Modul 1: DIGITAL INPUT/OUTPUT

**Selvia 13318031, Kelompok: 18**

Tanggal: 31 Januari 2020, Assisten: Krisna Diastama 13317016

TF 2207 - Laboratorium Teknik Fisika II

# Tujuan Percobaan

1. Menentukan kode program *Seven Segment-Counter*
2. Menentukan kode program *Counter – State Driven*
3. Menentukan kode program *Event – State Driven*
4. Menentukan kode program Multiplexing
5. Menentukan kode program stopwatch
6. Menetukan kode program keypad
7. Menentukan kode program calculator

# Alat dan Bahan

Table 2.1 Alat dan Bahan Praktikum

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |

# Teori Dasar

Untuk mendesain masukan atau keluaran suatu mikroprosesor, spesifikasi yang sangat menentukan adalah jumlah pin I/O dan jenisnya misalnya analog, digital, PWM, dan lainnya. Pada Arduino uno terdapat 20 buah pin I/O yaitu A0 - A5 dan D0 - D13 dengan konfigurasi sebagai berikut:

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)

Standar sinyal yang diikuti Arduino adalah transistor-transistor logic TTL:

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

2. Sinyal LOW : listrik dibawah 0,8 Volt.

Encoder adalah sirkuit kombinasional yang mengubah informasi biner dalam bentuk garis input 2N menjadi garis output N, yang mewakili kode bit N untuk input. Untuk pembuat enkode sederhana, diasumsikan bahwa hanya satu jalur input yang aktif pada satu waktu. Sedangkan dekoder berkebalikan dari pembuat enkode. Decoder adalah adalah sirkuit kombinasional yang mengubah garis input N menjadi garis output 2N.[[1]](#footnote-1)

Pada praktikum ini digunakan beberapa chip yaitu chip 74LS47, 7402N, 74C922 dan 74\*194. Chip tersebut mempunyai fungsi yang berbeda-beda. Chip 74LS47 berfungsi untuk 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 tersebut. Chip 74C922 digunakan pada keypad. Chip ini akan menerima umpan yang berupa luaran 8 bit sehingga akan menghasilkan 4 bit (OUTA-OUTD) yang menjadi masukan ke Arduino (D3-D6). Selain itu chip ini juga memiliki output DATA\_AV yang akan bernilai HIGH ketika salah satu tombol ditekan. Chip 74\*194 berfungsi untuk menjadi shift register dan berkaitan dengan fungsi shiftDigit() adalah cara kerja fungsi tersebut sama dengan shift register yaitu menggeser data dalam register agar dapat dibaca secara bergantian.

# Data dan Analisis

## Pencacah 1 Digit

Pada praktikum ini terdapat tiga kode program yang telah dijalankan yaitu kode1.1 *Seven Segment – Counter*, kode 1.2 *Counter – State Driven*, dan kode 1.3 *Counter-Event Driven*. Berikut ini merupakan kutipan dari setiap kode-kode tersebut.

Kode 1.1 *Seven Segment – Counter*

void writeDigit(int bcd) {

for (int i = 0; i < 4; i++) {

digitalWrite(BCD1 + i, bcd & 0x01);

bcd = bcd >> 1;

}

}

void setup() {

Serial.begin(9600);

pinMode(BUTTON1, INPUT);

pinMode(BUTTON2, INPUT);

for (int pin = BCD1; pin <= T2; pin++) {

pinMode(pin, OUTPUT);

}

digitalWrite(T1,HIGH);

digitalWrite(T2,HIGH);

}

void loop() {

for(int i = 0; i<16; i++){

writeDigit(i);

delay(1000);

}

}

Kode 1.2 *Counter – State Driven*

void setup() {

Serial.begin(9600);

pinMode(BUTTON1, INPUT);

pinMode(BUTTON2, INPUT);

for (int pin = BCD1; pin <= T2; pin++) {

pinMode(pin, OUTPUT);

}

next\_state = state = 0;

counter = 0;

writeDigit(counter);

digitalWrite(T1, index & 0x02);

digitalWrite(T2, index & 0x02);

}

void loop() {

// put your main code here, to run repeatedly:

s1 = s1IsPressed();

s2 = s2IsPressed();

switch (state) {

case 0 : state0(); break;

case 1 : state1(); break;

}

if (next\_state != state) {

Serial.print("State ");

Serial.println(state);

Serial.print(" ==> ");

Serial.println(next\_state);

state = next\_state;

}

}

void eventS2() {

Serial.print("Event S2 state ");

Serial.println(state);

switch (state) {

case STATE\_UP:

next\_state = STATE\_DOWN; break;

case STATE\_DOWN:

next\_state = STATE\_UP; break;

}

}

void setup() {

Serial.begin(9600);

pinMode(BUTTON1, INPUT);

pinMode(BUTTON2, INPUT);

for (int pin = BCD1; pin <= T2; pin++) {

pinMode(pin, OUTPUT);

}

next\_state = state = 0;

counter = 0;

writeDigit(counter);

digitalWrite(T1, index & 0x02);

digitalWrite(T2, index & 0x02);

}

void loop() {

// put your main code here, to run repeatedly:

s1 = s1IsPressed();

s2 = s2IsPressed();

if (s1) eventS1();

else if (s2) eventS2();

if (next\_state != state) {

Serial.print("State ");

Serial.println(state);

Serial.print(" ==> ");

Serial.println(next\_state);

state = next\_state;

}

}

Kode 1.3 *Counter – Event Driven*

Pada kode 1.1 *Seven Segment – Counter* menghasilkan output 7 segment yang menyala sesuai dengan datasheet yang diberikan pada modul dengan masukan yang diberikan yaitu 4 bit BCD (Binary Code Decimal). Selain itu, dari output yang dihasilkan, data tingkat kecerahan LED juga diambil dengan skala penilaian dari 0-4 dengan 0 berarti nyala LED paling redup dan 4 berati nyala LED paling terang. Kelompok kami mendapatkan data kecerahan setiap LED dari angka 1-15 yaitu nilai 4. Sementara itu, untuk kelompok lain yaitu kelompok 17 juga mendapatkan nilai 4 utnuk data tingkat kecerahan LED pada praktikum ini. Sedangkan pada kelompok 19 dan 20 mendapatkan nilai 3 untuk data tingkat kecerahan LED pada praktikum ini. Untuk data lengkapnya dapat dilihat pada file jurnal modul 1 bagian Jurnal 1.1. Tampilan 7-Segmen (Kode 1.1) yang terlampir. Resistor yang dipasang pada anoda dan katoda memiliki perbedaan. Jika resistor dipasang pada masing-masing aktoda maka akan membutuhkan banyak pin sehingga pin yang tersisa menjadi sedikit dan dapat menjadi kendala ketika ingin menambah rangkaian. Sedangkan jika resistor dipasang pada anoda akan terhubung ke common anoda 7 segment dari kolektor tiap transisitor sehinggan jumlah pin yang digunakan relative sedikit jika disbanding pemasangan resisitor pada katoda. Selain itu, pemasangan resistor pada anoda juga mempunyai kelemahan yaitu hanya satu digit yang menyala karena decoder hanya mengijinkan salah satu transistor yang aktif.

Program dengan kode 1.2 dan 1.3 dapat bekerja sesuai harapan. Berikut ini merupakan data hasil program dari kode 1.2 dan 1.3.

Tabel 4.1 Data State Driven dan Event Driven

|  |  |  |  |
| --- | --- | --- | --- |
| No | Perlakuan | Yang Diharapkan | Hasil |
| 1 | Arduino mulai | LED1=0 | LED1=0 |
| 2 | Tekan S1 | LED1=1 | LED1=1 |
| 3 | Tekan S1 8 kali | LED1=9 | LED1=9 |
| 4 | Tekan S1 sekali lagi | LED1=0 | LED1=0 |
| 5 | Tekan S2 | state down | state down |
| 6 | Tekan S1 | LED1=9 | LED1=9 |
| 7 | Tekan S1 8 kali | LED! = 1 | LED! = 1 |
| 8 | Tekan S1 sekali lagi | LED1 = 0 | LED1 = 0 |
| 9 | Tekan S2 | state up | state up |
| 10 | Tekan S1 | LED1= 1 | LED1= 1 |

Program dari kode 1.2 dan 1.3 pada dasarnya menghasilkan output yang sama. Perbedaan dari keduanya terlihat pada alur pemogramannya. Pada kode 1.2 menggunakan *state-driven* dan pada kode 1.3 menggunakan *event-driven*. Pada alur pemograman *state-driven* memiliki algoritma yang memfokuskan pada *state* yang mana setiap *state* didefinisikan menjadi dua kondisi yaitu *mode up* dan *mode down*. Sedangkan padaalur pemograman kode 1.3 memiliki algoritma yang memfokuskan pada perubahan event. Pada program tersebut terdapat dua event yaitu ketika S1 ditekan dan ketika S2 ditekan.

## Stopwatch 3 Digit

Pada praktikum stopwatch 3 digit akan dijalankan dua program yaitu program kode 1.4 dan 1.5. Pada kode 1.4 berkaitan dengan multiplexing yang diharapkan akan menghasilkan keluaran berupa LED pada 7 segment menyala bergantian dengan cepat. Dengan fungsi DISPLAY\_PERIODE dapat diambil data berupa priode yang optimal agar LED pada ketiga 7 segment tidak berkedip dan terlihat dengan jelas angka pada ketiga 7 segment. Berikut ini data yang diambil dari program kode 1.4.

Tabel 4.2.1 Data Program Kode 1.4 Multiplexing

|  |  |  |  |
| --- | --- | --- | --- |
| No | Perlakuan | Yang Diharapkan | Hasil |
| 1 | Upload program dengan DISPLAY\_PERIODE=1000 ms | LED1-LED3 = 123 berkedip | LED1-LED3 = 123 berkedip |
| 2 | Upload program dengan DISPLAY\_PERIODE=100 ms | LED1-LED3 = 123 berkedip | LED1-LED3 = 123 berkedip |
| 3 | Upload program dengan DISPLAY\_PERIODE= 50 ms | LED1-LED3=123, tidak berkedip | LED1-LED3 = 123 berkedip |
| 4 | Upload program dengan DISPLAY\_PERIODE= 22 ms | LED1-LED3=123, tidak berkedip | LED1-LED3 = 123 berkedip |
| 5 | Upload program dengan DISPLAY\_PERIODE= 10 ms | LED1-LED3=123, tidak berkedip | LED1-LED3 = 123 berkedip |
| 6 | Upload program dengan DISPLAY\_PERIODE= 7 ms | LED1-LED3=123, tidak berkedip | LED1-LED3=123, tidak berkedip |

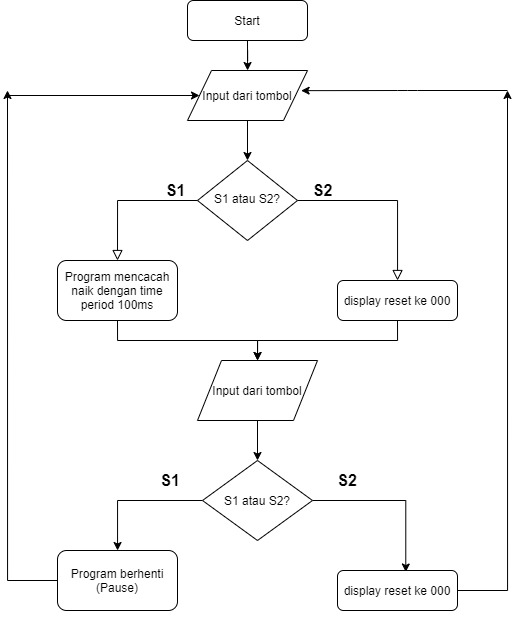
Dari data tersebut dapat diketahui bahwa periode optimal yang dibutuhkan agar LED pada ketiga 7 segment tidak berkedip dan terlihat dengan jelas angka pada ketiga 7 segment adalah 7 ms. Sementara pada kelompok lain yaitu kelompok 17, 19, dan 20 didapatkan priode optimal dari program tersebut adalah 5 ms, 8 ms, dan 5 ms.

Pada program kode 1.5 berkaitan dengan stopwatch. Berikut ini data yang diambil dari program kode 1.5 ketika dijalankan.

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

Tabel 4.2.2 Data Program Kode 1.5 Stopwatch

Dari data praktikum tersebut dapat diketahui bahwa program kode 1.5 stopwatch berjalan sesuai dengan harapan. Selain itu, dari pemograman pada kode 1.5 diperoleh flowchart sebagai berikut.



Gambar 4.2 flowchart pada kode 1.5

## Keypad

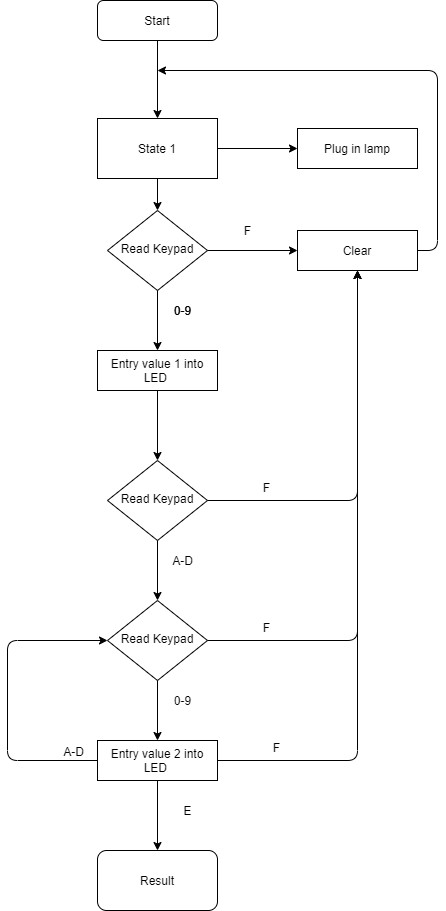
Pada praktikum keypad akan dijalankan dua program yaitu kode 1.6 keypad dan 1.7 calculator. Pada program kode 1.6 keypad berjalan sesuai harapan namun tabel translasi biner\_table yang benar berdasarkan praktikum yang telah kelompok kami lakukan berbeda dengan datasheet 74\*922 yang telah diberikan pada modul. Tabel translasi biner\_table pada datasheet di modul yaitu biner\_table[] = {1,2,3,4,5,6,7,8,9,0xA,0xB,0xC,0xD,0xE,0xF,0}. Ketika data pada biner\_table dimasukkan pada program tersebut menghasilkan output yang tidak sesuai. Sedangkan biner\_table yang benar sesuai dengan output yang diharapkan pada praktikum kali ini yaitu biner\_table[] = {0xD, 0xE, 0, 0xF, 0xC, 9, 8, 7, 0xB, 6, 5, 4, 0xA, 3, 2, 1}. Hal ini dapat terjadi karena pemasangan pin pada Arduino yang tidak sesuai yang mengakibatkan input yang diberikan dan output yang dihasilkan tidak sesuai.

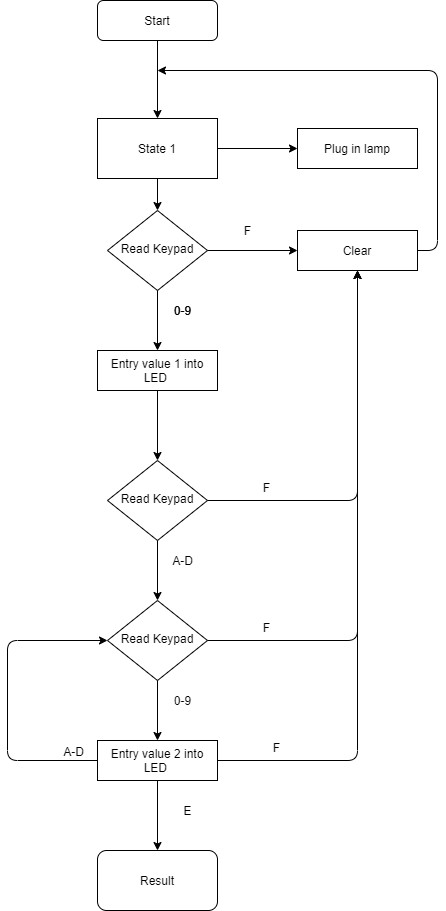
Program kode 1.7 calculator dapat berjalan sesuai dengan harapan. Hal tersebut dapat dilihat dari data yang diperoleh ketika program dijalankan. Data tersebut dapat dilihat pada table di bawah ini.

Tabel 4.3 Program Kode 1.7 Calculator

|  |  |  |  |
| --- | --- | --- | --- |
| No | Perlakuan | Yang Diharapkan | Hasil |
| 1 | Arduino mulai | LED = 001 | LED = 001 |
| 2 | Key = 1, 2 | LED = 012 | LED = 012 |
| 3 | Key = A | LED=a | LED=a |
| 4 | Key = 3, 4 | LED=034 | LED=034 |
| 5 | Key = # | LED = 046 | LED = 046 |
| 6 | Key = B | LED =b | LED =b |
| 7 | Key = 3, 4 | LED=034 | LED=034 |
| 8 | Key = C | LED= c | LED= c |
| 9 | Key = 5 | LED=005 | LED=005 |
| 10 | Key = D | LED=d | LED=d |
| 11 | Key = 4 | LED=004 | LED=004 |
| 12 | Key = # | LED=015 | LED=015 |
| 13 | Key = D | LED=d | LED=d |
| 14 | Key = 0 | LED=000 | LED=000 |
| 15 | Key = # | LED=000 | LED=000 |
| 16 | Key = \* | LED=000 | LED=000 |
| 17 | Key = B | LED=b | LED=b |
| 18 | Key = 8 | LED=008 | LED=008 |
| 19 | Key = # | LED=008 | LED=008 |

# Selain data tersebut, dari pemograman pada kode 1.7 dapat diperoleh flowchart sebagai berikut.





Gambar 4.3 Flowchart pada kode 1.7 1

# Kesimpulan

1. Kode program terlampir pada file Modul1\_kode 1.1
2. Kode program terlampir pada file Modul1\_kode 1.2
3. Kode program terlampir pada file Modul1\_kode 1.3
4. Kode program terlampir pada file Modul1\_kode 1.4
5. Kode program terlampir pada file Modul1\_kode 1.5
6. Kode program terlampir pada file Modul1\_kode 1.6
7. Kode program terlampir pada file Modul1\_kode 1.7

# Daftar Pustaka

Geeksforgeeks. *Encoder and Decoder In Digital Logic.*[Online]

Available at: <https://www.geeksforgeeks.org/encoders-and-decoders-in-digital-logic/>

[Accessed 3 Februari 2020]

Penyusun Modul Praktikum Laboratorium Teknik Fisika II, 2020. *Modul 1 Digital Input/Output*.

Bandung

Referensi Data Kelompok:

1. Kelompok 17 praktikum shift 2
2. Kelompok 19 praktikum shift 2
3. Kelompok 20 praktikum shift 2

Referensi Teman:

1. Tsabita Fidinillah 13318042

# Lampiran

Daftar File dan Folder Terlampir:

1. Modul1\_kode 1.1
2. Modul1\_kode 1.2
3. Modul1\_kode 1.3
4. Modul1\_kode 1.4
5. Modul1\_kode 1.5
6. Modul1\_kode 1.6
7. Modul1\_kode 1.7
8. Jurnal modul 1

1. Encoder and Decoder In Digital Logic, <https://www.geeksforgeeks.org/encoders-and-decoders-in-digital-logic/> [↑](#footnote-ref-1)