Modul I

Digital Input / Output

1.1 Tujuan

Praktikum ini memberi dasar kompetensi pemakaian mikroprosesor berbasis Arduino dalam memecahkan berbagai masalah yang menggunakan masukan/luaran digital. Kompetensi khusus yang harus dikuasai sesuai dengan percobaan yang akan dilaksanakan adalah:

No	Kompetensi	Percobaan		
		1	2	3
1	Rangkaian masukan digital dengan konfigurasi *pull-up*.	×	×	
2	Rangkaian masukan digital dengan konfigurasi *pull-down*.	×	×	
3	Rangkaian luaran LED 7 segment memakai **decoder**	×	×	×
5	Rangkaian masukan keypad memakai **encoder**			×
6	Algoritma event-driven	×	×	
7	Algoritma state-driven	×		×
8	Algoritma luaran ke 7-segment memakai decoder	×	×	×
10	Algoritma membaca keypad memakai encoder			×

1.2 Alat dan Bahan

No	Nama	Banyak	Keterangan
1	Komputer / Laptop	1	
2	Arduino + Kabel USB	1	
3	Kit arduino	1	
4	Kit 7 segment	1	
5	Kit 7 keypad	1	
6	Power Supply	1	
7	Kabel IDC-10	2	

1.3 Permasalahan

Ketika mendesain masukan/luaran suatu sistem mikroprosesor, spesifikasi yang sangat menentukan adalah banyaknya pin I/O dan jenisnya (misal analog, digital, PWM, dll). Untuk arduino Uno, jumlah pin I/O-nya ada 20 buah (A0 - A5, D0 - D13). Sementara itu jenisnya bisa dikonfigurasi dengan cukup fleksibel antara lain menjadi:

1. 20 pin digital I/O
2. 6 pin analog input
3. 6 analog output (PWM)
4. 2 pin interupsi
5. 2 pin komunikasi (USB)

Pada praktikum ini kita akan mempelajari jenis luaran/masukan yang pertama dulu. Sesuai namanya, digital I/O adalah proses pertukaran sinyal yang nilainya hanya ada dua kemungkinan, disebut HIGH dan LOW. Standar sinyal yang diikuti Arduino adalah transistor-transistor logic TTL, dimana :

1. Sinyal HIGH : listrik diatas 2 Volt, dengan tegangan supply VCC = 4,75 - 5,25 V
2. Sinyal LOW : listrik dibawah 0,8 Volt

Karena arduino hanya punya 20 pin yang bisa dipakai untuk berbagai masukan/luaran, alokasi pin ini menjadi tantangan tersendiri. Sebisa mungkin kita harus hemat menggunakan pin I/O. Untuk itu, trik yang biasa digunakan adalah:

1. Menggunakan demultiplexer / decoder atau shift register untuk luaran.
2. Menggunakan multiplexer / encoder untuk masukan.

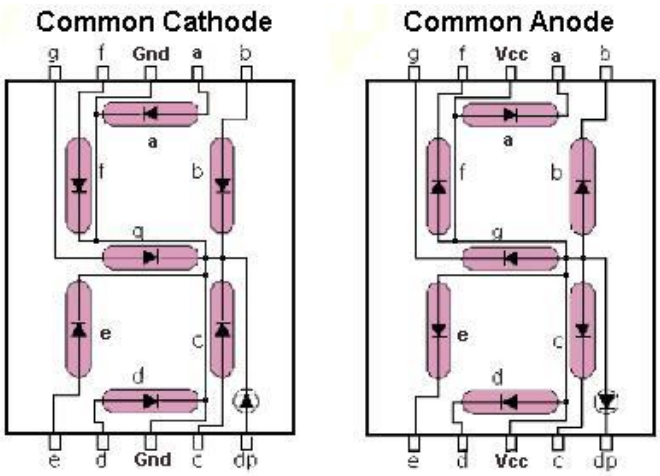
Hal tersebut akan kita lakukan pada 3 eksperimen berikut.

1.3.1 Pencacah 1 Digit

Pencacah (*counter*) adalah alat untuk menghitung jumlah kejadian tertentu. Kali ini kita akan membuat alat yang menghitung jumlah penekanan sebuah tombol, dan menampilkannya ke sebuah LED 7 segment. Dengan menggunakan 1 buah LED 7 segmen, maka jumlah maksimum pencacahan hanya mencapai 9. Setelah mencapai 9 maka pencacahan akan kembali ke 0.

1.3.1.1 Rangkaian Elektronika

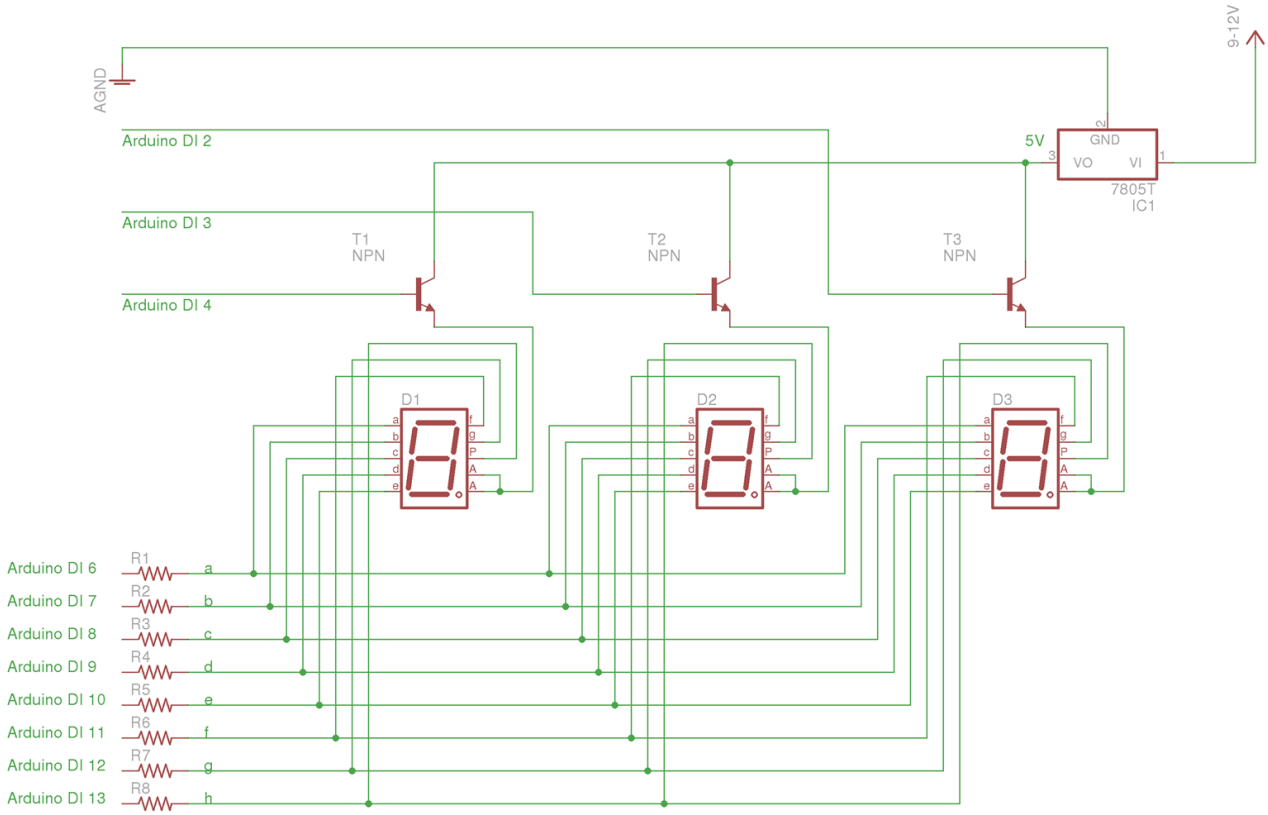
Pada dasarnya 7-segment adalah kumpulan dari 8 buah LED yang diintegrasikan seperti Gambar 1.1. Nampak bahwa ada 7 buah LED membentuk angka 8, dan sebuah LED menjadi titik (DP). Masing-masing LED bisa dinyalakan secara individual sehingga bisa membentuk angka 0 hingga 9, dan sebuah titik jika diperlukan.



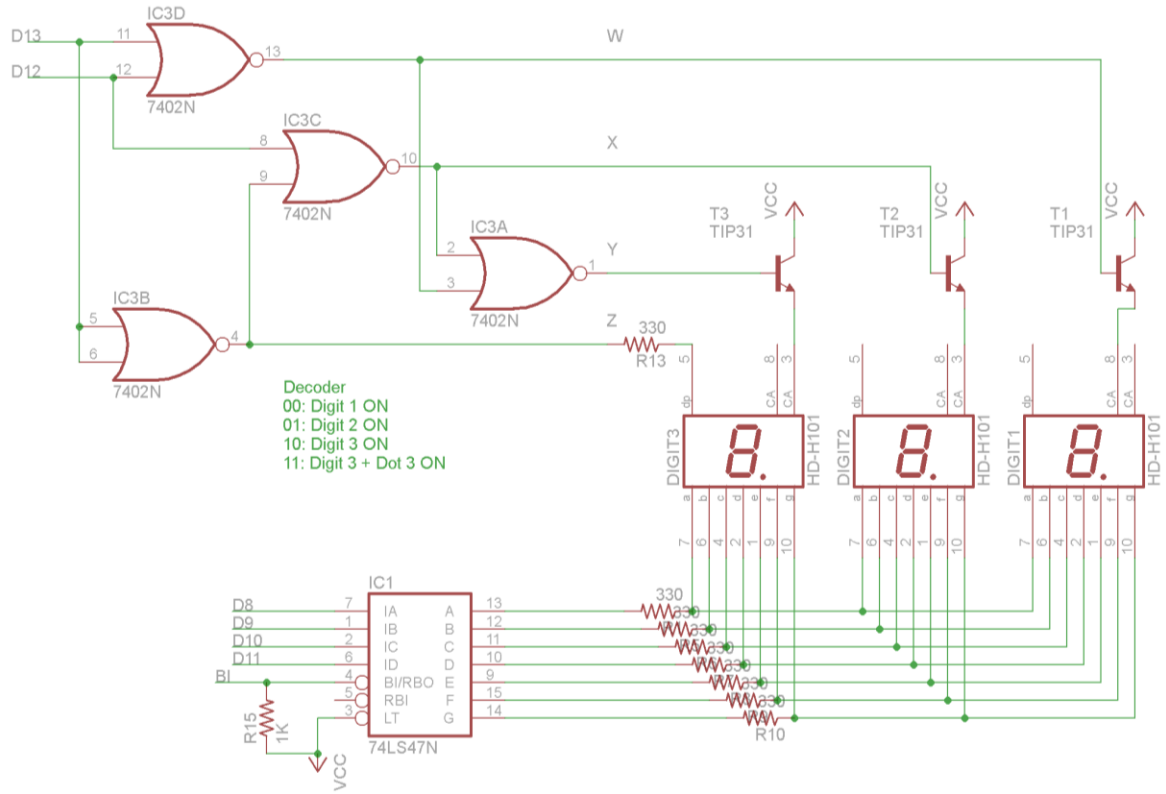
Gambar 1.1 Konfigurasi LED pada 7-segment

Secara umum ada dua jenis LED 7-segment yaitu *common-anode* dan *common-cathode*. Pada tipe *common-catodhe* semua katode LED tersambung ke pin GND, sehingga masing-masing LED bisa dinyalakan dengan menyambung pin a hingga g dan dp ke listrik positip (Vcc). Sebaliknya pada tipe *common-anode*, seluruh kaki anoda tersambung ke pin Vcc sehingga untuk menyalakan LED, masing-masing pin disambung ke listrik GND.

Gambar 1.2 memperlihatkan suatu rangkaian standar 3 buah LED 7 Segment. Pada rangkaian ini digunakan LED tipe *common anode.* Kaki Vcc masing-masing 7-segment disambung ke transistor (T1, T2, T3) yang berfungsi sebagai saklar elektronik. Lalu base transistor tersebut terkoneksi ke pin digital output Arduino, sehingga bila diberi luaran HIGH maka 7-segment yang bersangkutan akan mendapat pasokan listrik positif. Dengan demikian kita bisa memilih LED mana yang akan dinyalakan memakai 3 pin luaran digital (D2 - D4). Sementara itu untuk seluruh LED, kaki katode yang sama disambung ke sebuah resistor, lalu disambung ke pin Arduino. Selanjutnya bila pin tersebut memberi sinyal luaran LOW, maka LED yang bersangkutan akan menyala. Artinya, kita bisa menentukan segmen mana yang menyala dengan 8 pin luaran digital (D6-D13).



Gambar 1.2 Rangkaian multi LED 7-segment



Gambar 1.3 Modifikasi rangkaian multi LED 7-segment

Namun, rangkaian Gambar 1.2 tersebut punya sedikit masalah. Dia memerlukan 11 pin I/O dari Arduino. Tak banyak lagi yang tersisa seandainya kelak kita ingin menambahkan rangkaian lain. Untuk menghemat pin I/O itulah kita akan gunakan rangkaian Gambar 1.3 yang menggunakan:

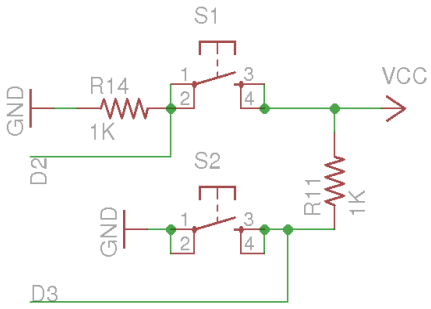
- Decoder BCD to 7 Segment (74*47). Chip ini mampu menerima masukan 4 bit (pin IA - ID) berupa kode BCD (Binary coded decimal), kemudian akan menyalakan 7 bit output (pin A-G) sesuai dengan segmen yang harus dinyalakan sesuai BCD ybs. Output chip ini bersifat open collector, karena itu 7 segment yang digunakan harus yang tipe common-anoda.
- Keping logika 7402 (*quad dual input NOR gate*) yang dirangkai sedemikian rupa sehingga dapat menghasilkan 4 output dari 2 input sesuai Tabel 1.1.

Tabel 1.1. *Tabel kebenaran decoder pemilih digit LED 7 segment.*

Input		**Output**				**Keterangan**
D12	D13	W	X	Y	Z	
0	0	H	L	L	H	T1 aktif, digit 1 menyala
0	1	L	H	L	L	T2 aktif, digit 2 menyala
1	0	L	L	H	H	T3 aktif, digit 3 menyala
1	1	L	L	H	L	T3 aktif, digit 3 dan titiknya menyala

Dengan kedua decoder tesebut, maka pin I/O yang diperlukan bisa berkurang menjadi 6 buah saja. Namun dibanding rangkaian sebelumnya, hanya satu digit yang bisa menyala sebab decoder hanya mengijinkan salah satu transistor yang aktif. Berikut ini fragmen menampilkan angka 3 di DIGIT-3.

```
// menampilkan angka 3 ke rangkaian LED 7-segmen dasar
// Asumsi: pin 6-13 sudah dalam mode OUTPUT
byte data=3, index=3;

// tampilkan data ke LED 7 segment lewat encoder di pin 8 - 11
digitalWrite(8, data & 0x01);
digitalWrite(9, data & 0x02);
digitalWrite(10, data & 0x04);
digitalWrite(11, data & 0x08);

// nyalakan digit sesuai index dengan decoder di pin 12-13
// perhatikan bahwa pin 12 adalah most significant byte
digitalWrite(12, index & 0x02);
digitalWrite(13, index & 0x01);
```



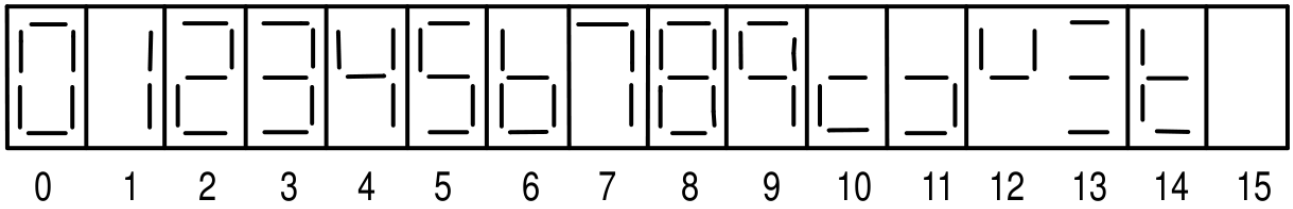
Gambar 1.4. Rangkaian Tombol

Selanjutnya kita memerlukan rangkaian tombol. Untuk itu gunakan rangkaian tombol seperti Gambar 1.4. Pull-down koneksikan ke pin D2, serta Pull-up koneksikan ke D3.

1.3.1.2 Program Arduino

Ada dua algoritma yang perlu mendapat perhatian pada pemrograman pencacah ini. Yang pertama adalah cara menyalakan LED 7-segmen, dan yang kedua tentu saja bagaimana menghitung cacahan tombol.

Untuk menyalakan LED 7-segmen, urusan menjadi mudah karena 74*47. Dengan memberi masukan 4 bit BCD (binary code decimal) ke chip tersebut, maka 7 segment akan menyala dengan benar.



Gambar 1.5 Tampilan 7 segment decoder menurut data-sheet

Untuk menguji hal tersebut, coba buatlah program sederhana berikut:

```
// Pemetaan Port arduino
#define BCD1       8
#define T1    12
#define T2   13

// fungsi untuk menampilkan BCD ke 7 segment melalui encoder
void writeDigit(int bcd){
  for(int i=0; i<4; i++) {
    digitalWrite(BCD1 + i, bcd & 0x01);
    bcd = bcd >> 1;
  }
}

// variabel global counter
int counter = 0;
byte index = 0;
```

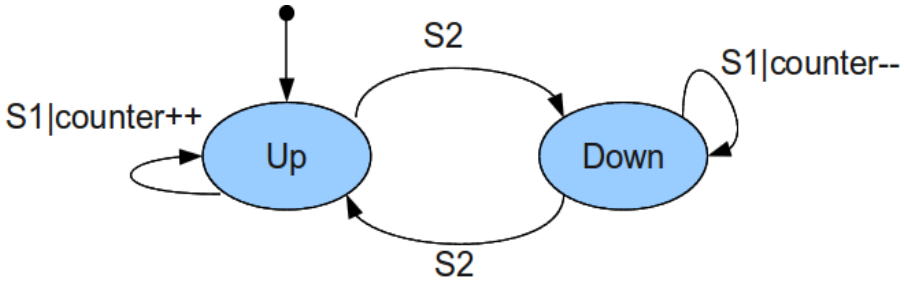
```
void setup() {
  Serial.begin(115200);
  for (int pin=BCD1; pin<=T2; pin++) {
    pinMode(pin, OUTPUT);
  }

  // nyalakan digit sesuai index dengan decoder di pin 12-13
  // perhatikan bahwa pin 12 adalah most significant byte
  digitalWrite(T1, index & 0x02);
  digitalWrite(T2, index & 0x01);
}

void loop() {
  // loop menampilkan counter naik 1 setiap 1 detik
  // modulo 10 agar tetap BCD
  // silakan coba modulo 16 untuk melihat kode hexa
  writeDigit(counter);
  counter = (counter + 1) % 10;
  Serial.print("Counter = ");
  Serial.println(counter);
  delay(1000); // tunda, buat lebih lama kalau mau melihat lebih jelas
}
```

Kode 1. 1. Seven Segment - Counter.

Sementara itu untuk mencacah, kita akan mulai berlatih diagram transisi keadaan (DTA). Spesifikasi program yang akan kita buat adalah:

1. Saat awal, sistem berada pada mode UP. Pada saat itu bila S1 ditekan, maka sistem akan mencacah naik.
2. Sementara itu bila S2 ditekan, sistem akan pindah ke mode DOWN. Pada saat itu bila S1 ditekan maka sistem mencacah turun.
3. Bila S2 ditekan lagi, sistem kembali ke mode UP.

Bagi sistem pencacah ini, DTA-nya cukup sederhana, seperti ditunjukkan pada Gambar berikut.



Gambar 1.6. Diagram transisi keadaan pencacah 1 digit

Implementasi program sesuai diagram tersebut dengan algoritma state-driven adalah sebagai berikut.

```
// Pemetaan Port Arduino pada Array
#define BUTTON1 4
#define BUTTON2 2
```

```
#define BCD1    8
#define T1  12
#define T2 13

// fungsi memeriksa apakah tombol S1 ditekan
boolean s1IsPressed() {
  static boolean b1_old = 0;
  boolean b1 = digitalread(BUTTON1);
  boolean s1 = (b1 && !b1_old);
  b1_old = b1;
  return s1;
}

// fungsi memeriksa apakah tombol S2 ditekan
boolean s2IsPressed() {
  // ..................
}

// fungsi untuk menampilkan BCD ke 7 segment melalui encoder
void writeDigit(int bcd){
 // .................
}

// variabel global
#define STATE_UP    0
#define STATE_DOWN  1
int counter = 0;
int state, next_state;
boolean s1, s2;
byte index = 0;

void setup() {
 Serial.begin(115200);
 pinMode(BUTTON1, INPUT);
 pinMode(BUTTON2, .....);
 for (int pin=BCD1; pin<=T2; pin++) {
   pinMode(pin, ....);
 }

 // menyalakan digit1
 digitalWrite(T2, index & 0x02);
 digitalWrite(T2, index & 0x02);

 // awal transisi
 next_state=state=0;
 counter=0;
 writeDigit(counter);
}

// saat state0, proses masing-masing event
void state0() {
 if (s1) {
   Serial.println("State UP - Event S1");
   // naikkan counter, kalau sudah 10 jadikan 0 lagi
   counter = (counter+1) % 10;
```

```
   writeDigit(counter);
  }
  else if (s2) {
   Serial.println("State UP - Event S2");
   next_state = STATE_DOWN;
  }
}

// saat state1, proses masing-masing event
void state1() {
  if (s1) {
   Serial.println("State DOWN - Event S1");
   // turunkan counter, kalau sudah -1 jadikan 9 lagi
   // ..............
  }
  else if (s2) {
   Serial.println("State DOWN - Event S2");
   // ubah state
   // ..........
  }
}

void loop() {
  // baca semua input
  s1 = s1IsPressed();
  s2 = s2IsPressed();

  // proses keadaan, state-driven
  switch(state){
   case 0 : state0(); break;
   case 1 : ........; break;
  }

  // pelaporan dan update state
  if (next_state != state) {
    Serial.print("State ");
    Serial.println(state);
    Serial.print(" ==> ");
    Serial.println(next_state);
    state = next_state;
  }
}
```

Kode 1. 2. Counter – state driven.

Hal yang sama juga bisa dilakukan dengan pendekatan sebaliknya, yaitu menggunakan algoritma event-driven seperti contoh berikut.

```
// Pemetaan Port Arduino pada Array
// ............

// fungsi memeriksa apakah tombol S1 ditekan
boolean s1IsPressed() {..........}
```

```
// fungsi memeriksa apakah tombol S2 ditekan
boolean s2IsPressed() { .............}

// fungsi untuk menampilkan BCD ke 7 segment melalui encoder
void writeDigit(int bcd){.................}

// variabel global
#define STATE_UP    0
#define STATE_DOWN  1
int counter = 0;
int state, next_state;
boolean s1, s2;
byte index = 0;

void setup() {
  // sama seperti counter – state driven
  // .....
}

// event S1, proses untuk masing-masing state
void eventS1() {
  Serial.print("Event S1 - state ");
  Serial.println(state);
  switch(state) {
     case STATE_UP :
       // naikkan counter, kalau sudah 10 jadikan 0 lagi
       counter = (counter+1) % 10;
       writeDigit(.......);
       break;
     case STATE_DOWN:
       // turunkan counter, kalau sudah -1 jadikan 9 lagi
       // ........
  }
}

// event S2, proses untuk masing-masing state
void eventS2() {
  Serial.print("Event S2 - state ");
  Serial.println(state);
  switch(state) {
    case STATE_UP:
       // ubah state
       next_state = STATE_DOWN; break;
    case STATE_DOWN:
       // ubah state
       // ..........
  }
}

void loop() {
  // baca semua input
  s1 = s1IsPressed();
  s2 = s2IsPressed();
```

```
  // proses event
  if (s1) eventS1();
  else if (s2) eventS2();

  // pelaporan dan update state
  if (next_state != state) {
    // .............
  }
}
```

Kode 1. 3. Counter – Event Driven

1.3.2 Stop-watch 7-Segment 3 Digit

Stop-watch adalah alat yang biasa dipakai untuk mengukur waktu. Cara kerjanya cukup mudah:

1. Saat awal, stop-watch akan menampilkan angka 000
2. Bila tombol start ditekan, maka stop-watch akan mulai menghitung naik setiap 1/10 detik.
3. Bila tombol start ditekan lagi, maka stop-watch akan berhenti menghitung, dan tampilan angka berhenti sesuai waktu terakhir.
4. Bila tombol reset ditekan, maka tampilan akan kembali menjadi 000.
5. Karena keterbatasan digit, maka setelah 99,9 detik, waktu akan kembali menjadi 000.

1.3.2.1 Desain Rangkaian Elektronika

Pada kasus ini, rangkaian elektronika yang digunakan masih sama dengan modul sebelumnya. Untuk tombol start/stop digunakan tombol pull-up, sedang untuk reset dipakai tombol pull-down.

1.3.2.2 Desain Kode Program Arduino

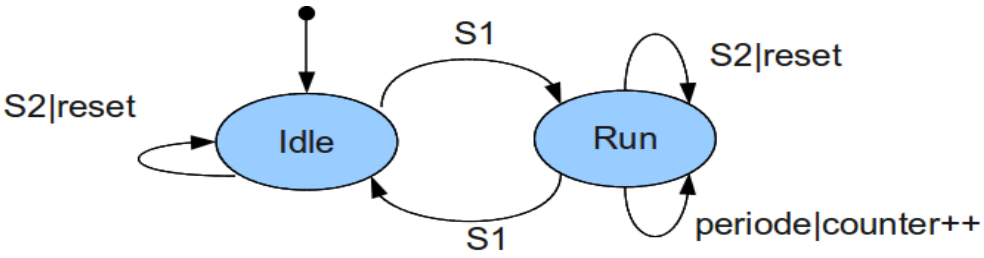
Algoritma khusus yang diperlukan di sini disebut teknik multiplexing. Teknik ini memungkinkan penggunaan jalur luaran digital yang minimal untuk menyalakan 3 buah LED 7-segmen. Jika tidak memakai multiplexing, maka semua akan diperlukan setidaknya 3 x 7 jalur luaran digital (atau 3x4 bila memakai decoder). Dengan multiplexing, ketiga 7-segment akan dinyalakan bergantian dengan cepat. Untuk itu kita menggunakan decoder untuk memicu nyala digit tertentu, dimana contoh sederhana programnya adalah sebagai berikut:

```
// Pemetaan Port Arduino pada Array
// ..............

#define DISPLAY_PERIODE  1000

// fungsi untuk menampilkan BCD ke 7 segment
void writeDigit(int bcd){..................... }

// fungsi memecah angka (0-999) menjadi 3 BCD ke array digits
```

```
// digits[0] = satuan, digits[1] = puluhan, digits[2] = ratusan
int digits[3];
void writeDigit3(int data) {
  digits[0] = data % 10;
  data = data / 10;
  digits[1] = data % 10;
  …. . …. . ..;
  …….. . …. . ..;
}

// fungsi untuk menampilkan digits[] ke 7 segment dengan decoder
int index = 0;
void shiftDigit() {
  writeDigit(digits[index]); // memasukan nilai pada salah satu digits
  digitalWrite(T1, index & 0x02);
  digitalWrite(T2, index & 0x01);
  index = (index + 1 ) % 3;
}

void setup() {
  Serial.begin(115200);
  // inisialisasi mode pin
  for (int pin=BCD1; pin<=SHIFT_CLK; pin++) {
     pinMode(pin, OUTPUT);
  }

  // siapkan angka 123
  writeDigit3(123);

  // Inisialisasi decoder
  digitalWrite(T1, LOW);
  digitalWrite(T2, LOW);
}

void loop() {
  // program menggeser nyala LED
  shiftDigit();
  delay(DISPLAY_PERIODE);
}
```

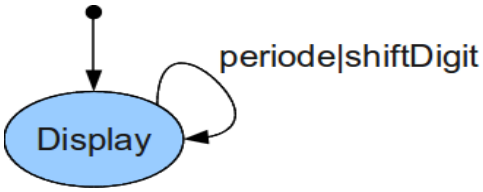
Kode 1. 4. Multiplexing

Setelah paham cara menampilkan angka 3 digit ke 3 buah 7-segmen, kini saatnya membuat program stop-watch. Sesuai dengan cara kerjanya, maka kita rancang diagram transisi keadaaanya sebagai berikut:



Gambar 1.7 Diagram transisi keadaan sistem stop-watch

Selain itu, kita juga harus ingat bahwa 7-segment perlu di-multiplexing setiap periode tertentu. Hal itu bisa dianggap sebagai mesin keadaan yang independent, dengan diagram sebagai berikut:



Gambar 1.8 Diagram transisi keadaan 7 segment banyak digit.

Walau nampaknya rumit, gabungan dua mesin keadaan tersebut mudah diprogram dengan algoritma event-driven sebagai berikut:

```
// Pemetaan Port Arduino
// ..............

// konstanta periode
#define DISPLAY_PERIODE 50
#define CLOCK_PERIODE 100

// fungsi memeriksa apakah tombol S1 ditekan
boolean s1IsPressed() { ..........}

// fungsi memeriksa apakah tombol S2 ditekan
boolean s2IsPressed() { ............}

// fungsi untuk menampilkan BCD ke 7 segment melalui encoder
void writeDigit(int bcd){..............}

// fungsi mengisi 3 digit array BCD dari angka
int digits[3];
void writeDigit3(int angka){..........}

// fungsi untuk menampilkan 3 digit BCD ke 7-segment memakai shift register
void shiftDigit() {.......}

// variabel global
#define STATE_IDLE    0
#define STATE_RUN     1
int counter = 0;
int state=0;
int next_state=0;
boolean s1, s2;
```

```
long next_clock, next_display;

void setup() {
  Serial.begin(115200);
  // inisialisasi mode pin
  // ...........

  // inisialisasi digits
  writeDigit3(0);

  // inisialisasi shift digit
  digitalWrite(T1, LOW);
  digitalWrite(T2, LOW);

  // inisialisasi waktu
  next_clock = next_display = millis();
}

void eventS1() {
  Serial.print("Event S1 - state ");
  Serial.println(state);
  switch(state) {
     case STATE_IDLE :
       // ubah state
       next_state = STATE_RUN;
       break;
     case STATE_RUN:
       // ubah state ......
  }
}

void eventS2() {
  Serial.print("Event S2 - state ");
  Serial.println(state);
  switch(state) {
    case STATE_IDLE:
    case STATE_RUN:
       // reset counter
       counter = .;
       writeDigit3(counter);
  }
}

void eventClock() {
  Serial.print("Event Clock - state ");
  Serial.println(state);
  switch(state) {
    case STATE_IDLE: break;
    case STATE_RUN:
       // naikkan counter 0 - 999
       counter = (..... . .) . ....;
       writeDigit3(counter);
  }
}
```

```
void loop() {
  // baca semua input
  s1 = .............;
  s2 = .............;
  // baca waktu dalam mili detik
  long t = ........;

  // proses event
  // yang prioritasnya tinggi didahulukan
  if (t >= next_clock) {
    eventClock();
    next_clock = next_clock + CLOCK_PERIODE;  // waktu aksi berikutnya
  }
  else if (s1) {
     eventS1();
  }
  else if (s2) {
     .........;
  }
  else if (t >= next_display) {
    shiftDigit();
    next_display = ............ . .............; // waktu aksi berikutnya
  }

  // pelaporan dan update state
  if (next_state != state) {
     // ..............
  }
}
```

Kode 1. 5. Stopwatch

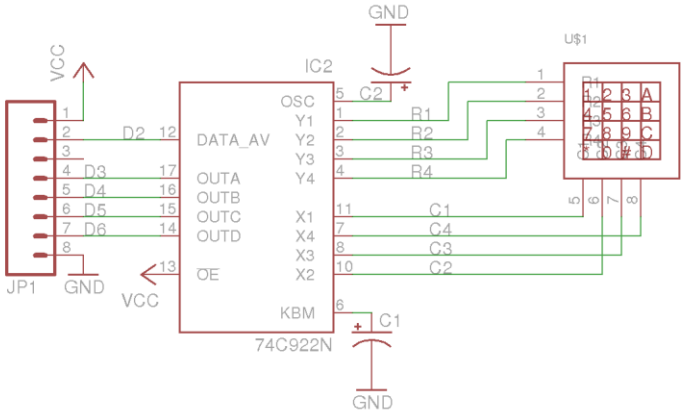
1.3.3 Keypad

Keypad adalah piranti masukan banyak tombol, dimana tiap tombol melambangkan digit 0 - 9, serta sisanya bisa melambangkan simbol #, * maupun huruf A, B, C, D. Konstruksi keypad sesungguhnya adalah switch yang diletakkan pada suatu matrix kolom dan baris, sehingga luarannya adalah konektor sebanyak jumlah kolom dan baris. Apabila suatu keypad 4x4 dikoneksikan ke arduino secara langsung, maka akan diperlukan 8 pin I/O. Untuk menghemat pin I/O, sekaligus menyederhanakan pemrograman, maka keypad ini sebaiknya disambung dulu ke sebuah encoder, baru dimasukkan ke arduino.

1.3.3.1 Desain Rangkaian Elektronika

Pada kasus ini, akan digunakan rangkaian elektronika seperti Gambar 1.9. Nampak disini bahwa dari keypad ada luaran 4 baris (R1-R4) dan 4 kolom (C1-C4). Bila suatu tombol ditekan maka baris dan kolom yang sesuai akan memberi sinyal HIGH. Misalnya jika ditekan angka 6, maka R2 dan C3 akan bernilai HIGH, sementara yang lain tetap LOW. Luaran 8 bit itu diumpankan ke chip 74*922, sehingga akan menghasilkan kode 4 bit (OUTA - OUTD) yang menjadi masukan ke arduino (D3-D6). Translasi sinyal baris dan kolom itu ke kode bisa dilihat di data-sheet. Selain itu chip 74*922 juga memiliki output DATA_AV yang akan bernilai HIGH bila salah satu tombol sedang ditekan. Sinyal ini

dimasukkan ke pin D2 arduino.



Gambar 1.9. Rangkaian keypad dengan encoder.

1.3.3.2 Desain Kode Program Arduino

Dengan dipakainya encoder, kita membaca keypad bila sinyal DATA_AV berubah dari LOW ke HIGH. Kode 4 bit dari encoder selanjutnya dibaca dan dimasukkan ke variabel. Masalah kita disini, kode tersebut adalah sesuai dengan posisi tombol, bukan lambang pada tombol. Jadi kita perlu menerjemahkannya. Untuk menelusurinya, silahkan coba program berikut:

```
// Pemetaan Port Arduino untuk keypad
#define DATA_AV  2
#define KEY0     4

// fungsi memeriksa apakah keypad ditekan
// kalau ditekan jadi LOW, maka seperti pull-up
boolean keyIsPressed() {
  static boolean key_old = 0;
  boolean b1 = digitalRead(DATA_AV);
  boolean s1 = (!b1 && key_old);;
  key_old = b1;
  return s1;
}

// fungsi membaca kode key
// dipanggil bila keyIsPressed sudah mengembalikan TRUE
// ini pakai cara gampang, langsung or per bit
int readKey() {
  int key = digitalRead(KEY0);
  key |= digitalRead(KEY0+1) << 1;
  key |= digitalRead(KEY0+2) << 2;
  key |= digitalRead(KEY0+3) << 3;
  return key;
}

// fungsi menerjemahkan scan-code keypad (0-15),
// sehingga pas kalau ditekan '0' jadi biner 0,
// ditekan '1' jadi biner 1, dst sampai '9'.
```

```
// Lalu kalau ditekan 'A' jadi 0xA, dst sampai 'D'.
// Lalu '*' = 0xE, '#' = 0xF.
// Memakai array look-up table, ini salah, harap dibetulkan
char biner_table[] = {1,2,3,4,5,6,7,8,9,0xA,0xB,0xC,0xD,0xE,0xF,0};
char keyToBiner(int key) {
  return biner_table[key];
}

void setup() {
 Serial.begin(115200);
 // inisialisasi mode pin
 for (int pin=DATA_AV; pin<=KEY0+3; pin++) {
   pinMode(pin, INPUT);
 }
}

void loop() {
 if (keyIsPressed) {
   int key = readKey();
   Serial.print("Key code = ");
   Serial.print(key, HEX);
   Serial.print(" Biner = ");
   Serial.println(keyToBiner(key), HEX);
 }
}
```

Kode 1. 6. Keypad

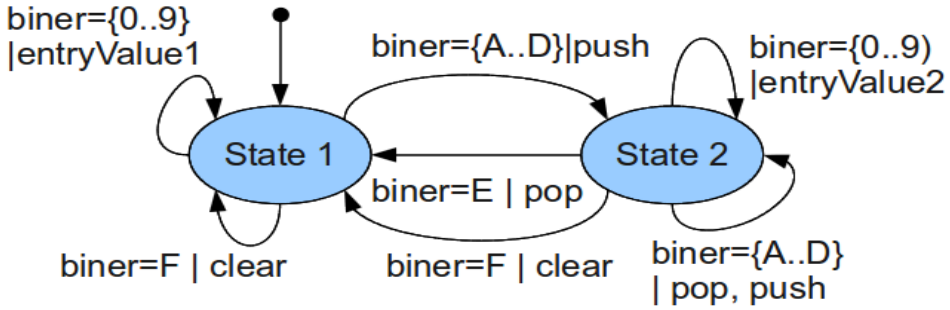
Setelah paham proses membaca keypad, kini kita coba membuat program yang berguna, yakni kalkulator jadul. Cara kerjanya :

1. Dalam keadaan awal, maka kalkulator akan menampilkan angka 000.
2. Orang bisa memasukkan digit pada keypad, maka kalkulator akan mencatatnya sebagai angka (misal jika ditekan 1 lalu 2, maka akan dicatat sebagai 12).
3. Jika kemudian ditekan simbol operator, maka kalkulator mengingat angka terakhir, dan jenis operatornya. Dalam hal ini, operatornya kita pakai :

A	tambah	C	kali
B	kurang	D	bagi

4. Kemudian orang bisa memasukkan angka kedua.
5. Jika orang menekan operator, maka kalkulator akan mengoperasikan angka1 dan angka2, lalu mengingat operator terakhir dan siap menerima angka2 lagi.
6. Jika orang menekan tombol hitung (pakai ke '*'), maka kalkulator akan mengoperasikan angka1 dan angka2, lalu menampilkan hasilnya.
7. Jika orang menekan tombol clear (pakai simbol '#'), maka kalkulator kembali ke awal.

Diagram transisi keadaan sistem kalkulator ini nampak sebagai berikut:



Gambar 1.10. Diagram transisi keadaan kalkulator post-fix

Nah, ini dia programnya memakai algoritma state-driven.

```
// Pemetaan Port Arduino untuk 7 segmen
// ..............
// Pemetaan Port Arduino untuk keypad
// ..............

// konstanta periode
#define DISPLAY_PERIODE 50

// konstanta OPERATOR
#define ADD  0xA
#define SUB  0xB
#define MUL  0xC
#define DIV  0xD
#define CLR  0xE
#define CALC 0xF

// fungsi memeriksa apakah keypad ditekan
boolean keyIsPressed() { ..........}

// fungsi membaca keypad
int readKey() { ............}

// fungsi mentranslasi kode keypad (0-15) menjadi biner (0 - 0xF)
int keyToBiner(int key) {............}

// fungsi untuk menampilkan BCD ke 7 segment melalui encoder
void writeDigit(int bcd){..............}

// fungsi mengisi 3 digit array BCD dari angka
int digits[3];
void writeDigit3(int angka){..........}

// fungsi untuk menampilkan 3 digit BCD ke 7-segment memakai shift register
void shiftDigit() {.......}
// fungsi mengisi 3 digit array BCD dengan operator
void writeOperator(int op) {
  digit[0] = op;  // op akan sesuai hexa a,b,c,d
  digit[1] = 15;  // tidak tampil apa-apa
  digit[2] = 15;  // tidak tampil apa-apa
```

```
}

// variabel global
#define STATE_1    0
#define STATE_2    1
int state=0;
int next_state=0;
boolean da;
int key, biner;
int value1, value2;
int opr;
long next_display;

void setup() {
  Serial.begin(115200);
  // inisialisasi mode pin
  // ...........

  // inisialisasi digits & shift 7-segment
  // ........

  // inisialisasi waktu
  next_display = millis();

  // inisialisasi kalkulator
  biner1 = biner2 = 0;
}

// fungsi transisi catatValue1
// menggubah harga value1 sesuai masukan biner,
// tapi lakukan hanya kalau tidak menyebabkan overflow
void entryValue1(int biner) {
   if (value1 < 100) {
      value1 = (value1 * 10) + biner;
      writeDigit3(value1);
   }
   Serial.print(" value1= ");
   Serial.println(value1);
}

// fungsi transisi catatValue2
// menggubah harga value2 sesuai masukan biner
void entryValue2(int biner) {
   // ...................
}

// fungsi transisi push
// mengingat operator yang ditekan
void push(int op) {
   Serial.print(" push ");
   Serial.println(value1);
   opr = op;
   writeOperator(opr);
}
```

```
// fungsi transisi pop
// mengoperasikan value1 thd value2 sesuai operator
void pop() {
  Serial.print(" pop ");
  Serial.print(value1);
  Serial.print(" ");
  Serial.print(opr, HEX);
  Serial.print(" ");
  Serial.print(value2);
  switch(opr) {
    case ADD: value1 = value1 + value2; break;
    case SUB: value1 = ..... . .....; break;
    case MUL: value1 = ..... . .....; break;
    case DIV: value1 = ..... . .....; break;
  }
  value2 = 0;
  writeDigit3(value1);
  Serial.print(" => ");
  Serial.println(value1);
}

// fungsi transisi clear
void clear() {
 value2 = value1 = 0;
 writeDigit3(value1);
 Serial.println(" clear");
}

// fungsi keadaaan state1
// sesuai diagram transisi keadaan 1.10
void state1() {
 Serial.print("State 1 ");
 switch(biner) {
   case 0:
   case 1:
   case 2:
   case 3:
   case 4:
   case 5:
   case 6:
   case 7:
   case 8:
   case 9:
     entryValue1(biner); break;
   case ADD:
   case SUB:
   case MUL:
   case DIV:
     push(biner);
     next_state = STATE_2;
     break;
   case CLR:
     clear();
     break;
 }
```

```
}

// fungsi keadaaan state2
// sesuai diagram transisi keadaan 1.10
void state2() {
  // ..........................
}

void loop() {
  // baca semua input
  da = keyIsPressed();
  // baca waktu dalam mili detik
  long t = millis();

  if (da) {
    // baca key dan translasi ke biner
    key = readKey();
    biner = keyToBiner(key);

    // tampilkan untuk monitoring
    Serial.print("Key = ");
    Serial.print(key, BIN);
    Serial.print(" = ");
    Serial.println(biner, HEX);

    // proses sesuai state
    switch(state) {
      case STATE_1 : state1(); break;
      case STATE_2 : state2(); break;
    }
  }
  else if (t > next_display) {
    shiftDigit();
    next_display += DISPLAY_PERIODE;
  }

  // update state
  if (next_state != state) {
    state = next_state;
  }
}
```

Kode 1. 7. Calculator

Woah, ruwet juga ternyata teknologi kalkulator jadul ini. Padahal, program diatas belum mengantisipasi kejadian terlarang seperti overflow, underflow, maupun pembagian dengan nol. Makanya, jangan pandang enteng orang jaman dahulu.

1.4 Tugas Awal

Kerjakan tugas awal berikut, harus dibawa saat akan praktikum

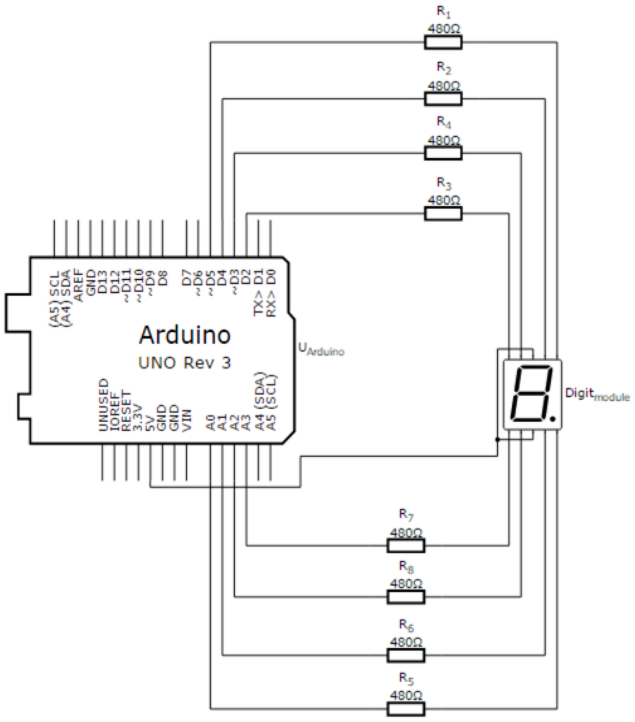
1. Jelaskan secara umum tentang istilah decoder vs encoder pada sistem digital.
2. Cari data-sheet chip 74*47, salin diagram blok dan tabel kebenarannya.
3. Cari data-sheet chip 74*922, salin diagram blok dan tabel kebenarannya.
4. Cari data-sheet chip 74*194, salin diagram blok dan tabel kebenarannya.
5. Jelaskan cara kerja chip 74*47 pada rangkaian 1-3, dan kaitannya dengan fungsi writeDigit() pada *Kode 1.1. Seven Segment - Counter*. Berikan komentar baris per baris.

```
// fungsi untuk menampilkan digit 0-9 ke 7 segment melalui encoder
void writeDigit(int bcd){
  for(int i=0; i<4; i++) {       // mengapa i looping dari 0 sd 3
    digitalWrite(BCD1 + i, bcd & 0x01); // mengapa BCD+i dan pakai operator &
    bcd = bcd >> 1; // mengapa pakai operator >>
  }
}
```

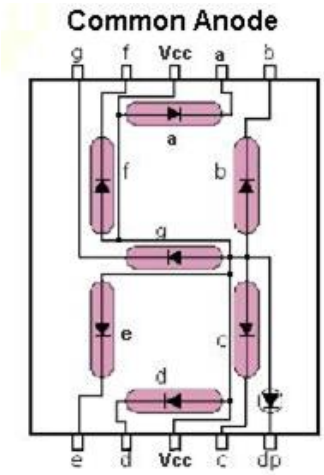
6. Jelaskan cara kerja chip 74*194 pada rangkaian 1-3, dan kaitannya dengan fungsi shiftDigit() pada *Kode 1.4. Multiplexing* . Berikan komentar baris per baris.
7. Pelajari cara kerja chip 72*922 pada rangkaian 1-9, lalu jelaskan fungsi keyIsPressed, readKey, dan keyToBiner yang ada pada *Kode 1.6. Keypad*. Berikan komentar baris per baris.
8. Lengkapi kode sumber Counter1.pde.
9. Lengkapi kode sumber Stopwatch3.pde.
10. Lengkapi kode sumber Calc.pde.

11. Buatlah Rangkaian seperti gambar di bawah ini :

- Komponen yang dibutuhkan :
  - Arduino Uno
  - Kabel USB
  - Breadboard
  - 8 buah resistor 480 Ohm
  - 1 buah 7 Segment tipe *Common-Anodhe*
  - Jumper
- Program :
  - Buatlah program untuk menampilkan angka 0 – 9 dengan jeda 1 detik.
  - Petunjuk :
    - hubungkan antara 7 segment dan arduino :



Port Arduino	Kaki 7 Segment
D2	G
D3	F
D4	A
D5	B
A0	E
A1	D
A2	C
A3	DP
VCC1 dan VCC2	5 volt @ Arduino



1.5 Praktikum

Bersiap-siaplah melakukan praktikum dengan mulai:

- Nyalakan komputer.
- Koneksikan Arduino dengan USB ke komputer.
- Jalankan Arduino IDE.

- Coba unggah contoh program blink untuk memastikan Arduino sudah siap.

1.5.1 Pencacah 1 Digit

1. Siapkan rangkaian arduino dengan 7-segment dan tombol seperti dijelaskan pada bagian 1.3.1.
2. Cobalah *Kode 1.1. Seven Segment - Counter*, dan pahami cara kerja decoder BCD to 7 segment. Ubahlah modulo digit dari 10 ke 16. Catat hasilnya pada jurnal 1.1.
3. Ketikkan *Kode 1.2. Counter – State driven* pada Arduino IDE, kompilasi, dan download.
4. Lakukan percobaan dan isikan pada jurnal 1.2.
5. Jika sudah merasa benar, catat kode sumber anda yang sudah final.
6. Lakukan hal yang sama dengan *Kode 1.3. Counter – Event driven* Isi jurnal 1.3.

1.5.2 Stop-watch 3 Digit

1. Gunakan rangkaian sebelumnya.
2. Cobakan program *Kode 1.4. Multiplexing*, pahami cara kerja chip shift register. Edit kembali kode sumber tersebut, kecilkan DISPLAY_PERIODE (misal 100, 50, 25, ...) sampai LED 7 segmen nampak tidak berkedip. Cobalah cari DISPLAY_PERIODE terbesar yang membuat LED menampilkan angka 888 tanpa terlihat berkedip. Setelah dapat, tulis hasilnya di jurnal 1.2.1.
3. Ketikkan *Kode 1.5. Stopwatch* pada Arduino IDE, kompilasi, dan download.
4. Lanjutkan percobaan sesuai skenario normal di jurnal 1.2.2, catat hasilnya.
5. Catat kode sumber anda yang sudah final.

1.5.3 Keypad untuk Calculator

1. Gunakan rangkaian seven segment dan rangkaian keypad.
2. Cobalah *Kode 1.6. Keypad*. Setelah di-unggah, jalankan serial monitor lalu lihat tampilannya ketika keypad ditekan. Catat look-up table yang benar.
3. Ketikkan *Kode 1.7. Calculator* yang sudah anda siapkan. Unggah lalu lakukan percobaan sesuai jurnal 1.3, catat hasilnya.
4. Catat kode sumber anda yang sudah final.

1.6 Laporan

Buat laporan sesuai dengan petunjuk pada pengantar. Untuk analisis per percobaan, jawab pertanyaan-pertanyaan berikut.

1.6.1 Pencacah 1 Digit

1. Pada rangkaian 1-2, terlihat ada 7 resistor pada setiap katoda 7-segment yang terkoneksi ke arduino (plus satu untuk dot). Sementara itu pada rangkaian 1-3, ketujuh resistor itu diganti dengan 3 resistor yang terhubung ke common anoda 7-segment dari kolektor tiap transistor. Jelaskan bedanya, apa untung ruginya ? Apa anda dapat mengamatinya adanya efek resistor itu selama digit berubah dari 0 - 9 ?
2. Berikan komentar tentang algoritma state-driven vs event-driven. Apa untung ruginya ?

1.6.2 Stop-watch

1. Berapakah DISPLAY_PERIODE terbesar yang optimal agar LED tampak tidak berkedip ? Carilah teori yang mendukung hasil itu !
2. Pada rangkaian 1-3, ada 3 buah LED 7-segment dengan konfigurasi common-anoda. Jelaskan fungsi dari transistor T1, T2 maupun T3 yang memakai tipe NPN dan kaitannya dengan output 74*47 yang bersifat open collector. Mungkinkah dipakai transistor tipe PNP ?
3. Jelaskan mengapa pada rangkaian 1-3, pada chip 74*194 kaki A terhubung ke VCC sementara kaki B,C,D, dan SR terhubung ke GND ? Mengapa pula S1 mendapat ouput D12 dari arduino sementara S0 disambung ke VCC ?
4. Pada pemrosesan event di fungsi loop(), ada komentar bahwa prioritas tinggi didahulukan. Apa maksudnya ? Apa konsekuensinya seandainya programnya demikian :

```
// proses event
if (t > next_display) {
  shiftDigit();
  next_display += .....;
}
else if (t >= next_clock) {
  eventClock();
  next_clock += CLOCK_PERIODE;
}
else if (s1) eventS1();
else if (s2) eventS2();
```

1.6.3 Keypad

1. Setelah melakukan percobaan keypad.pde, bagaimanakah tabel translasi biner_table yang benar ? Sesuaikah itu dengan datasheet 74*922 ?
2. Menurut anda apakah *Kode 1.7. Calculator* akan lebih baik jika memakai event-driven ? Jelaskan alasan anda.
3. Coba cari beberapa kekurangan program ini dan berikan ide anda untuk memperbaikinya.

1.7 Referensi

- 7 Segment, http://extremeelectronics.co.in/avr-tutorials/multiplexed-seven-segment-displays/
- Counter, http://en.wikipedia.org/wiki/Counter
- Stop-watch, http://en.wikipedia.org/wiki/Stopwatch
- Calculator, http://en.wikipedia.org/wiki/Calculator

JURNAL PRAKTIKUM TF2207

MODUL 1

Tanggal		Nomor Kit Modul	
NIM Peserta			

Jurnal 1.1. Tampilan 7-Segmen (Kode 1.1)

Untuk masing-masing angka, isikan tampilan seven-segment, dan tingkat kecerahannya:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Jurnal 1.2. Pencacah 1 Digit (State-driven) (Kode 1.2)

S1 = Tombol cacah; S2=tombol mode, LED1 = 7-Segmen pertama

No	Perlakuan	Yang Diharapkan	Hasil
1	Arduino mulai	LED1 = 0	
2	Tekan S1	LED1 =1	
3	Tekan S1 8 kali	LED1 =9	
4	Tekan S1 sekali lagi		
5	Tekan S2		
6	Tekan S1		
7	Tekan S1 8 kali		
8	Tekan S1 sekali lagi		
9	Tekan S2		
10	Tekan S1		

Jurnal 1.3. Pencacah 1 Digit (Event-driven) (Kode 1.3)

S1 = Tombol cacah; S2=tombol mode, LED1 = 7-Segmen pertama

No	Perlakuan	Yang Diharapkan	Hasil
1	Arduino mulai	LED1 = 0	
2	Tekan S1	LED1 =1	
3	Tekan S1 8 kali	LED1 =9	
4	Tekan S1 sekali lagi		
5	Tekan S2		
6	Tekan S1		
7	Tekan S1 8 kali		
8	Tekan S1 sekali lagi		
9	Tekan S2		
10	Tekan S1		

Jurnal 1.2.1 Pencacah 3 Digit (Coba DISPLAY_PERIODE) (Kode 1.4)

LED1, LED2, LED3 = 7-Segmen pertama,kedua,ketiga

No	Perlakuan	Yang Diharapkan	Hasil
1	Upload program dengan DISPLAY_PERIODE=1000 ms	LED1 - LED3 = 123 Berkedip	
2	Upload program dengan DISPLAY_PERIODE=100 ms		
3	Upload program dengan DISPLAY_PERIODE= 50		
4	Upload program dengan DISPLAY_PERIODE=		
5	Upload program dengan DISPLAY_PERIODE=		
6	Upload program dengan DISPLAY_PERIODE=....	LED0 - LED2 = 123 Tidak berkedip	

Jurnal 1.3.2 Stop-watch 3 Digit (Kode 1.5)

S1 = tombol start/stop; S2 = tombol reset; LED = 3 digit 7 segment

No	Perlakuan	Yang Diharapkan	Hasil
1	Arduino mulai	LED = 000	
2	Tekan S1	LED akan mencacah naik dengan periode 10 ms	
3	Tekan S1	LED berhenti mencacah	
4	Tekan S1		
5	Tekan S1		
6	Tekan S2		
7	Tekan S1		
8	Biarkan lama sampai clock mencapai 999 lalu melewatinya.		
9	Tekan S1		

No	Perlakuan	Yang Diharapkan	Hasil
10	Tekan S2		

Jurnal 1.3.1 Calculator (Kode 1.7)

Key = keypad; LED = 3 digit 7 segment

No	Perlakuan	Yang Diharapkan	Hasil
1	Arduino mulai	LED = 000	
2	Key = 1, 2	LED = 012	
3	Key = A	LED = a	
4	Key = 3, 4	LED = 034	
5	Key = #	LED = 36	
6	Key = B		
7	Key = 3, 4		
8	Key = C		
9	Key = 5		
10	Key = D		
11	Key = 4		
12	Key = #		
13	Key = D		
14	Key = 0		
15	Key = #		
16	Key = *		
17	Key = B		
18	Key = 8		

No	Perlakuan	Yang Diharapkan	Hasil
19	Key = #		