…..
4	State ke-1	….	….	…..	…..	…..

Catatan:

- State ke-1: semua LED menyala
- State ke-2: LED Merah menyala, LED Kuning mati, LED Hijau mati
- State ke-3: LED Kuning menyala, LED Merah mati, LED Hijau mati
- State ke-4: LED Hijau menyala, LED Merah mati, LED Kuning mati

3. Berdasarkan tabel percobaan tersebut, apakah terdapat state mikrokontroler yang tidak sesuai dengan urutan pada diagram state? Jika ada, sebutkan pada urutan tombol ditekan ke berapa dan percobaan ke berapa?

4. Jelaskan mengapa kondisi tersebut terjadi!

C. Percobaan 2: Percobaan FSM dengan *Debouncing*

C.1 Kode Program

Kode program yang akan digunakan pada percobaan ini ditunjukkan dalam Tabel 2.5. Diagram state percobaan ini mengikuti diagram yang sama dengan percobaan sebelumnya seperti dalam Gambar 2.5.

Tabel 2.5 Kode program *debouncing*

```
1  #define LED_MERAH   13
2  #define LED_KUNING  12
3  #define LED_HIJAU   11
4  #define TOMBOL      2
5
6  int state=1;
7  int input_sblm = HIGH;
8  int tepi_naik = HIGH;
9  unsigned long waktu_debouncing = 0;
10 unsigned long delay_debouncing = 50;
11 int input = LOW;
12 int kondisi_sblm = LOW;
13
14 void state_1();
15 void state_2();
16 void state_3();
17 void state_4();
18
19 void setup() {
20   pinMode(LED_MERAH, OUTPUT);
21   pinMode(LED_KUNING, OUTPUT);
22   pinMode(LED_HIJAU, OUTPUT);
23   pinMode(TOMBOL, INPUT_PULLUP);
24 }
25
26 void loop() {
27   int kondisi = digitalRead(TOMBOL);
28
29   if(kondisi != kondisi_sblm){
30     waktu_debouncing = millis();
31   }
32   if((millis()-waktu_debouncing) > delay_debouncing){
33     if(kondisi != input){
34       input = kondisi;
35     }
36   }
37   kondisi_sblm = kondisi;
```

```
38
39  if(input == LOW && input_sblm == HIGH){
40    input_sblm = input;
41  }
42  else if(input == HIGH && input_sblm == LOW){
43    input_sblm = input;
44    tepi_naik = LOW;
45  }
46
47  switch(state){
48    case 1:
49      state_1();
50      if(tepi_naik == LOW){
51        state = 2;
52      }
53      break;
54    case 2:
55      state_2();
56      if(tepi_naik == LOW){
57        state = 3;
58      }
59      break;
60    case 3:
61      state_3();
62      if(tepi_naik == LOW){
63        state = 4;
64      }
65      break;
66    case 4:
67      state_4();
68      if(tepi_naik == LOW){
69        state = 1;
70      }
71      break;
72  }
73
74  tepi_naik = HIGH;
75 }
76
77 void state_1() {
78  digitalWrite(LED_MERAH, HIGH);
79  digitalWrite(LED_KUNING, HIGH);
80  digitalWrite(LED_HIJAU, HIGH);
81 }
82 void state_2(){
83  digitalWrite(LED_MERAH, HIGH);
```

```
84   digitalWrite(LED_KUNING, LOW);
85   digitalWrite(LED_HIJAU, LOW);
86  }
87  void state_3(){
88   digitalWrite(LED_MERAH, LOW);
89   digitalWrite(LED_KUNING, HIGH);
90   digitalWrite(LED_HIJAU, LOW);
91  }
92  void state_4(){
93   digitalWrite(LED_MERAH, LOW);
94   digitalWrite(LED_KUNING, LOW);
95   digitalWrite(LED_HIJAU, HIGH);
96  }
```

C.2 Analisis

1. Tekan tombol RESET pada Arduino Uno. Amati LED dan serial monitor. *State* manakah yang dikerjakan oleh mikrokontroler saat pertama kali beroperasi?
2. Tekan tombol push button secara berurutan dan amati state yang dikerjakan oleh mikrokontroler. Lengkapi tabel berikut.

Tabel 2.6 Data percobaan FSM tanpa *debouncer*

Tombol ditekan ke-	State sesuai FSM	Percobaan ke-1	Percobaan ke-2	Percobaan ke-3	Percobaan ke-4	Percobaan ke-5
0	State ke-1	State ke-1	State ke-1	State ke-1	State ke-1	State ke-1
1	State ke-2	….	….	….	….	….
2	State ke-3	….	….	….	….	….
3	State ke-4	….	….	…..	…..	…..
4	State ke-1	….	….	…..	…..	…..

Catatan:

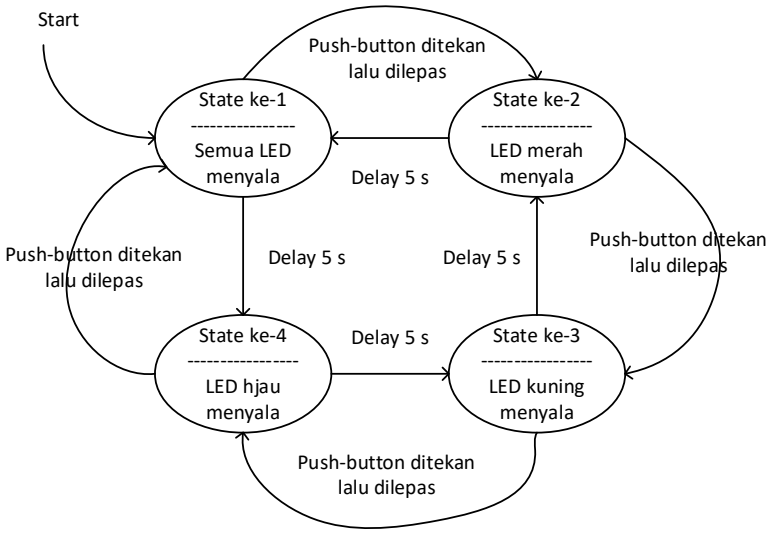
- State ke-1: semua LED menyala
- State ke-2: LED Merah menyala, LED Kuning mati, LED Hijau mati
- State ke-3: LED Kuning menyala, LED Merah mati, LED Hijau mati
- State ke-4: LED Hijau menyala, LED Merah mati, LED Kuning mati

3. Berdasarkan tabel percobaan tersebut, apakah terdapat state mikrokontroler yang tidak sesuai dengan urutan pada diagram state? Jika ada, sebutkan pada urutan tombol ditekan ke berapa dan percobaan ke berapa?
4. Apa yang membedakan dengan kode program pada Tabel 2.3?

D. Percobaan 3: Percobaan FSM Lebih dari Satu Masukan

D.1 Kode Program

Pada percobaan ini akan menggunakan dua masukan untuk berpindah dari satu state ke state lainnya. Jumlah state yang akan digunakan sebanyak empat buah seperti pada percobaan sebelumnya. Program akan berpindah dari satu state ke state berikutnya pada saat push button ditekan. Jika tombol push button tidak ditekan selama 5 detik, maka state akan berpindah sendiri ke state selanjutnya. Diagram state yang digunakan pada percobaan ini ditunjukkan dalam Gambar 2.6.



Gambar 2.6 Diagram state percobaan ke-3

Kode program yang digunakan pada percobaan ke-1 ini ditunjukkan dalam Tabel 2.7.

Tabel 2.7 Kode Program FSM tanpa debouncing

```
1  #define LED_MERAH   13
2  #define LED_KUNING  12
3  #define LED_HIJAU   11
4  #define TOMBOL      2
5
6  int state=1;
7  int input_sblm = HIGH;
8  int tepi_naik = HIGH;
9  unsigned long waktu_debouncing = 0;
10 unsigned long delay_debouncing = 50;
11 int input = LOW;
12 int kondisi_sblm = LOW;
13 unsigned long waktu_state = 0;
14 unsigned long delay_state = 5000;
15
16 void state_1();
17 void state_2();
18 void state_3();
19 void state_4();
```

```
20
21  void setup() {
22   pinMode(LED_MERAH, OUTPUT);
23   pinMode(LED_KUNING, OUTPUT);
24   pinMode(LED_HIJAU, OUTPUT);
25   pinMode(TOMBOL, INPUT_PULLUP);
26  }
27
28  void loop() {
29   int kondisi = digitalRead(TOMBOL);
30
31   if(kondisi != kondisi_sblm){
32    waktu_debouncing = millis();
33   }
34   if((millis()-waktu_debouncing) > delay_debouncing){
35    if(kondisi != input){
36     input = kondisi;
37    }
38   }
39   kondisi_sblm = kondisi;
40
41   if(input == LOW && input_sblm == HIGH){
42    input_sblm = input;
43   }
44   else if(input == HIGH && input_sblm == LOW){
45    input_sblm = input;
46    tepi_naik = LOW;
47   }
48
49   switch(state){
50    case 1:
51     state_1();
52     if(tepi_naik == LOW){
53      state = 2;
54      waktu_state = millis();
55     }
56     else if((millis()-waktu_state) > delay_state){
57      state = 4;
58      waktu_state = millis();
59     }
60     break;
61    case 2:
62     state_2();
63     if(tepi_naik == LOW){
64      state = 3;
65      waktu_state = millis();
```

```
66      }
67      else if((millis()-waktu_state) > delay_state){
68        state = 1;
69        waktu_state = millis();
70      }
71      break;
72    case 3:
73      state_3();
74      if(tepi_naik == LOW) > delay_state){
75        state = 4;
76        waktu_state = millis();
77      }
78      else if((millis()-waktu_state) > delay_state){
79        state = 2;
80        waktu_state = millis();
81      }
82      break;
83    case 4:
84      state_4();
85      if(tepi_naik == LOW){
86        state = 1;
87        waktu_state = millis();
88      }
89      else if((millis()-waktu_state) > delay_state){
90        state = 3;
91        waktu_state = millis();
92      }
93      break;
94   }
95
96   tepi_naik = HIGH;
97  }
98
99  void state_1() {
100  digitalWrite(LED_MERAH, HIGH);
101  digitalWrite(LED_KUNING, HIGH);
102  digitalWrite(LED_HIJAU, HIGH);
103 }
104 void state_2(){
105  digitalWrite(LED_MERAH, HIGH);
106  digitalWrite(LED_KUNING, LOW);
107  digitalWrite(LED_HIJAU, LOW);
108 }
109 void state_3(){
110  digitalWrite(LED_MERAH, LOW);
111  digitalWrite(LED_KUNING, HIGH);
```

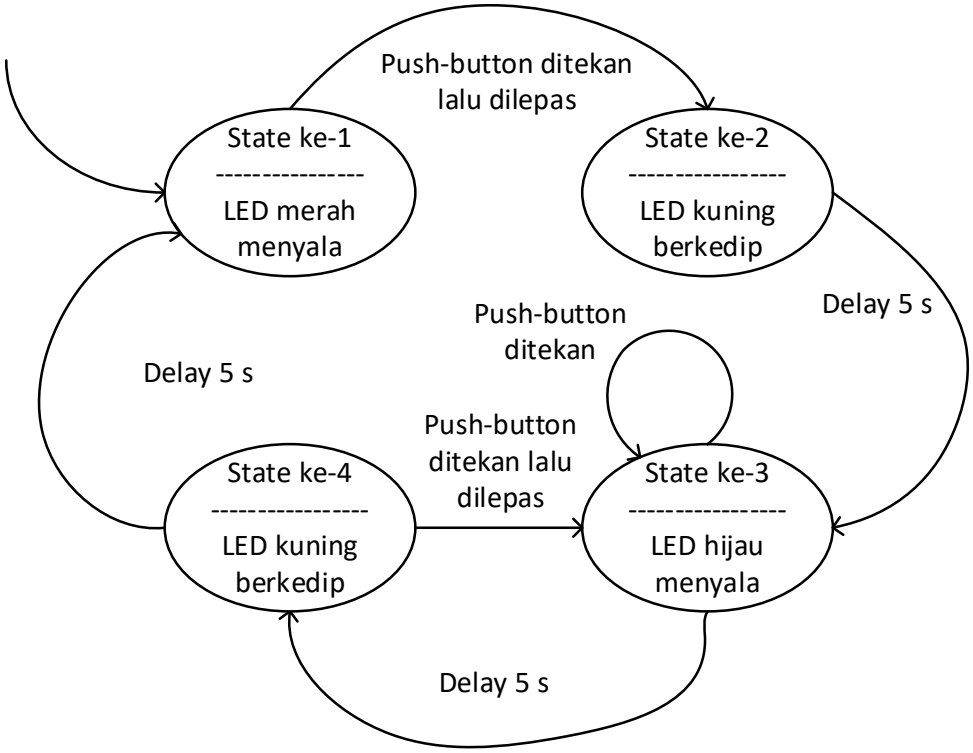
```
112   digitalWrite(LED_HIJAU, LOW);
113  }
114  void state_4(){
115   digitalWrite(LED_MERAH, LOW);
116   digitalWrite(LED_KUNING, LOW);
117   digitalWrite(LED_HIJAU, HIGH);
118  }
```

D.2 Analisis

1. Tekan tombol RESET pada Arduino Uno. Amati LED dan serial monitor. *State* manakah yang dikerjakan oleh mikrokontroler saat pertama kali beroperasi?
2. Tekan tombol push button secara berurutan dan amati state yang dikerjakan oleh mikrokontroler. Apakah urutan state sudah sesuai dengan diagram state yang dirancang.
3. Biarkan tombol push button dan amati LED selama kurang lebih 10 detik. Apa yang terjadi pada state program?
4. Apa yang membedakan antara kode program pada percobaan ke-2 dan percobaan ke-3? Jelaskan.

2.5 TUGAS

Susunlah kode program dalam FSM sistem penyeberang jalan yang mengikuti diagram *state* seperti dalam Gambar 2.7. Saat pertama kali dinyalakan sistem akan menyalakan LED merah. Ketika operator menekan push-button, sistem akan mematikan LED merah dan membuat LED kuning berkedip. Setelah selang waktu 5 detik, sistem akan menyalakan LED hijau sebagai isyarat penyeberang jalan boleh melintas. Apabila push button ditekan, maka sistem akan mempertahankan LED hijau menyala. Apabila tidak ada yang menekan tombol selama 5 detik, sistem akan mematikan LED hijau dan menyalakan LED merah dengan terlebih dahulu mengedipkan LED kuning sebagai peringatan.



Gambar2.7 Diagram state lampu isyarat penyebrangan

2.6 KESIMPULAN

Berdasarkan percobaan yang telah dilakukan, maka tuliskan kesimpulan percobaan di tempat yang telah disediakan di bawah ini: