ZAIYAN AHYADI

BELAJAR ANTARMUKA ARDUINO

Secara Cepat dari Contoh

Diterbitkan Atas Kerjasama
Deepublish dengan Politeknik Banjarmasin

BELAJAR ANTARMUKA ARDUINO SECARA CEPAT DARI CONTOH

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

Zaiyan Ahyadi

BELAJAR ANTARMUKA ARDUINO SECARA CEPAT DARI CONTOH

POLIBAN PRESS

BELAJAR ANTARMUKA ARDUINO SECARA CEPAT DARI CONTOH

Penulis :
Zaiyan Ahyadi

ISBN :
978-602-53458-0-7

ISBN Elektronis :
978-602-53809-4-5

Desain Sampul dan Tata letak :
Rahma Indera

Penerbit :
POLIBAN PRESS
Cetakan Pertama, Desember 2018

Redaksi :
Politeknik Negeri Banjarmasin, Jl. Brigjen H. Hasan Basry,
Pangeran, Komp. Kampus ULM, Banjarmasin Utara
Telp : (0511)3305052
Email : press@poliban.ac.id

Dicetak oleh :
PERCETAKAN DEEPUBLISH
Jl.Rajawali, G. Elang 6, No 3, Drono, Sardonoharjo, Ngaglik, Sleman
Jl.Kaliurang Km.9,3 – Yogyakarta 55581
Telp/Faks: (0274) 4533427
Website: www.deepublish.co.id
www.penerbitdeepublish.com
E-mail: cs@deepublish.co.id

Katalog Dalam Terbitan (KDT)
Zaiyan Ahyadi —Cet. **1**. — **Belajar Antarmuka Arduino Secara Cepat dari Contoh** :
Poliban Press, 2018.

vii; 96 hlm.; 15,5 x 23 cm

KATA PENGANTAR

Atas rahmat Allah SWT, dan rasa cinta Nabi Muhammad SAW serta dukungan seluruh pihak, akhirnya buku ajar "Belajar Antarmuka Arduino" ini dapat diselesaikan. Terima kasih yang sebesar-besarnya secara khusus kami haturkan kepada Ketua P3M Poliban, yang telah memberi kesempatan, dukungan, dan pendanaan atas terbitnya buku ini.

Buku ajar ini secara khusus diperuntukkan bagi mahasiswa Elektronika Poliban semester 4 yang mengambil mata kuliah antarmuka mikroprosesor. Namun buku ini dapat dipergunakan luas bagi mahasiswa Jurusan Elektro secara umum. Untuk kalangan umum yang ingin belajar tentang Arduino dan antarmukanya melalui buku ini, mewajibkan telah mempunyai pengetahuan dasar tentang pemrograman, terutama Bahasa C.

Buku hanya membahas sebagian kecil dari antarmuka Arduino dengan komponen elektronik. Pembahasan menitik beratkan konsep dasar tentang bagaimana Arduino dapat mengendalikan, membaca status logika dan berkomunikasi dengan komponen dasar elektronika. Dari metode dasar ini, diharapkan mahasiswa dapat mengembangkan untuk komponen dan peralatan elektronika lainnya.

Penulis sadar bahwa buku ini jauh dari sempurna. Untuk itu segala kritik dan saran dari para pembaca sangat diharapkan, sehingga ada perbaikan dan pengembangan atas buku ini di kemudian hari.

Banjarmasin, November 2018

Penulis

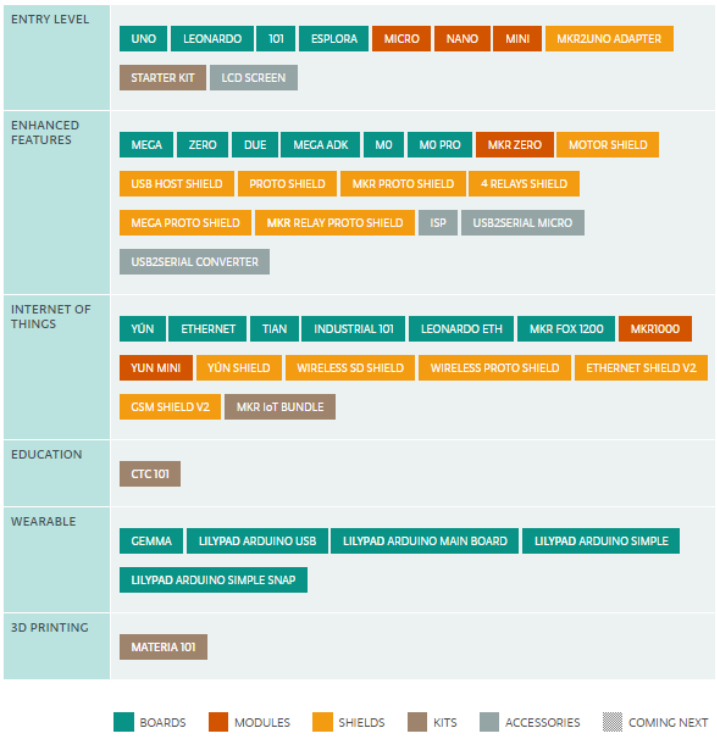
DAFTAR ISI

1. PENGENALAN ARDUINO

Arduino adalah suatu open-source platform elektronik yang berbasis kemudahan penggunaan (*easy to use*) baik hardware maupun software. Dengan kata lain, Arduino merupakan sebuah sistem dasar yang terdiri dari hardware dan software yang mengutamakan kemudahan penggunaannya. Core dari Arduino adalah mikrokontroler dari bermacam-macam tipe.

1.1. Jenis-Jenis Arduino

Ada banyak nama jenis Arduino, di antaranya adalah sebagai berikut:



Gambar 1.1 Jenis-jenis Arduino

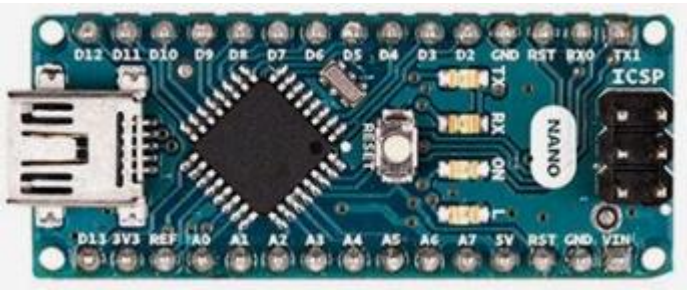
Pada gambar 1.1 dapat dilihat berbagai nama jenis arduino berdasarkan level para penggunanya dan tingkat kerumitan sistemnya.

Pada buku ini hanya akan menggunakan Arduino jenis Uno, Nano dan ProMini. Ketiga jenis Arduino ini menggunakan mikrokontroler AVR yang sama yaitu AT Mega 328. Perbedaannya terletak pada ada tidaknya rangkaian penyearah dan chip CH-340 sebagai pengubah serial ke USB.

Jenis Arduino	Penyearah	CH-340
Uno	Ada	Ada
Nano	Tidak Ada	Ada
Pro Mini	Tidak Ada	Tidak ada

Berdasarkan tabel di atas dapat disimpulkan dari ketiga jenis arduino tersebut maka Arduino Uno mempunyai spesifikasi tertinggi sehingga harganya pun yang termahal.

Gambar 1.2.a Arduino Uno



Gambar 1.2.b Arduino Nano

Gambar 1.2.c Arduino Pro Mini

1.2. Keunggulan dan Kelemahan Arduino

Arduino mempunyai keunggulan dibandingkan dengan sistem mikrokontroler lainnya, berikut ini:

- Banyak library yang sudah tersedia sehingga pemrogram dengan level awal pun dapat menggunakannya dengan mudah. Library tersedia banyak di internet, open source dan free. Disertai dengan contoh-contoh menggunakannya.
- Harganya sangat murah. Karena pengawatan Arduino berlisensi *free* maka banyak perusahaan yang mengopi pengawatannya dan memproduksinya sehingga di pasaran harganya bersaing.

- Mudah digunakan. Disertai dengan program IDE Arduino yang dapat diunduh di internet dengan gratis. Banyak IDE untuk mikrokontroler AVR, namun untuk versi full harus bayar.
- Langsung bisa dapat diprogram hanya dengan menggunakan kabel USB biasa.
- Tidak memerlukan power yang besar, bahkan hanya dengan menggunakan power dari port USB komputer sistem arduino sudah dapat diprogram dan run.

Meskipun begitu banyaknya keunggulan Arduino, namun dari pengalaman, Penulis mendapati ada beberapa kekurangan, yaitu:

- Program yang dihasilkan lebih lambat dan lebih besar
- Tidak mengetahui lebih dalam mengenai arsitektur mikrokontroler AVR.

1.3. Peta kaki Arduino dan Kesetaraannya dengan Kaki AVR

Sistem arduino memberikan nama alias untuk penamaan kaki AVR. Nama alias ini hanya dikenali pada IDE arduino, sehingga jika diterapkan pada editor program AVR yang lain maka akan menghasilkan sintaks error. Gambar berikut memperlihatkan peta dan penamaan pin arduino untuk nano yang juga sama dengan penamaan Uno.

PortB	PB.0	PB.1	PB.2	PB.3	PB.4	PB.5
ArduinoPin	8	9	10	11	12	13

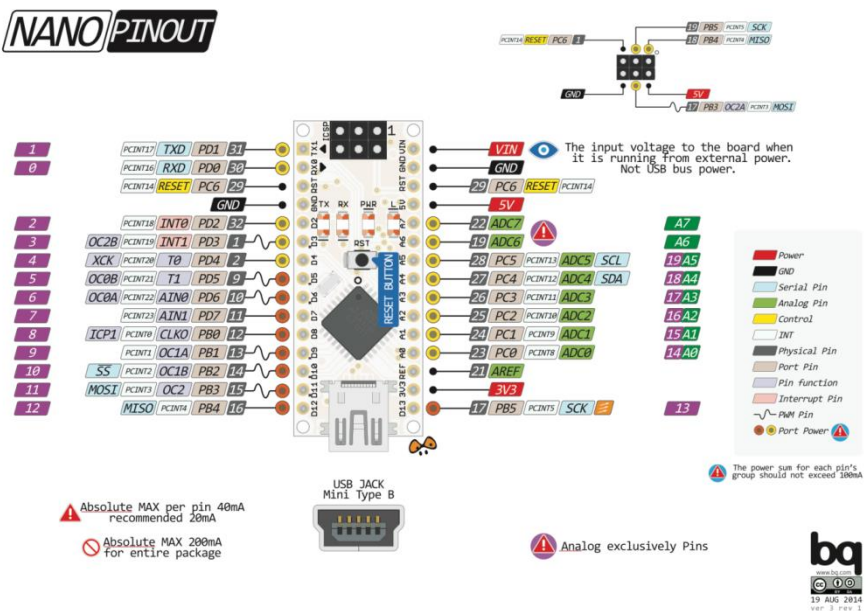
PortC	PC.0	PC.1	PC.2	PC.3	PC.4	PC.5
ArduinoPin	14	15	16	17	18	19

PortD	PD.0	PD.1	PD.2	PD.3	PD.4	PD.5	PD.6	PD.7
ArduinoPin	0	1	2	3	4	5	6	7

Untuk PORTC.4 dan PORTC.5 jika dijadikan sebagai input maka nilai logikanya terkunci pada kondisi LOW. Sedangkan untuk

PORTB.5 terkunci pada mode OUTPUT, karena pin 13 untuk arduino terhubung dengan indikator led.

Port input output pada Arduino dapat dialamatkan sebagai bit maupun sebagai byte. Untuk pengalamatan secara bit dapat menggunakan perintah Arduino dengan perintah digitalWrite, digitalRead dan pinMode. Sedangkan untuk pengalamatan byte dapat menggunakan instruksi assignment Bahasa C. Dengan mengirim output secara byte pada register PORTx dan DDRx, atau membaca register PINx. Contoh DDRD=0xff, PORTD=0xB4, inData=PIND.



Gambar 2.2 a Peta pin Arduino Nano

Soal:

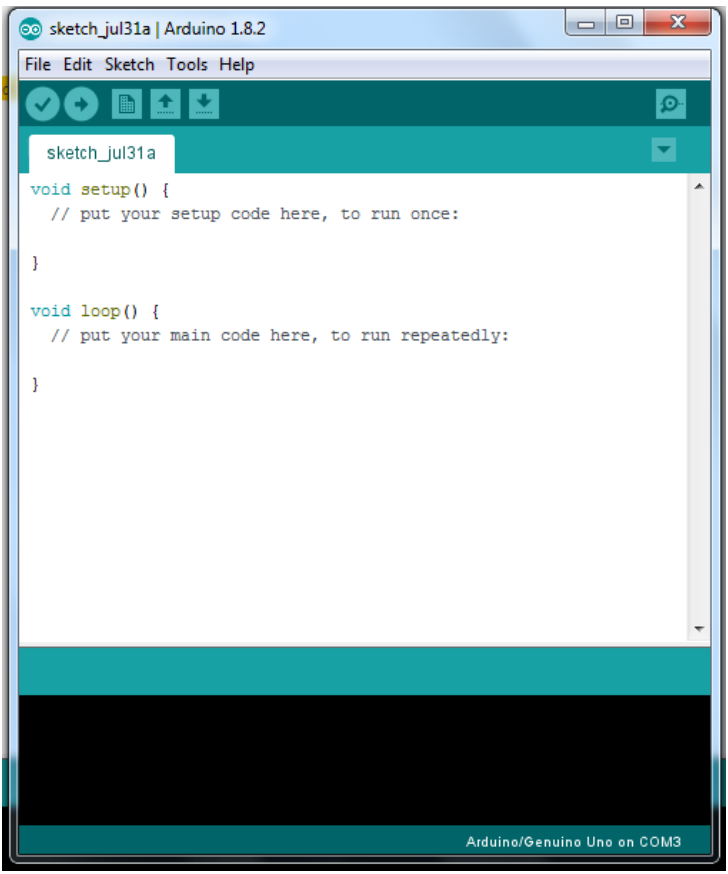
1. Ada berapa banyak pin input output (I/O) seluruhnya pada arduino?
2. Ada berapa banyak pin input output (I/O) seluruhnya chip ATMega 328P?
3. Jelaskan perbedaan kabel untuk memprogramkan Arduino Uno, Arduino Nano, dan Arduino Pro Mini!

2. DASAR INPUT OUTPUT

2.1. Pengenalan IDE Arduino

Ide Arduino dibuat dengan menggunakan Program Java, namun sintaks pemrograman untuk arduino adalah C++. IDE arduino dapat mengenali perintah arduino dan perintah C++.

Tampilan IDE Arduino dapat dilihat pada Gambar 2.1 berikut ini



Gambar 2.1 Tampilan IDE Arduino

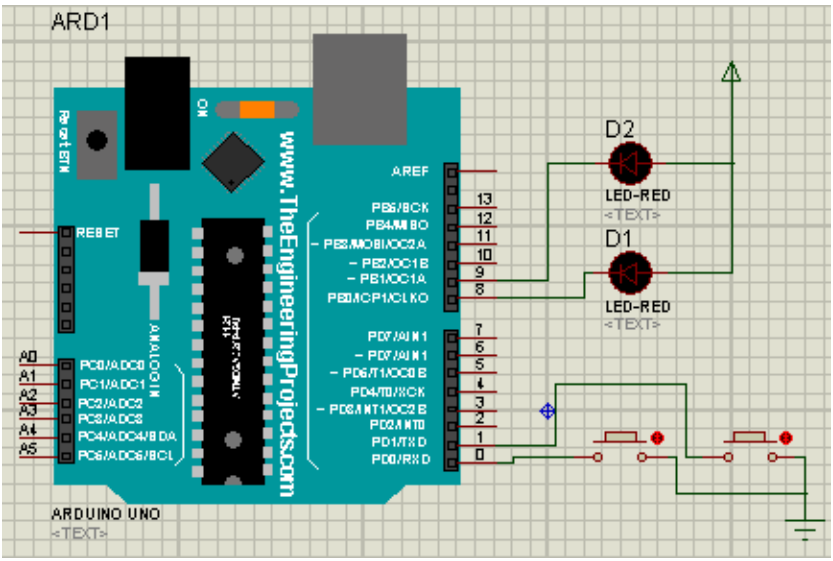
Ketika membuka file baru dengan memilih File → New, IDE Arduino akan memperlihatkan dua buah fungsi void setup() dan void loop() .

Fungsi setup akan dijalankan satu kali saja pada awal Arduino dijalankan, sedangkan fungsi loop akan dijalankan berulang-ulang terus sampai arduino dimatikan.

2.2. Input On dan Input Off untuk Satu Output

Sebelum sampai pada membuat program kita harus mengetahui dulu rangkaian hardware dari sistem elektronik yang mau dikontrol oleh Arduino.

Pada bagian ini kita akan mempelajari dasar perintah input output pada arduino. Untuk itu buatlah rangkaian berikut ini pada proteus.



Gambar 2.2 Rangkaian dasar input output

Pada Gambar 2.2 terlihat ada dua buat saklar atau button yang digunakan sebagai peralatan input digital. Salah satu kaki dari

kedua button1 dan button2 terhubung ke pin 0 dan pin 0 dan pin 1 Arduino, sedangkan kaki lainnya terhubung ke ground. Sehingga ketika saklar tidak ditekan maka pin arduino tidak mendapatkan input, baik itu logika 0 ataupun logika 1. Sedangkan bila button ditekan maka pin Arduino akan mendapatkan input logika 0.

Led D1 dan D2 digunakan sebagai peralatan output. Salah satu kaki dari kedua led D1 dan D2 terhubung ke pin8 dan pin 9 Arduino, sedangkan kaki lainnya terhubung dengan ke Vcc 5 volt. Dengan sambung seperti ini jika pin Arduino di set logik 0 maka led akan menyala sedangkan jika di set berlogika 1 maka led tidak akan menyala.

Ketiklah program berikut ini pada IDE Arduino

Program 2.1

```
#define I1p 0
#define I2p 1
#define Q1p 8
#define Q2p 9
boolean I1,I2,Q1,Q2;

void setup() {
  pinMode(I1p, INPUT_PULLUP);
  pinMode(I2p, INPUT_PULLUP);
  pinMode(Q1p, OUTPUT);
  pinMode(Q2p, OUTPUT);
}

void loop() {
  I1=not digitalRead(I1p);
  I2=not digitalRead(I2p);

  if(I1) Q1=1;
  if(I2) Q1=0;

  digitalWrite(Q1p,not Q1);
```

```
  digitalWrite(Q2p,not Q2);
}
```

Pada Program 2.1 di atas digunakan untuk rangkaian pada gambar 2.2. patut diingat sebaiknya program harus mengikuti hardware, bukan hardware yang mengikuti program. Hal ini karena program dengan mudah dapat diubah, sedangkan untuk mengubah hardware agar sesuai program maka harus melepas dan menyolder ulang komponen.

Pada awal baris terdapat instruksi

```
#define I1 0
```

Instruksi ini menyatakan bahwa semua karakter kata I1 pada program akan diganti menjadi angka 0 pada proses kompilasi. Hal ini berguna untuk memudahkan programmer untuk mengingat bahwa input I1 berada pada pin 0 arduino. Bisa dilihat juga untuk baris #define yang lain. Perintah yang dengan diawali tanda pagar adalah instruksi preprosessor, artinya perintah tersebut akan diproses sebelum masuk ke prosessor yaitu diolah dulu pada saat kompilasi dengan mengubah angka dengan nama alias yaitu kata setelah define.

Pada fungsi setup terdapat instruksi

```
pinMode(I1, INPUT_PULLUP);
```

Instruksi ini bermaksud untuk mengatur I1 atau pin 0 arduino berfungsi sebagai input digital dengan pull up resistor. Jika diinginkan input tanpa pull up resistor maka cukup dengan perintah

```
pinMode(I1, INPUT);
```

Sedangkan untuk mengatur pin arduino sebagai output maka digunakan instruksi seperti contoh berikut

```
pinMode(Q1, OUTPUT);
```

Instruksi di atas menyatakan bahwa Q1 diatur sebagai output.

Pada fungsi loop terdapat instruksi

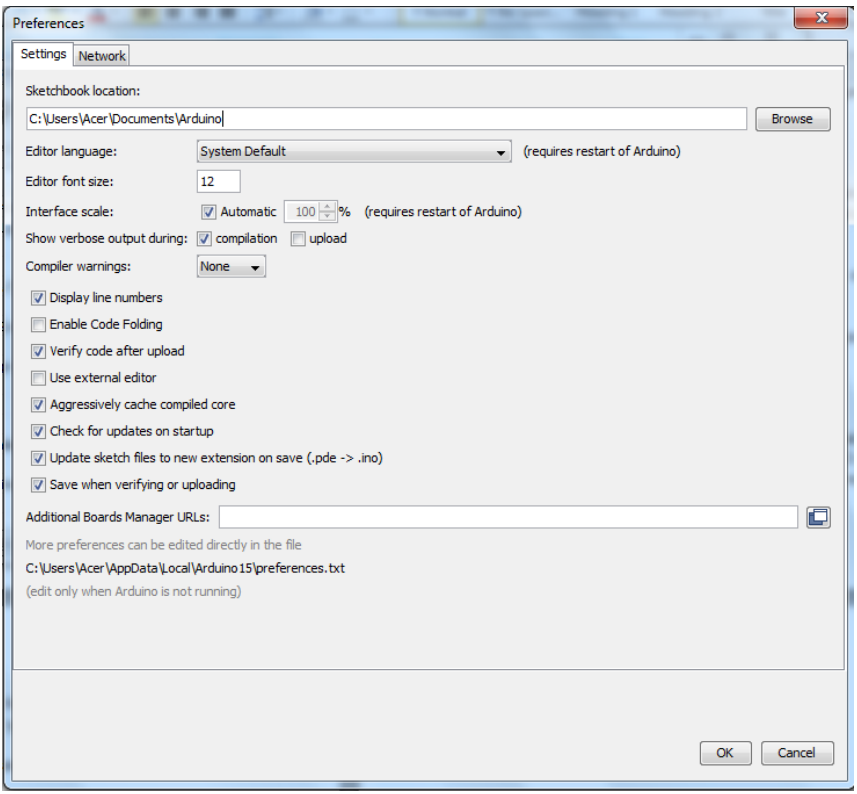
```
if(not I1) Q1=not HIGH;
```

```
if(not I2) Q2=not LOW;
```

Instruksi ini berarti jika I1 bernilai 0 (b1 ditekan) maka Q1 akan bernilai 0 (not high) yang berarti led nyala, dan jika I2 bernilai 0 (b2 ditekan) maka Q1 akan bernilai 1 (not LOW) yang berarti led akan mati.

Sebelum program kompilasi dan hasilnya dijalankan pada simulasi proteus, maka ada beberapa hal yang harus disunting pada IDE Arduino.

Klik File → preferences maka akan tampil seperti gambar 2.3 berikut ini



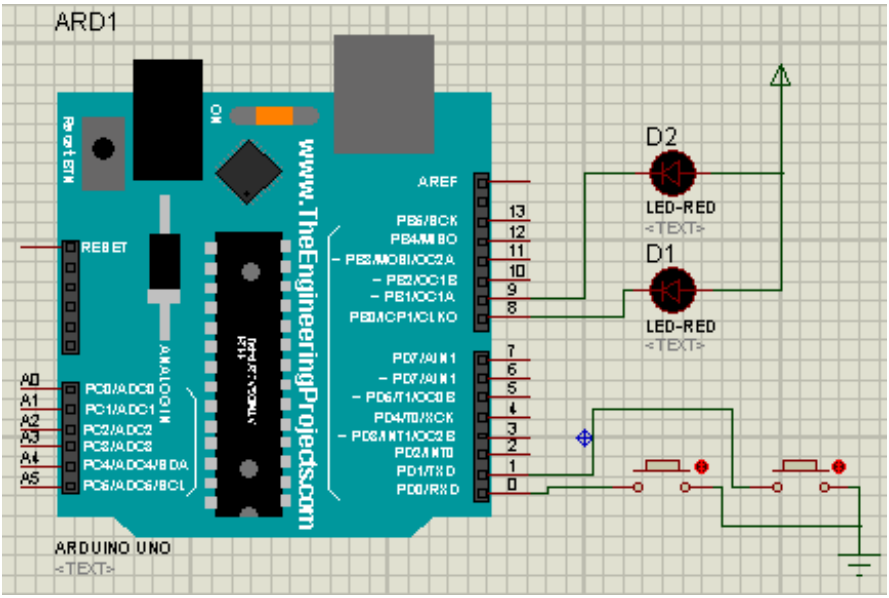
Gambar 2.3 Setting preference IDE arduino

Pastikan show verbose output during: pada bagian compilation tercentang. Ini berguna agar pada akhir kompilasi kita dapat mengetahui dimana file .hex.ino tersimpan. Untuk menyimulasikan pada proteus file inilah yang akan digunakan.

Selain itu untuk kemudahan pembacaan dan deteksi error, alangkah baiknya editor arduino dapat menampilkan nomor baris. Untuk itu pastikan Display line numbers tercentang.

Sebelum sampai pada membuat program kita harus mengetahui dulu rangkaian hardware dari sistem elektronik yang mau dikontrol oleh Arduino.

Pada bagian ini kita akan mempelajari dasar perintah input output pada arduino. Untuk itu buatlah rangkaian berikut ini pada proteus.



Gambar 2.2 Rangkaian dasar input output

Pada Gambar 2.2 terlihat ada dua buat saklar atau button yang digunakan sebagai peralatan input digital. Salah satu kaki dari kedua button1 dan button2 terhubung ke pin 0 dan pin 0 dan pin 1 Arduino, sedangkan kaki lainnya terhubung ke ground. Sehingga ketika saklar tidak ditekan maka pin arduino tidak mendapatkan input, baik itu logika 0 ataupun logika 1. Sedangkan bila button ditekan maka pin Arduino akan mendapatkan input logika 0.

Led D1 dan D2 digunakan sebagai peralatan output. Salah satu kaki dari kedua led D1 dan D2 terhubung ke pin8 dan pin 9 Arduino, sedangkan kaki lainnya terhubung dengan ke Vcc 5 volt. Dengan sambung seperti ini jika pin Arduino di set logik 0 maka led akan menyala sedangkan jika di set berlogika 1 maka led tidak akan menyala.

Ketiklah program berikut ini pada IDE Arduino
Program 2.1

```
#define I1p 0
#define I2p 1
#define Q1p 8
#define Q2p 9
boolean I1,I2,Q1,Q2;

void setup() {
  pinMode(I1p, INPUT_PULLUP);
  pinMode(I2p, INPUT_PULLUP);
  pinMode(Q1p, OUTPUT);
  pinMode(Q2p, OUTPUT);
}

void loop() {
  I1=not digitalRead(I1p);
  I2=not digitalRead(I2p);

  if(I1) Q1=1;
```

```
  if(I2) Q1=0;

  digitalWrite(Q1p,not Q1);
  digitalWrite(Q2p,not Q2);
}
```

Pada Program 2.1 di atas digunakan untuk rangkaian pada gambar 2.2. patut diingat sebaiknya program harus mengikuti hardware, bukan hardware yang mengikuti program. Hal ini karena program dengan mudah dapat diubah, sedangkan untuk mengubah hardware agar sesuai program maka harus melepas dan menyolder ulang komponen.

Pada awal baris terdapat instruksi

```
#define I1 0
```

Instruksi ini menyatakan bahwa semua karakter kata I1 pada program akan diganti menjadi angka 0 pada proses kompilasi. Hal ini berguna untuk memudahkan programmer untuk mengingat bahwa input I1 berada pada pin 0 arduino. Bisa dilihat juga untuk baris #define yang lain. Perintah yang dengan diawali tanda pagar adalah instruksi preprosessor, Artinya perintah tersebut akan diproses sebelum masuk ke prosessor yaitu diolah dulu pada saat kompilasi dengan mengubah angka dengan nama alias yaitu kata setelah define.

Pada fungsi setup terdapat instruksi

```
pinMode(I1, INPUT_PULLUP);
```

Instruksi ini bermaksud untuk mengatur I1 atau pin 0 arduino berfungsi sebagai input digital dengan pull up resistor. Jika diinginkan input tanpa pull up resistor maka cukup dengan perintah

```
pinMode(I1, INPUT);
```

Sedangkan untuk mengatur pin arduino sebagai output maka digunakan instruksi seperti contoh berikut

```
pinMode(Q1, OUTPUT);
```

Instruksi di atas menyatakan bahwa Q1 diatur sebagai output.

Pada fungsi loop terdapat instruksi

```
if(not I1) Q1=not HIGH;
if(not I2) Q2=not LOW;
```

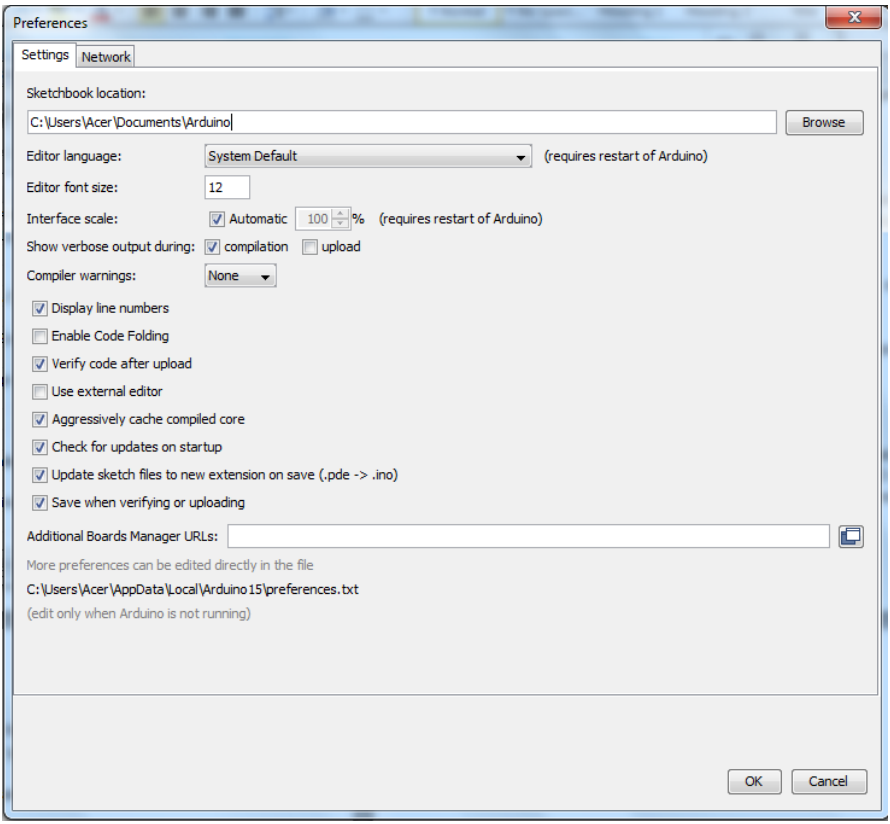
Instruksi ini berarti jika I1 bernilai 0 (b1 ditekan) maka Q1 akan bernilai 0 (not high) yang berarti led nyala, dan jika I2 bernilai 0 (b2 ditekan) maka Q1 akan bernilai 1 (not LOW) yang berarti led akan mati.

Sebelum program kompilasi dan hasilnya dijalankan pada simulasi proteus, maka ada beberapa hal yang harus disetting pada IDE Arduino.

Klik File → preferences maka akan tampil seperti gambar 2.3

Pastikan show verbose output during: pada bagian compilation tercentang. Ini berguna agar pada akhir kompilasi kita dapat mengetahui dimana file .hex.ino tersimpan. Untuk menyimulasikan pada proteus file inilah yang akan digunakan.

Selain itu untuk alangkah baiknya editor arduino dapat menampilkan nomor baris. Untuk itu pastikan Display line numbers tercentang.



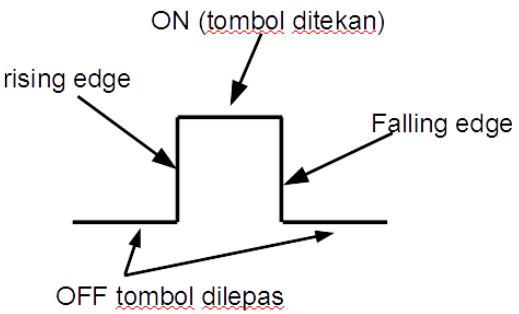
Gambar 2.3 Setting preference IDE arduino

2.3. Input On dan Off dengan Satu Tombol

Pada bagian 2.2 kita telah membuat program arduino dengan dua buah input, dimana 1 input untuk meng-On kan dan input lainnya untuk meng-Off kan output. Bagaimana kalau kita menginginkan hanya satu tombol saja dapat melakukan input dan output secara bergantian (toggle). Fungsi tombol seperti ini banyak terdapat pada peralatan elektronik sekarang ini, seperti pada televisi, mobile phone, komputer, printer dan lain sebagainya. Dimana On dan Off hanya menggunakan satu tombol. Namun untuk dapat mengerti

bagaimana hal tersebut dapat dibuat maka perlu dipahami dulu mengenai rising edge dan falling edge.

Suatu tombol jika ditekan (posisi ON) kemudian dilepas (posisi OFF) dapat digambarkan level tegangan pada salah kaki tombol sebagai berikut



Gambar 2.4 timing diagram level logika penekanan tombol digital

Pada program terdahulu deteksi input pada prinsipnya adalah mendeteksi level tegangan input apakah mempunyai level logika 1 atau logika 0. Selain deteksi level, dari gambar 2.4 terlihat ada yang dinamakan rising edge yaitu level tegangan transisi dari off ke on dan falling edeg yaitu level tegangan dari off ke on. Istilah lain dari raising edge adalah *differentiate up* (DIFU) yang biasa digunakan pada istilah PLC atau *positive going transitition* (PGT) yang biasa digunakan pada sistem digital. Sedangkan falling edge mempunyai istilah lian differentiate Down (DIFD) dan juga NGT negative going transition. Deteksi transisi level dapat dilakukan dengan menggunakan satu buah variabel bantu, untuk mudahnya diberi nama saja variabel Old.

Lihat program 2.2. berikut ini

Program2.2

```
#define I1p 0
#define I2p 1
#define Q1p 8
#define Q2p 9
bool I1,I2,Q1,Q2;
bool I1Old, I2Old;
void setup() {
  pinMode(I1p, INPUT_PULLUP);
  pinMode(I2p, INPUT_PULLUP);
  pinMode(Q1p, OUTPUT);
  pinMode(Q2p, OUTPUT);
}

void loop() {
  I1Old=I1;
  I2Old=I2;
  I1=not digitalRead(I1p);
  I2=not digitalRead(I2p);
  if(not I1Old and I1)
  {
    if(Q1) Q1=0; else Q1=1;
  }
  if(I2Old and not I2)
  {
    if(Q2) Q2=0; else Q2=1;
  }

  digitalWrite(Q1p,not Q1);
  digitalWrite(Q2p,not Q2);
}
```

Simulasikan pada rangkaian gambar 2.2, maka akan terlihat jika I1 ditekan dan ditahan maka led D1 akan menyala, ketika dilepas D1 tetap menyala. Ketika ditekan sekali lagi maka D1 akan padam dan terus padam. Demikian seterusnya nyala dan padam untuk D1 terjadi bergantian pada saat tombol ditekan.

Sedangkan input I2 jika ditekan dan hol maka Q2 tidak berubah, kecuali ketika tombol dilepas maka kondisi Q2 akan toggle dari padam menjadi nyala dan sebaliknya.

Pada deteksi transisi selain variabel bantu (Old) yang berfungsi menyimpan kondisi input satu siklus ke belakang, maka perlu juga kondisi dari output dijadikan input. Coba jawab, kapan output akan nyala jika tombol ditekan? Jawabannya jika kondisi sekarang output padam. Demikian juga sebaliknya.

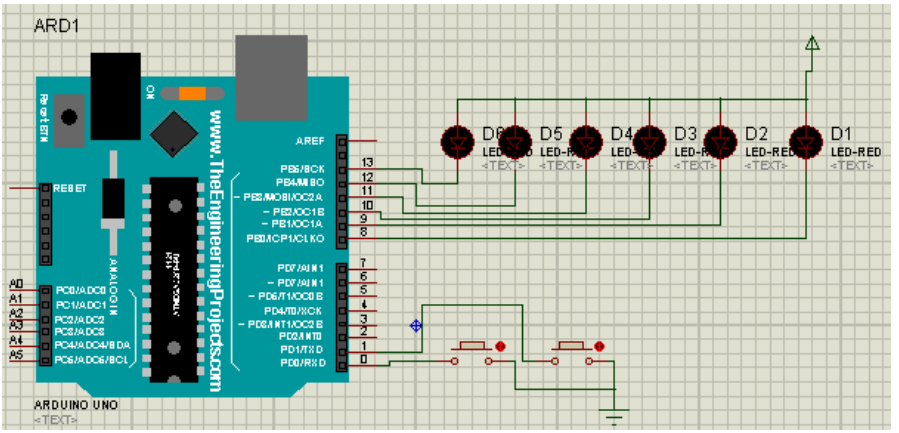
Coba perhatikan baris dari program 2.2 berikut

```
if(not I1Old and I1)
{
  if(Q1) Q1=0; else Q1=1;
}

Baris di atas dapat diganti menjadi
if(not I1Old and I1)
{
  Q1 = not Q1;
}
```

Potongan program terakhir menyatakan bahwa jika terjadi transisi pada input I1, maka output menjadi toggle, jika sebelumnya Q1 on maka akan menjadi off demikian juga sebaliknya.

2.4. Running Led dengan Menggunakan Delay

Perhatikan gambar rangkaian berikut



Gambar 2.4 running led

Perhatikan Program untuk gambar 2.4

Program 2.3

```
#define I1p 0
#define I2p 1

bool I1,I2;
bool I1Old, I2Old;
byte leds=1;
void setup() {
  pinMode(I1p, INPUT_PULLUP);
  pinMode(I2p, INPUT_PULLUP);
  DDRB=0xFF;
  leds=1;
}

void loop() {
  I1Old=I1;
```

```
  I2Old=I2;
  I1=not digitalRead(I1p);
  I2=not digitalRead(I2p);
  if(not I1Old and I1)
  {
    leds = leds << 1;
  }

  PORTB = ~leds;
  delay (1000);
}
```

Pada program 2.3 terdapat instruksi

`DDRB=0xFF;`

Instruksi tersebut menjadikan PORTB sebagai output. Instruksi itu adalah instruksi byte, dimana perubahan terjadi secara 8 bit. Ini menggantikan perintah pinMode(Q1, OUTPUT). Perintah pinMode yang digunakan untuk setting pin secara bit merupakan perintah arduino sedangkan DDRB=0xff adalah perintah C.

Instruksi

`leds = leds << 1;`

Adalah instruksi untuk menggeser nilai biner leds satu bit ke kiri.

Instruksi

`PORTB = ~leds;`

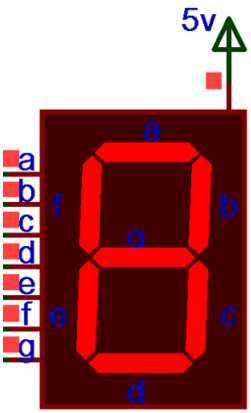
Adalah instruksi untuk mengirimkan nilai leds yang di-complement ke PORTB.

Soal:

1. Jelaskan berapa kali instruksi pada fungsi setup() dan fungsi loop() dijalankan!
2. Apakah penamaan variabel pada program Arduino “case senstitve” (membedakan antara huruf kecil dan capital)?
3. Jelaskan perbedaan fungsi PORT, PIN, dan DDR!

3. ANTARMUKA DENGAN 7-SEGMEN

Suatu 7-segmen terdiri dari 7 buah led yang tersusun sedemikian rupa sehingga mampu membentuk pola angka dari 0 sampai dengan 9. Seven-segmen ada dua jenis yaitu common anoda dan common katoda. Pada common anoda, pin common harus diberikan logika 1 (5 volt), sedangkan pin data segmen harus diberikan logika 0 agar segmen tersebut akan menyala. Sedangkan untuk common katoda proses terjadi sebaliknya.



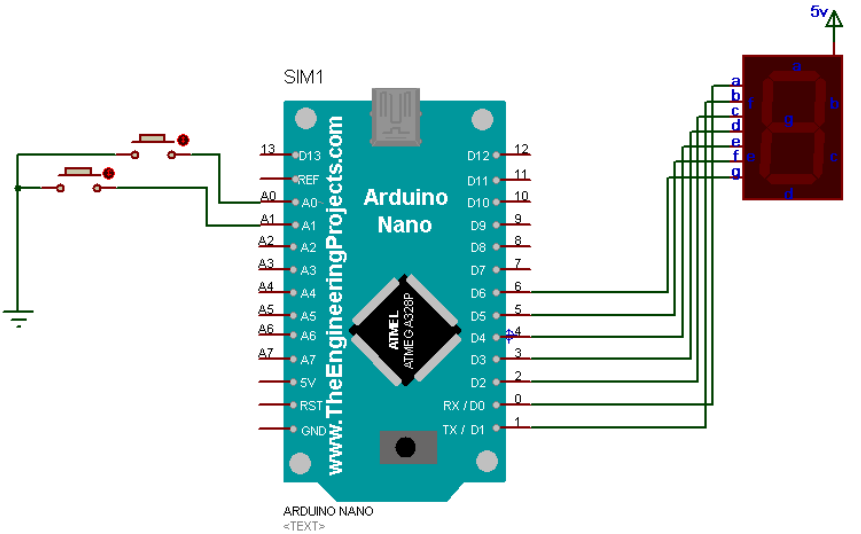
Gambar 3.1
seven-segmen common anoda

Pola angka 7 segmen

hgfedcba
0 = 0b11000000 = 0xC0
1 = 0b11111001 = 0xF9
2 = 0b10100100 = 0xA4
3 = 0b10110000 = 0xB0
4 = 0b10011001 = 0x99
5 = 0b10010010 = 0x92
6 = 0b10000010 = 0x82
7 = 0b11111000 = 0xF8
8 = 0b10000000 = 0x80
9 = 0b10010000 = 0x90

Untuk menampilkan pola maka segmen-segmen harus di-set dengan pola yang ditunjukkan pada gambar 3.1

3.1. Penampil Angka dengan 7-Segmen 1 Digit



Gambar 3.2 Rangkaian seven-segmen terhubung ke arduino nano

Buatlah rangkaian seperti gambar 1 di atas dengan menggunakan Protheus.

Tuliskan Program 1 berikut dengan IDE arduino

Program 3.1a

```
byte angka[]=

{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
};
byte i=0, number;
void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
}

void loop() {
```

```
  //proses
  number = i;
  delay(100);
  i++

  //display
  PORTB = 0x0f;
  PORTD = angka[number%10];

  i++;
}
```

Jika semuanya benar, maka pada simulasi protheus untuk gambar 3.2 akan memperlihatkan angka mulai dari 0 sampai dengan 9 berulang-ulang.

Kalau diamati lebih teliti pada perulangan ke-25 ketika display memperlihatkan angka 5 setelah itu langsung ke-0. Coba pikirkan kenapa hal tersebut bisa terjadi.

Program 3.1b

```
#define tblNaikPin  14
#define tblTurunPin 15
byte
angka[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x8
0,0x90};
byte i=0,number;
bool tblNaik=0, tblTurun=0, tblNaikOld,
tblTurunOld;

void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
  pinMode(tblNaikPin, INPUT_PULLUP);
  pinMode(tblTurunPin, INPUT_PULLUP);
```

```
}

void loop() {
  //baca input
  tblNaikOld = tblNaik;
  tblNaik = not digitalRead(tblNaikPin);
  tblTurunOld = tblTurun;
  tblTurun = not digitalRead(tblTurunPin);

  //proses
  number=i;
  if(tblNaik and not tblNaikOld) i++;
  if(tblTurun and not tblTurunOld) i--;

  //display (output)
  PORTB = 0x0f;
  PORTD = angka[number%10];
}
```

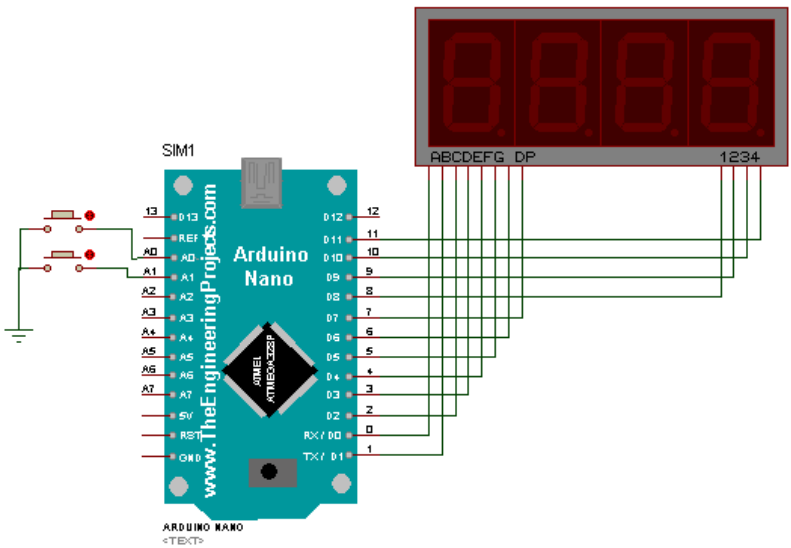
3.2. Display 7-Segmen 4 Digit Multiplex

Untuk menampilkan angka dua digit atau lebih, beberapa seven-segmen diperlukan. Pada bagian sebelumnya kita mengetahui untuk menampilkan satu digit angka memerlukan 7 bit data. Bagaimana untuk menampilkan 4 digit angka, apakah sistem memerlukan jalur data sebanyak 4 x 7 atau 28 bit? Padahal Arduino Uno, Nano maupun Pro mini cuma mempunyai pin digital sebanyak 21 bit. Bagaimana bisa?

Hal ini dapat diatasi dengan menggunakan system scanning output pada digit 7-segmen. System scanning output berarti mengirim data ke output secara cepat dan bergantian sehingga output seolah-olah terlihat oleh mata aktif secara bersamaan. Pada kasus scanning 7-segmen yang bergantian adalah digit 7-segmennya akan aktif secara bergantian. Untuk common anoda, untuk mengaktifkan segmen berarti memberikan tegangan 5 volt

pada anodanya. Yang patut diperhatikan dalam system scanning adalah timing waktunya (frekuensi) harus disesuaikan dengan komponen output-nya. Karena jika frekuensinya terlalu lambat maka akan terlihat kedip pada 7-segmen. Sedangkan bila terlalu cepat maka tampilan 7-segmen akan redup.

Gambar 3.2 memperlihatkan rangkaian system scanning empat buah 7-segmen yang terhubung ke arduino nano, dimana programnya dapat dilihat pada program 3.2.



Gambar 3.3 System scanning 7-segmen pada system arduino nano

Program 3.2 : Program menampilkan 4 buah seven-segmen

```
byte angka[]=

{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
};
byte ribuan,ratusan=0,puluhan=0,satuan=0;
int bil=2457;
```

```
void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
}

void loop() {
  //subrutin proses
  satuan= bil%10;
  puluhan= (bil/10)%10;
  ratusan= (bil/100)%10;
  ribuan= (bil/1000)%10;
  bil++;

  // subrutin display
  PORTB = 0x08;
  PORTD = 0xFF;
  PORTD = angka[satuan];
  digitalWrite(7,HIGH);
  delay(10);
  PORTB = 0x04;
  PORTD = 0xFF;
  PORTD = angka[puluhan];
  digitalWrite(7,HIGH);
  delay(10);
  PORTB = 0x02;
  PORTD = 0xFF;
  PORTD = angka[ratusan];
  digitalWrite(7,HIGH);
  delay(10);
  PORTB = 0x01;
  PORTD = 0xFF;
  PORTD = angka[ribuan];
  digitalWrite(7,HIGH);
```

```
  delay(10);
}
```

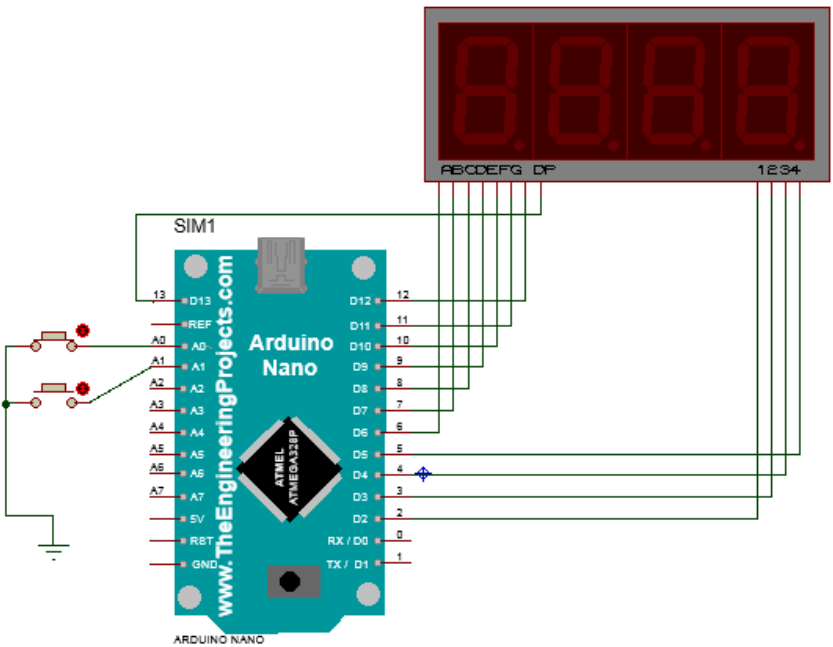
Pada sub rutin display program 3.2 di atas memperlihatkan bahwa seluruh digit akan menyala semua dalam satu siklus loop dengan penundaan sekitar 10 ms. Cara lain adalah dengan menampilkan satu digit saja yang aktif dalam setiap siklus. Cara seperti ini memerlukan variable tambahan berupa counter yang berfungsi untuk memilih digit yang akan aktif secara bergantian untuk setiap siklus. Ini diperlihatkan pada program 3.3 berikut ini.

Program 3.3 : Menampilkan 4 digit dengan satu digit per-siklus

```
byte angka[]=

{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
};
byte ribuan,ratusan=0,puluhan=0,satuan=0;
int bil=2457,tunda;
byte sel;
void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
}

void loop() {
  //subrutin proses
  satuan= bil%10;
  puluhan= (bil/10)%10;
  ratusan= (bil/100)%10;
  ribuan= (bil/1000)%10;
  tunda++;
  if(!(tunda%16)) bil++;

  // subrutin display
```

```
  PORTB = 1<<sel;

   switch(sel)
    {
      case 0: PORTD=angka[ribuan]; break;
      case 1: PORTD=angka[ratusan];break;
      case 2: PORTD=angka[puluhan];break;
      case 3: PORTD=angka[satuan];break;
    }
    delay(10);

    sel++;
    if(sel==4) sel=0;
}
```

Pada semua contoh system seven-segmen di atas, kita selalu menggunakan semua segmen terhubung pada satu buah port (PORTB) dan semua digit dikendalikan oleh 1 port (PORTB). Bagaimana jika segmen tidak dikendalikan oleh satu PORT saja? Seperti system yang ditampilkan pada gambar berikut ini. Delapan segmen terhubung dengan pin 6 sampai dengan pin 13, yang merupakan gabungan PORTD dan PORTB.



Untuk mengendalikan segmen atau digit yang tidak tergabung dalam satu port, kita harus mengaktifkan segmen atau digit nya dengan perintah bit I/O, yaitu digitalWrite. Perintah dilakukan 8 kali sehingga semua segmen mendapatkan nilai yang benar. Ini dapat dilihat pada contoh program 3.4 berikut ini

Program 3.4

```
byte angka[]=

{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
};
volatile byte sel=0;
volatile int bil;
byte volatile satuan,puluhan,ratusan,ribuan;
bool up,upOld, dn,dnOld;
byte varPORTD,varPORTB=0;
```

```
byte numDigits = 4;
byte digitPins[] = {2, 3, 4, 5,0}; //Digits:
1,2,3,4
byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13};
//segmen a b c d e f g dot

void setup()
{
  DDRD=0xFF;
  DDRB=0xFF;
  pinMode(14,INPUT_PULLUP);
  pinMode(15,INPUT_PULLUP);
  bil=1356;
}

void loop()
{

  upOld=up;
  up=digitalRead(14);
  dnOld=dn;
  dn=digitalRead(15);
  if(upOld and not up) bil++;
  if(dnOld and not dn) bil=0;

  satuan=bil%10;
  puluhan=(bil/10)%10;
  ratusan=(bil/100)%10;
  ribuan=bil/1000;
  tampilkan();
  delay(5);
}

void tampilkan()
```

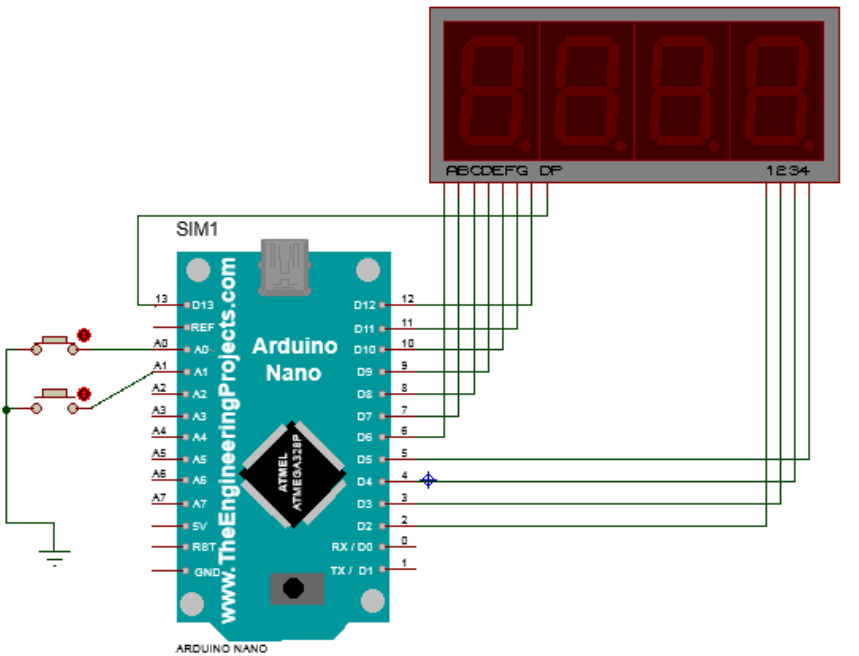
```
{
    varPORTB=1<<sel;
    for(int i=0; i<numDigits; i++)
      digitalWrite(digitPins[i], varPORTB &
(1<<i));

    switch(sel)
    {
      case 0: varPORTD=angka[ribuan]; break;
      case 1: varPORTD=angka[ratusan];break;
      case 2: varPORTD=angka[puluhan];break;
      case 3: varPORTD=angka[satuan];break;
    }
    for(int i=0; i<8;i++)
      digitalWrite(segmentPins[i], varPORTD &
(1<<i));

    sel++;
    if (sel==numDigits)sel=0;
}
```

Catatan untuk program 3.4: Untuk simulasi proteus nilai numDigits di lebih satu dari digit sebenarnya. Jadi misalnya banyaknya digit adalah 4 maka nilai numDigits=5. Dan juga jangan lupa menambahkan nomor pin pada array digitPins[]. Tambahkan dengan nomor pin yang tidak digunakan (pada program contoh adalah pin 0).

Penambahan digit ini hanya berlaku untuk simulasi Porteus agar tampilan yang dihasilkan benar, untuk semua digit muncul angka. Jika tidak ditambahkan maka digit paling samping tidak akan terlihat.

3.3. Jam Digital dengan penampil 7-segmen

Setelah mengetahui bagaimana beberapa 7-segmen bekerja secara system scanning, akan menarik jika kita dapat mengaplikasikannya pada sistem jam digital. Dimana pada sistem jam digital system scanning menjadi kompleks. Dalam system jam digital biasanya mempunyai beberapa mode, mode run akan menampilkan jam dan menit, dan mungkin juga detik yang diwakili dengan kedip titik pemisah jam dan menit. Di samping itu juga ada mode tampilan tanggal dan bulan. Selain itu juga mempunyai mode edit untuk mengubah nilai jam, atau tanggal. Program 3.5 merupakan kode untuk membuat rangkaian pada gambar 3.3 dapat bekerja menjadi sebuah jam digital 4 digit.



Program 3.5

```
byte angka[]=

{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
};
volatile byte sel=0;
volatile int bil;
byte volatile menitSat,menitPul,detikSat,detikPul;
byte detik, menit;
int tunda=0;
bool up,upOld, dn,dnOld;
byte varSegment,varDigit=0;
byte numDigits = 5;
byte digitPins[] = {2, 3, 4, 5, 0};
byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13};
//segmen a b c d e f g dot

void setup()
{
  DDRD=0xFF;
  DDRB=0xFF;
  pinMode(14,INPUT_PULLUP);
  pinMode(15,INPUT_PULLUP);
}

void loop()
{

  upOld=up;
  up=digitalRead(14);
  dnOld=dn;
  dn=digitalRead(15);
  if(upOld and not up) bil++;
```

```
  if(dnOld and not dn) bil=0;

  detikSat=detik%10;
  detikPul=(detik/10)%10;
  menitSat=menit%10;
  menitPul=(menit/10)%10;
  tampilkan();
  delay(1);
  tunda++;
  if(!(tunda%256)) detik++;
  if(detik==60)
  {
    detik=0;
    menit++;
    if(menit==60) menit=0;
  }
}

void tampilkan()
{
    varDigit=1<<sel;
    for(int i=0; i<numDigits; i++)
      digitalWrite(digitPins[i], varDigit &
(1<<i));

    switch(sel)
    {
      case 0: varSegment=angka[menitPul]; break;
      case 1: varSegment=angka[menitSat]; break;
      case 2: varSegment=angka[detikPul]; break;
      case 3: varSegment=angka[detikSat]; break;
    }
    for(int i=0; i<8;i++)
```

```
      digitalWrite(segmentPins[i], varSegment &
(1<<i));

    sel++;
    if (sel==numDigits)sel=0;
}
```

Latihan:

1. Coba hilangkan perintah delay. Lihat apa yang terjadi.
2. Coba ganti rangkaian system jam di atas, sehingga semua segmen terkumpul pada satu (PORTD) dan pin digit terkumpul pada satu port (PORTB.0 ...PORTB.3). Ubahlah program menjadi perintah byte seperti pada contoh 3.3. dengan modifikasi delay dihapus dan if(!(tunda%256)) diganti menjadi if(!(tunda%8192)). Coba bandingkan dengan latihan no 1.

3.4. Membuat Fungsi dan Prosedur

Untuk membuat suatu program mudah dipahami, alangkah lebih baik jika dibuat terstruktur dalam bentuk fungsi dan prosedur. Sehingga program dapat dimodulkan secara bertingkat berdasarkan level detailnya. Beberapa baris program yangmembentuk suatu kerja khusus yang disebut dengan rutin dapat dimodulkan menjadi suatu fungsi. Dengan begitu detail proses yang dilakukan rutin tersebut dapat di abstraksi menjadi satu bari perintah pemanggilan fungsi tersebut. Jika ingin mengetahui bagaimana detail dari perintah tersebut dapat dibaca *body* fungsi tersebut.

Program 3.4 berikut menyederhanakan program 3.3 dimana tampilan perintah untuk mengirimkan output pada system scanning dijadikan suatu fungsi. Sedangkan program 3.5 merupakan program komplet jam digital yang dilengkapi dengan mode edit.

Program 3.5 display 7segmen dibuat menjadi fungsi

```
byte angka[]=

{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
};
byte i=0,detik=0,menit=0,jam=0;
void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
}

void loop() {
  //subrutin proses
  if(i==1)
  {
    i=0;
    dot=not dot;
    detik++;
    if(detik==60)
```

```
    {
      detik=0;
      menit++;
      if(menit==60)
      {
        menit=0;
        jam++;
        if(jam==12) jam=0;
      }
    }
  }
  i++;


//tampilkan(angka[detik%10],angka[(detik/10)%10],a
ngka[menit%10],angka[(menit/10)%10]);

}
  // subrutin display
  void tampilkan(byte digit1 , byte digit2, byte
digit3, byte digit4)
  {
  PORTB = 0x08;
  PORTD = 0xFF;
  PORTD = digit1; //angka[detik%10];
  digitalWrite(7,HIGH);
  delay(10);
  PORTB = 0x04;
  PORTD = 0xFF;
  PORTD = digit2; //angka[(detik/10)%10];
  digitalWrite(7,HIGH);
  delay(10);
  PORTB = 0x02;
  PORTD = 0xFF;
```

```
  PORTD = digit3; //angka[menit%10];
  digitalWrite(7,LOW);
  delay(10);
  PORTB = 0x01;
  PORTD = 0xFF;
  PORTD = digit4; //angka[(menit/10)%10];
  digitalWrite(7,HIGH);
  delay(10);
}
```

Program 3.4 di atas memperlihatkan sistem jam tanpa bias diedit. Program 3.5 berikut memperlihatkan sistem jam dengan fungsi mode edit.

Program 3.5: Program Jam digital dengan mode edit

```
byte angka[]=

{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
,0xff};
byte i=0,detik=0,menit=0,jam=0,
tanggal=1,bulan=1,j=0, digitOn=0, tempjam;
bool tombol1, tombol2, tombol1old, tombol2old,
edjam;
enum _mode {runn,editJam, editMenit,editTgl,
editBulan};
_mode mode;

void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
  pinMode(14, INPUT_PULLUP);
  pinMode(15, INPUT_PULLUP);
}
```

```
void loop() {
  //baca input
  tombol1old=tombol1;
  tombol1 = !digitalRead(14);
  tombol2old=tombol2;
  tombol2 = !digitalRead(15);

  //subrutin proses
    if(i==10)
  {
    i=0;

    detik++;
    if(detik==60)
    {
      detik=0;
      menit++;
      if(menit==60)
      {
        menit=0;
        jam++;
        if(jam==12) jam=0;
      }
    }
  }
  i++;



  switch (mode)
  {
    case runn:         runnFunc(); break;
    case editJam :     editJamFunc(); break;
    case editMenit :   editMenitFunc(); break;
```

```
    case editBulan:   editBulanFunc(); break;
    case editTgl :    editTglFunc(); break;
  }

}

  void runnFunc()
  {
    if (!tombol1old and tombol1) {mode = editJam;
tempjam=jam;}
    if (tombol2)

tampilkan(angka[bulan%10],angka[(bulan/10)%10],ang
ka[tanggal%10],angka[(tanggal/10)%10]);
    else

tampilkan(angka[menit%10],angka[(menit/10)%10],ang
ka[jam%10],angka[(jam/10)%10]);
  }

  void editJamFunc()
  {
    j++;
    if (j==20){ edjam= not edjam; j=0;}
    if (!tombol2old and tombol2) {tempjam++; if
(jam==12) jam=0;}
    if(edjam)

tampilkan(angka[menit%10],angka[(menit/10)%10],ang
ka[tempjam%10],angka[tempjam/10]);
    else

tampilkan(angka[menit%10],angka[(menit/10)%10],ang
ka[10],angka[10]);
```

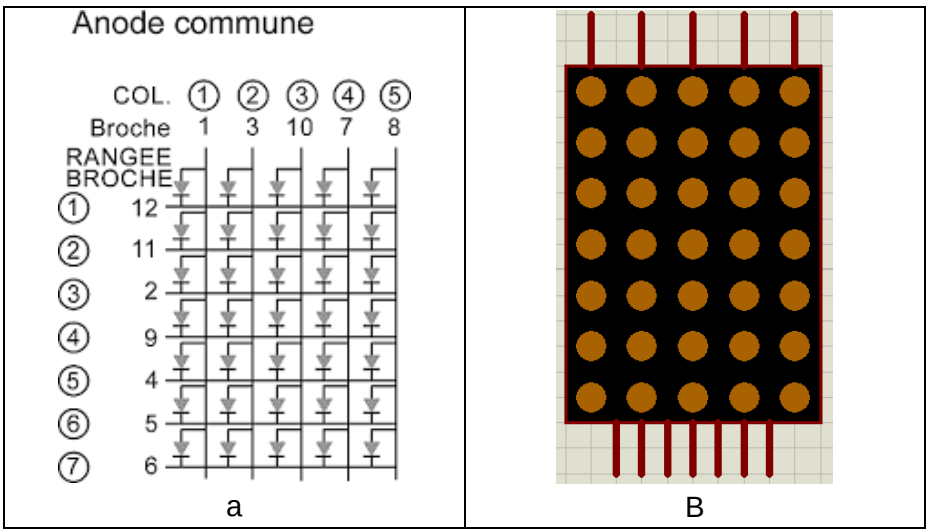
```
    if (!tombol1old and tombol1) mode = editMenit;
  }


  void editMenitFunc()
  {
    if (!tombol1old and tombol1) mode = editTgl;
  }

  void editTglFunc()
  {
    if (!tombol1old and tombol1) mode = editBulan;
  }

  void editBulanFunc()
  {
    if (!tombol1old and tombol1) mode = runn;
  }
// subrutin display
  void tampilkan(byte digit1 , byte digit2, byte
digit3, byte digit4)
  {
  digitOn++; if(digitOn==4) digitOn=0;
  switch (digitOn) {
    case 0: PORTB = 0x01; //0x01 << 0;
            PORTD = 0xFF;
            PORTD = digit4;
//angka[(menit/10)%10];
            digitalWrite(7,HIGH);
            break;
    case 1: PORTB = 0x02;  //0x01 << 1;
            PORTD = 0xFF;
            PORTD = digit3; //angka[menit%10];
```

```
            digitalWrite(7,(detik/10)%2);
            break;
    case 2: PORTB = 0x04;  //0x01 << 2;
            PORTD = 0xFF;
            PORTD = digit2;
//angka[(detik/10)%10];
            digitalWrite(7,HIGH);
            break;
    case 3: PORTB = 0x08;  //0x01 << 3;
            PORTD = 0xFF;
            PORTD = digit1; //angka[detik%10];
            digitalWrite(7,HIGH);
            break;
    }
    delay(10);
  }
```

Soal Latihan:

1. Coba buat modifikasi tampilan seven-segmen sehingga dapat menampilkan huruf a, b, c, d, e dan f!
2. Coba buat rangkaian dan program untuk menampilkan angka 1 sampai dengan angka 9. Gunakan program yang telah ada dengan modifikasi minimal.
3. Coba buat program counter untuk seven-segmen yang kaki-kakinya tidak terhubung pada satu port saja.

4. ANTARMUKA DENGAN DOT MATRIKS

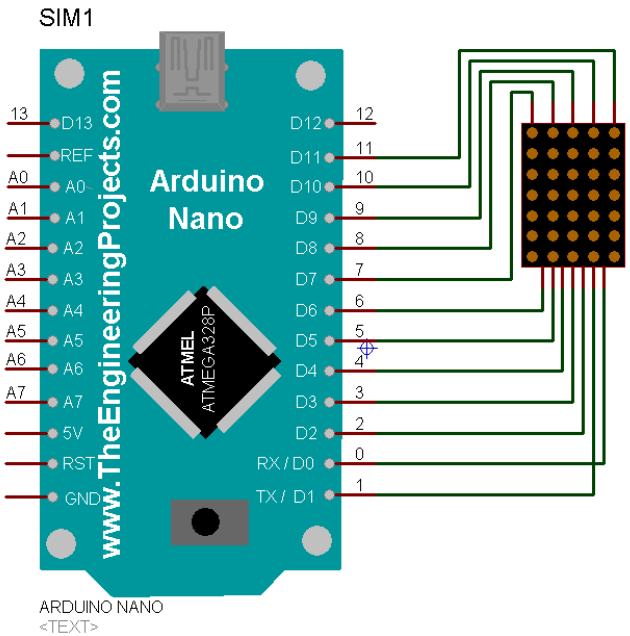


Gambar 4.1. Internal dot matriks 7x5 pada library simulasi protheus
a. Internal dot matriks b. Tampilan dot matriks pada proteus

Suatu dot matriks mempunyai rangkaian internal seperti pada gambar 4.1. Dapat dilihat bahwa satu dot yang tidak lain adalah led akan menyala jika anoda mendapatkan tegangan 5 volt dan katoda mendapatkan 0 volt.

Untuk mendapatkan tampilan yang mempunyai arti pada dot matriks harus menggunakan system scanning output. Karena dot matriks merupakan kumpulan led yang tersusun dalam sebagai baris dan kolom maka ada terdapat dua jenis scanning, yaitu scanning baris dan scanning kolom. Pada scanning baris pola dikirim dalam bentuk kolom, sedangkan barisnya akan diaktifkan secara bergantian. Sebaliknya pada scanning kolom maka pola led akan dikirim secara baris dan kolomnya akan diaktifkan secara bergantian.

Gambar 4.1 memperlihatkan salah satu cara bagaimana suatu dot matriks 7x5 terhubung dengan arduino nano.



Gambar 4.1. Rangakaian antarmuka arduino nano dengan dot matriks 7x5

4.1. Pengertian Scanning Baris dan Scanning Kolom

Cek rangkaian gambar 4.1 dengan program 1a. Lima buah led pada baris terbawah akan on sedangkan led pada baris lainnya off. Setelah 1 detik 5 buah led lagi pada baris di atasnya akan on sedangkan led pada baris lainnya akan off. Demikian berulang-ulang. Ini yang dinamakan dengan scanning baris, yang artinya satu baris led akan on selama sesaat sedangkan lainnya akan mati dan bergantian terus menerus dengan baris lainnya. Jika delay yang diberikan tepat (dengan perhitungan) maka akan terlihat seolah-olah seluruh led menyala secara bersamaan. Hal ini disebabkan mata

manusia tidak dapat melihat on-off led dengan frekuensi sekitar 25 Hz atau periode 40 milidetik.

Untuk scnaning dot matriks dengan 7 baris maka perhitungan delay yang tepat adalah sebagai berikut 25 x 7 = 175 Hz. program pada fungsi loop diharapkan dijalankan dengan frekuensi 350 Hz atau sekitar 5 mili detik.

Untuk program 1b memperlihatkan scanning kolom. Coba lihat apa perbedaannya.

Program 1a: scanning baris

```
char led = 1, i=0;
void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
}

void loop() {
  // put your main code here, to run repeatedly:
  PORTD = ~(led << i);
  PORTB=0xFF;
  delay(1000);
  i++;
  if (i==7)  i=0;
}
```

Program 1b: scanning kolom

```
char led = 1, i=0;
void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  PORTD = ~(led << i);
  PORTB=0xFF;
  delay(1000);
  i++;
  if (i==7)  i=0;
}
```

4.2. Menampilkan huruf A

Modifikasilah Program sebelumnya sehingga menjadi program 2a dan akan terlihat pola huruf A pada dot matriks.
Susunlah pola huruf A pada sebuah array.

Program 2a : Menampilkan huruf A pada scan baris

```
char huruf[]= {0b10001, 0b10001, 0b11111,
0b10001, 0b10001,0b01010, 0b00100};
char led = 1, i=0;
void setup() {
  // put your setup code here, to run once:
  DDRD=0xFF;
  DDRB=0xFF;
}

void loop() {
  // put your main code here, to run repeatedly:
  PORTD =  ~(led << i);
  PORTB=huruf[i];
  delay(3);
  i++;
  if (i==7)  i=0;
}
```

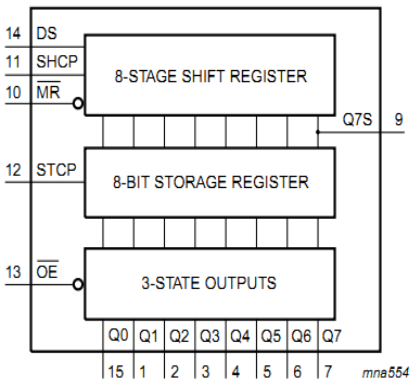
Soal Latihan:

1. Jelaskan perbedaan antara scanning baris dan scanning kolom!
2. Coba buat program untuk menampilkan huruf A, B dan C berganti ganti setiap 1 menit.
3. Coba buat program sehingga huruf A pada dot matriks akan bergeser dari kanan ke kiri berulang ulang.

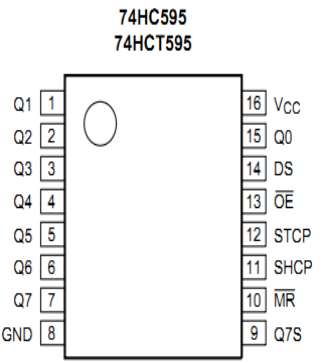
5. ANTARMUKA DENGAN DOT MATRIKS DAN SHIFT REGISTER

5.1. Shift Register

Shift Register mengubah input serial menjadi input paralel. Salah satu chip shift register adalah 75HC595.



a



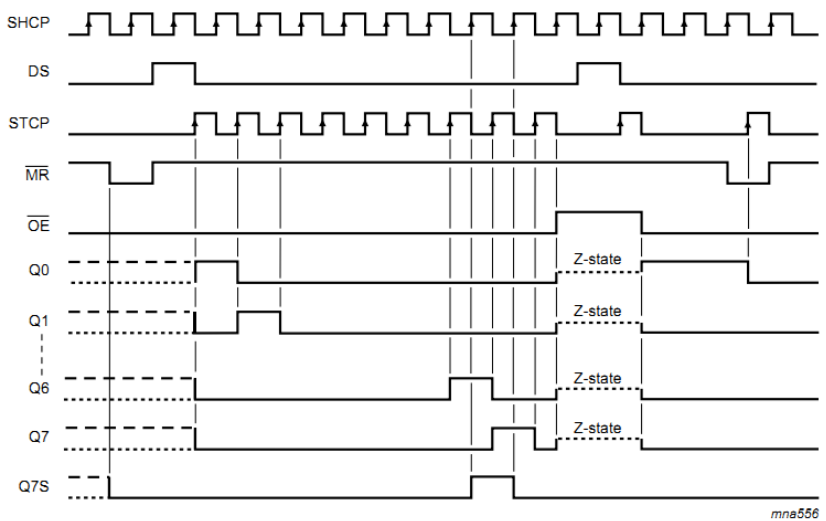
b

Control				Input	Output		Function
SHCP	STCP	$\overline{OE}$	$\overline{MR}$	DS	Q7S	Qn	
X	X	L	L	X	L	NC	a LOW-level on $\overline{MR}$ only affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6S	NC	logic HIGH-level shifted into shift register stage 0. Contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6S) appears on the serial output (Q7S).
X	↑	L	H	X	NC	QnS	contents of shift register stages (internal QnS) are transferred to the storage register and parallel output stages
↑	↑	L	H	X	Q6S	QnS	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

c

Gambar 5.1 chip shift register 74HC595
a. Functional diagram b. Tampilan Package c. Tabel Fungsi

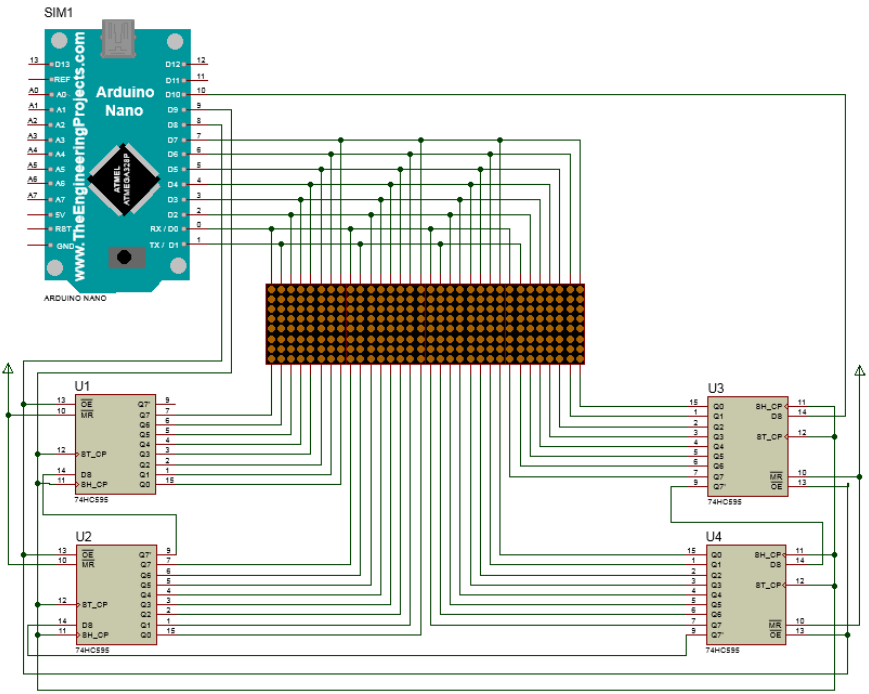
Input serial adalah DS dan input clock adalah SHCP. Pin MR adalah untuk reset dan /OE adalah output enable untuk membuat Q0 sd Q7 menjadi high impedans atau menjadi output. Pin STCP berfungsi untuk menyimpan data pada shift register ke dalam storage register. Jika STCP tidak diaktifkan maka data pada shift register tidak akan diteruskan ke buffer output. SHCP dan STCP dapat digabung jadi satu, dimana output STCP akan mengikuti output SHCP namun terlambat sebesar satu periode clock. Contoh timing diagram dari cara kerja 74HC595 dapat dilihat pada gambar 5.2



Gambar 5.2 Timing diagram cara kerja 74HC595

Salah satu aplikasi shift register untuk dot matriks 8x8 yang dikontrol oleh arduino dapat dilihat pada gambar 5.3. Pin 0 sd pin 7 (PORT D) digunakan untuk scanning baris. Untuk data serial terhubung ke pin 10 dan untuk clock terhubung dengan pin 9 dan /OE ke pin 8.Program sederhana untuk memahami cara kerja rangkaian tersebut dapat dilihat pada kode program 5.1

5.2. Menampilkan Tulisan pada Dot Matriks



Gambar 5.3 Rangkaian dot matriks dengan shift register

Jika semuanya benar maka tampilan yang didapat adalah led akan menyala satu per kolom dari kanan ke kiri dan ada leda dalam satu baris akan mati berganti baris dari bawah ke atas.

Program 5.1 program scanning baris dot matriks dengan shift register

```
char i=0;
void setup() {
  DDRB = 0xFF;
  DDRD = 0xFF;
  digitalWrite(8,HIGH);
}
```

```
void loop() {
  for(i=0; i<32;i++)
  {
    digitalWrite(10,HIGH);
    digitalWrite(9,HIGH);
    digitalWrite(9,LOW);
    digitalWrite(8,LOW);
    PORTD = 1<<i%8;
    delay(1000);
  }
}
```

Untuk dapat menampilkan tulisan pada dot matriks maka algoritma nya adalah sebagai berikut

1. Pin 8 akan on untuk mengaktifkan sinyal /OE, dengan membuatnya menjadi high impedans (agar data tidak keluar)
2. For i=0 to i<32 lakukan langkah 3 sd 4 berikut ini (i adalah kolom)
3. Beri input data DS (pin10) baris ke-j dan data kolom ke –i
4. Bangkitkan clock
5. Aktifkan sinyal clock 1 kali
6. Aktifkan sinyal pemilih baris ke-j
7. Aktifkan sinyal /OE dengan memberikan logika 0 (low).
8. Delay 1 ms
9. J++
10. Jika j==8, kembalikan j menjadi 0;
11. Selesai

Program 5.2 menampilkan tulisan FULANI pada dot matriks

Program5.2

```
char kata[]={   0,0,
  126, 10, 10, 10, 0,
  62, 64, 64, 126, 0,
```

```
  126, 64, 64, 64, 0,
  124, 18, 18, 124, 0,
  126, 8, 16, 126, 0,
  66, 126, 66,0,0};
char i=0,j=0,led=0x1;

void setup() {
  DDRB = 0xFF;
  DDRD = 0xFF;
}

void loop() {
  digitalWrite(8,HIGH);
  for(i=0; i<32;i++)
  {
    digitalWrite(10,(kata[i])&(led<<j));
    digitalWrite(9,HIGH);
    digitalWrite(9,LOW);
  }
  PORTD = ~(led<<j);
  digitalWrite(8,LOW);
  delay(1);
  j++;
  if(j==8) j=0;
}
```

5.3. Menampilkan Tulisan Bergeser dengan Shift Register

Program 5.3 merupakan pengembangan dari program 5.2 dimana tulisan Fulani bergeser dari kanan ke kiri kemudian setelah tampil seluruhnya tulisan langsung hilang.

Program 5.3 menampilkan tulisan bergeser

```
char kata[]={   0,0,
  126, 10, 10, 10, 0,
```

```
  62, 64, 64, 126, 0,
  126, 64, 64, 64, 0,
  124, 18, 18, 124, 0,
  126, 8, 16, 126, 0,
  66, 126, 66,0,0};
char i=0,j=0,led=0x1,k=0,lajugeser=0;

void setup() {
  DDRB = 0xFF;
  DDRD = 0xFF;
}

void loop() {
  digitalWrite(8,HIGH);
   for(i=k; i<34;i++)
  {
    digitalWrite(10,LOW);
    digitalWrite(9,HIGH);
    digitalWrite(9,LOW);
  }
  for(i=0; i<k;i++)
  {
    digitalWrite(10,(kata[i])&(led<<j));
    digitalWrite(9,HIGH);
    digitalWrite(9,LOW);
  }
  PORTD = ~(led<<j);
  digitalWrite(8,LOW);
  delay(1);
  j++;
  if(j==8)
  {
    j=0;
    lajugeser++;
```

```
    if (lajugeser==20)
    {
      lajugeser=0;
      k++;
      if (k==34) k=0;
    }
  }
}
```

5.4. Menampilkan Tulisan dengan Berbagai Animasi

Program 5.4.a : Tulisan timbul dari kiri

```
char kata[]={   0,0,
  126, 10, 10, 10, 0,
  62, 64, 64, 126, 0,
  126, 64, 64, 64, 0,
  124, 18, 18, 124, 0,
  126, 8, 16, 126, 0,
  66, 126, 66,0,0};
char i=0,j=0,led=0x1,k=0,lajugeser=0;

void setup() {
  DDRB = 0xFF;
  DDRD = 0xFF;
}

void loop() {
  digitalWrite(8,HIGH);
  for(i=0; i<32;i++)
  {
    if(k>i)
      digitalWrite(10,(kata[i])&(led<<j));
    else
      digitalWrite(10,LOW);
    digitalWrite(9,HIGH);
```

```
    digitalWrite(9,LOW);
  }
  PORTD = ~(led<<j);
  digitalWrite(8,LOW);
  delay(1);
  j++;
  if(j==8)
  {
    j=0;
    lajugeser++;
    if (lajugeser==20)
    {
      lajugeser=0;
      k++;
      if (k==32) k=0;
    }
  }
}
```

Program 5.4. b merupakan modifikasi dari program 5.4.a dengan menambah satu baris (cetak tebal). Dengan penambahan satu baris ini maka akan terlihat ada garis vertical yang akan bergerak dari kiri ke kanan seolah-olah seperti membuka tulisan.

Program 5.4.b Tulisan timbul dari kiri dengan garis kolom yang membuka

```
  for(i=0; i<32;i++)
  {
    if(k>i)
      digitalWrite(10,(kata[i])&(led<<j));
    else
      digitalWrite(10,LOW);
    if(k==i)      digitalWrite(10,HIGH);
    digitalWrite(9,HIGH);
```

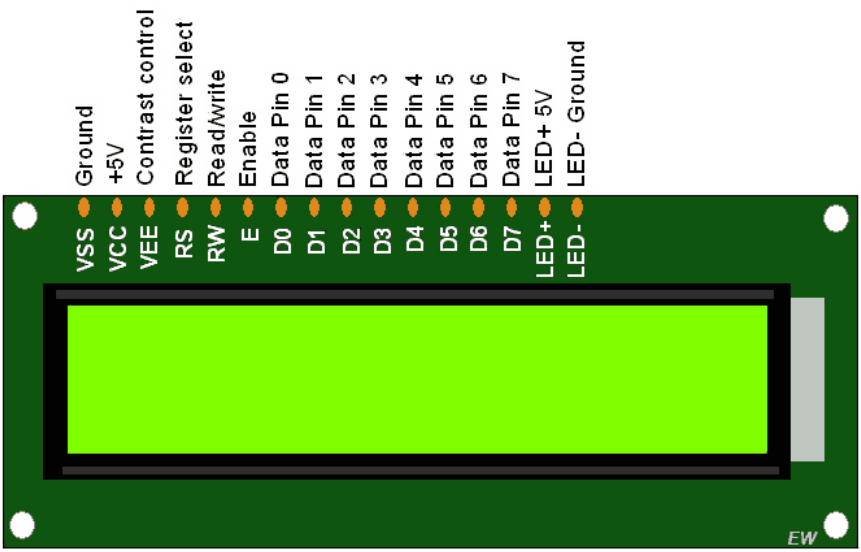
```
    digitalWrite(9,LOW);
  }
```

Soal

1. Coba buat program untuk dot matriks dengan shift register seperti sistem pada gambar 5.3 dengan animasi kalian sendiri. Jelaskan program kalian dengan algoritma seperti contoh.

6. ANTARMUKA DENGAN LCD CHARACTER

6.1. Prinsip Kerja LCD

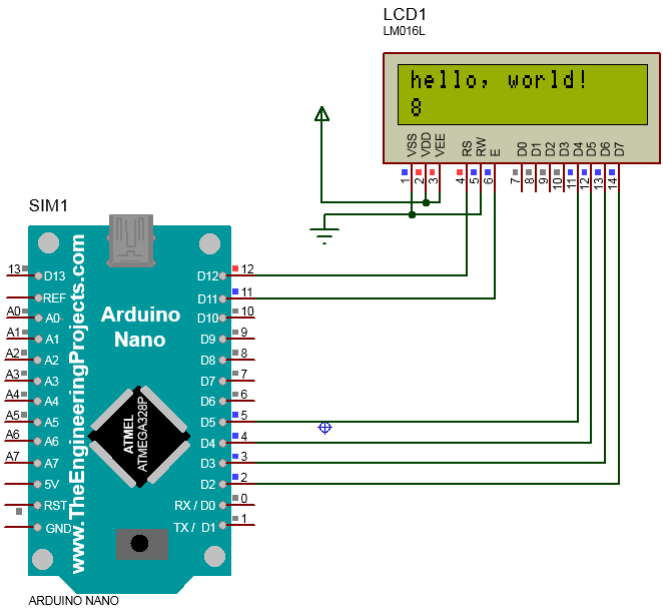


Gambar 6.1 LCD display 16x2

Untuk menghubungkan LCD karakter 16x2 ada berbagai macam cara:

- Paralel data 4 bit
- Paralel data 8 bit
- I2C

6.2. Display LCD dengan Paralel Data 4 bit



Gambar 6.2 Rangkaian LCD 16x2 terhubung paralel 4 bit dengan arduino nano

Kode 6.1

```
#include <LiquidCrystal.h>

// inisialisasi library dengan nomor pin yang
digunakan
// (RS,E,D0,D1,D2,D3);
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // memulai  LCD 16x2
  lcd.begin(16, 2);
```

```
  lcd.print("hello, world!");
}

void loop() {
  // set cursor di kolom 0, baris 1
  // penghitungan baris dan kolom dimulai dari 0
  lcd.setCursor(0, 1);
  // mencetak waktu dalam detik,fungsi milis
menghasil milidetik
  lcd.print(millis() / 1000);
}
```

Berikut adalah beberapa method yang terdapat dalam pustaka LiquidCrystal.h

- lcd.begin(16,2) : memulai penggunaan LCD, parameter method adalah ukuran LCD. Contoh 16x2.
- lcd.backlight() : menyalakan cahaya backligth
- lcd.print(“text”) : untuk menampilkan text pada LCD. Parameter method adalah tulisan yang ingin ditampilkan
- lcd.setCursor(x,y) : untuk meletakkan kursor LCD pada baris x kolom y.
- lcd.blink() : method untuk mengontrol style cursor menjadi block dan berkedip.
- lcd.cursor() : memunculkan cursor
- lcd.noCursor(): menghilangkan tampilan cursor
- TextDirection – mengontrol arah aliran teks dari cursor.
- Scroll - Scroll teks ke kiri atau kanan.
- Serial display – menerima input serial dan tampilkan.
- Autoscroll – menggeser teks secara otomatis kekiri atau kanan.

Berikut ini adalah beberapa contoh penggunaan method yang ada pada pustaka , diambil dari contoh (example) yang telah tersedia pada program IDE Arduino.

Kode 6.2 : Method Text Direction

```
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 =
3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int thisChar = 'a';

void setup() {
  // mumeulai LCD 16x2
  lcd.begin(16, 2);
  // menghidupkan tampilan kursor
  lcd.cursor();
}

void loop() {
  // mulai cetak terbalik dengan karakter 'm':
  if (thisChar == 'm') {
    // cetak kekiri untuk karakter berikutnya
    lcd.rightToLeft();
  }
  //tambahkan huruf 's':
  if (thisChar == 's') {
    lcd.leftToRight();
  }
  // reset pada 'z':
  if (thisChar > 'z') {
    // kursor ke posisi (0,0):
    lcd.home();
    // mulai pada posisi 0
    thisChar = 'a';
  }
  // cetak huruf
  lcd.write(thisChar);
```

```
  :
  delay(1000);
  // huruf selanjutnya
  thisChar++;
}
```

Kode 6.3 : Method autoscroll

```
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 =
3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
    lcd.begin(16, 2);
}

void loop() {
  // set cursor ke (0,0):
  lcd.setCursor(0, 0);
  // print dari 0 sampai 9:
  for (int thisChar = 0; thisChar < 10;
thisChar++) {
    lcd.print(thisChar);
    delay(500);
  }

  lcd.setCursor(16, 1);
  // set tampilan menjadi otomatis scroll:
  lcd.autoscroll();
  // print from 0 to 9:
  for (int thisChar = 0; thisChar < 10;
thisChar++) {
```

```
    lcd.print(thisChar);
    delay(500);
  }
  // matikan automatic scrolling
  lcd.noAutoscroll();

  // clear screen
  lcd.clear();
}
```

Soal Latihan:

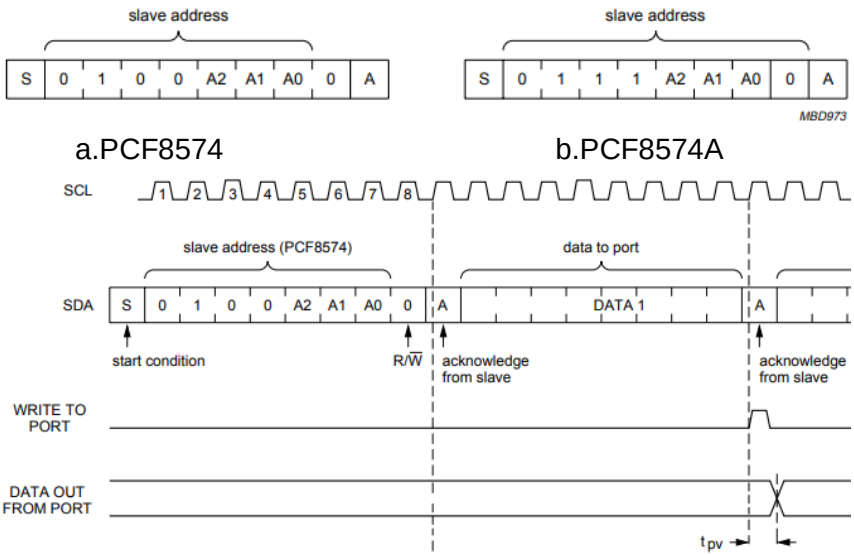
1. Buatlah suatu sistem LCD display yang dapat memperlihatkan tulisan bergerak dari kanan ke kiri. “Selamat datang di Ploiban”.
2. Buatlah sistem LCD dua baris yang memperlihatkan tulisan bergerak (scroll) dari bawah ke atas. Seperti jadwal penerbangan pada layar TV di bandara.

6.3. Display LCD dengan hubungan I2C

Untuk komunikasi LCD dengan menggunakan komunikasi secara I2C diperlukan IC PCF8574 yang berfungsi untuk mengubah data serial menjadi paralel 4 bit. Sedangkan untuk programnya diperlukan library **LiquidCrystal_I2C.h dan Wire.h.** Ada juga library yang langsung menggabungkan kedua library tersebut seperti library LCD dari adafruit (file Adafruit_LiquidCrystal-master.zip) yang dapat diunduh pada alamat https://github.com/adafruit/Adafruit_LiquidCrystal/archive/master.zi.

Sedangkan semua method yang digunakan untuk LCD paralel 4 bit dapat digunakan pada LCD I2C.

Chip i2c yang dijual di pasaran sudah dalam bentuk board digunakan adalah PCF8574, dimana alamat chip ini adalah biasanya 0x27 atau 0x3F. Penentuan alamat ini dapat dilihat dari gambar 6.3 berikut ini (sumber datasheet PCF8574 dari NXP).

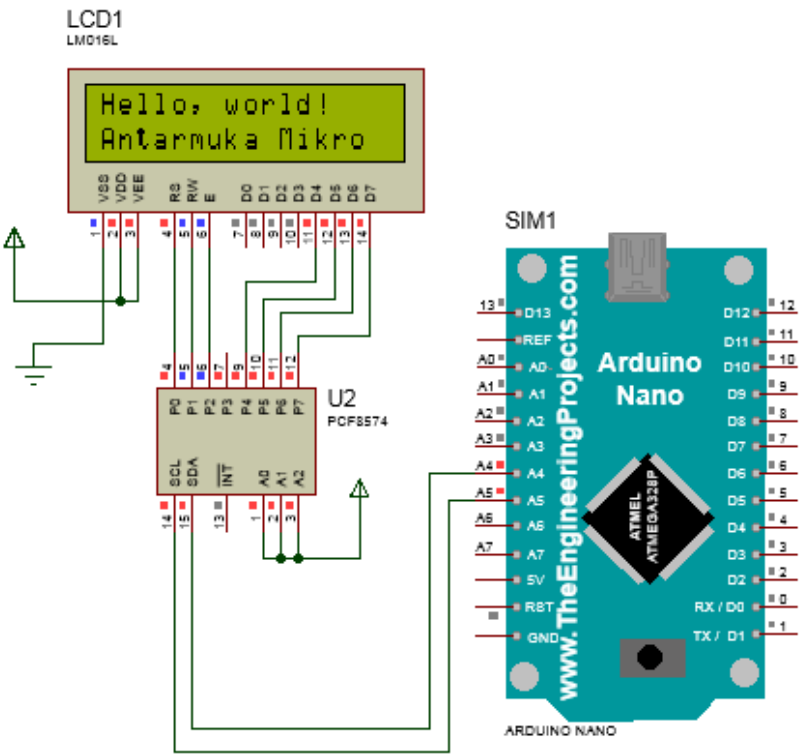


c. Timing diagram untuk mode write 8574

Gambar 6.3 Pengalamatan chip I2C PCF8574 dan PCF 8574A

Dari gambar 6.3(c) dapat dilihat 7 bit alamat (b6..b7) adalah
010 0 A2 A1 A0. Jika A2, A1, dan A0 terhubung ke vcc maka akan bernilai 27h. Sedangkan untuk PCF8574 A mempunyai format (b6..b0) adalah
011 1 A2 A1 A0, sehingga jika A2, A1, dan A0 terhubung ke vcc maka akan bernilai 0111111b =3Fh.

Sambungan LCD yang terhubung secara I2C dengan arduino nano dan IC PCF8574 dapat dilihat pada gambar 6.3. Sedangkan program 6.3 merupakan program sederhana untuk menampilkan tulisan.



Gambar 6.3 Rangkaian LCD 16x2 terhubung I2C bit dengan arduino nano

Program 5.3 Menampilkan tulisan Antarmuka Mikroprosesor dengan koneksi I2C

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

LiquidCrystal_I2C lcd(0x27,16,2);  /* set the LCD
address to 0x27 for a 16 chars and 2 line
display*/

void setup()
{
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Hello, world!");
  lcd.setCursor(0,1);
  lcd.print("Antarmuka Mikro");
}

void loop()
{
}
```

Soal Latihan:

1. Bagaimana caranya pada komunikasi I2C LCD display mempunyai alamat 26h.
2. Coba buat sistem LCD display dengan komunikasi I2C yang menggunakan 2 buah LCD display.

7. ANTARMUKA DENGAN KEYPAD

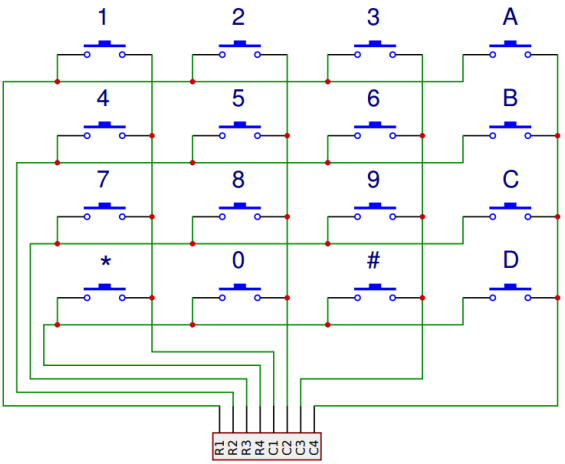
Keypad adalah peralatan input yang terdiri dari matriks button. Tampilan fisik beberapa jenis keypad dapat pada gambar 7.1 berikut ini.
Internal keypad diperlihatkan oleh gambar 7.1 berikut ini.

Gambar 7.1 Tampilan fisik 2 jenis keypad yang biasa Digunakan untuk project mahasiswa

7.1. Cara Kerja Keypad

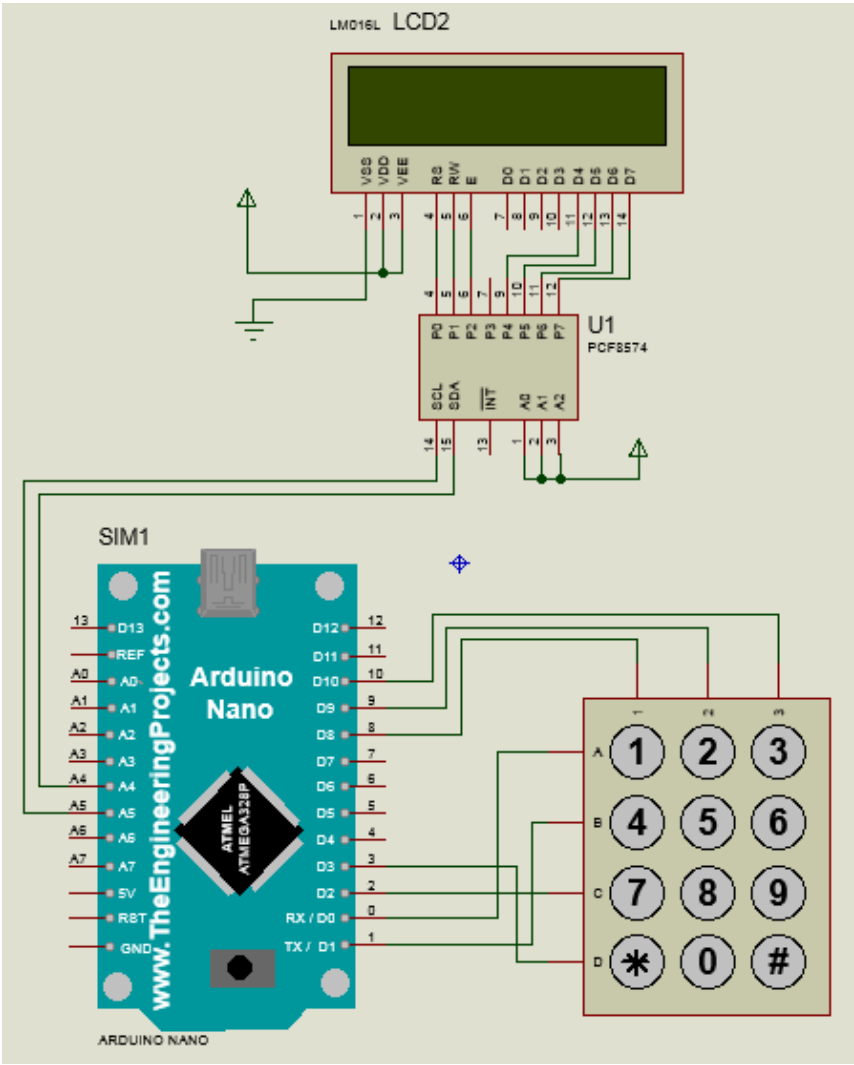
Gambar 7.2 memperlihatkan rangkaian internal sebuah keypad yang terdiri dari 4 baris dan 4 kolom (4x4) button. Cara kerja keypad tersebut sebagai berikut. Jika baris diberi tegangan maka kolom digunakan untuk membaca tegangan tersebut. Pemberian tegangan pada baris harus bergantian dengan cepat (teknik scanning). Cara yang biasa digunakan adalah dengan memberikan tegangan 5 volt untuk 3 baris dan 0 V (GND) untuk satu baris. Sedangkan untuk kolom dihubungkan dengan input pull up, sehingga jika tidak mendapatkan tegangan 0 maka input akan terbaca sebagai logika 1. Sebagai contoh, jika ingin dilakukan pembacaan pada baris pertama

(R1) maka R1 diberi logika 0 sedangkan R2, R3, dan R4 diberikan tegangan logika 1. Jika tombol pada baris R1 tidak ada yang ditekan maka kolom 1 sampai dengan kolom 4 akan terbaca logika 1 semua. Namun andai ada penekanan pada salah satu tombol, maka kolom yang terhubung dengan tombol tersebut akan terbaca sebagai logika 0. Pemberian tegangan logika 0 diurutkan berganti ganti dari baris 1,2,3 dan 4 untuk kembali lagi ke 1 dan seterusnya.



Gambar7.2 Rangkaian internal keypad 4x4

7.2. Sistem Arduino dengan Input Keypad 4x3



Gambar 7.3 Sistem Arduino dengan input keypad 3x4 dan display LCD I2C

Kode program 7.1

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

LiquidCrystal_I2C lcd(0x27,16,2);  /* set the LCD
address to 0x27 for a 16 chars and 2 line
display*/
byte angka;
byte baris,kolom,i;
void setup()
{
  DDRD=0xff;
  DDRB=0x00;
  PORTB=0xff;
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Hello, world!");
  lcd.setCursor(0,1);
  lcd.print("Antarmuka Mikro");
  delay(1000);
  lcd.clear();
}

void loop()
{
  PORTD=~(1<<0);
  kolom=PINB;
  if(!bitRead(kolom,0)) angka=1;
  else if(!bitRead(kolom,1)) angka=2;
  else if(!bitRead(kolom,2)) angka=3;
  PORTD=~(1<<1);
  kolom=PINB;
```

```
  if(!bitRead(kolom,0)) angka=4;
  else if(!bitRead(kolom,1)) angka=5;
  else if(!bitRead(kolom,2)) angka=6;
  PORTD=~(1<<2);
  kolom=PINB;
  if(!bitRead(kolom,0)) angka=7;
  else if(!bitRead(kolom,1)) angka=8;
  else if(!bitRead(kolom,2)) angka=9;
  PORTD=~(1<<3);
  kolom=PINB;
  if(!bitRead(kolom,0)) angka=10;
  else if(!bitRead(kolom,1)) angka=0;
  else if(!bitRead(kolom,2)) angka=11;

  lcd.setCursor(0,0);
  lcd.print("tekan keypad:");
  lcd.setCursor(0,1);
  lcd.print(angka);
  delay(20);
}
```

Gambar 7.3 memperlihatkan sistem Arduino dengan input keypad 4x3 dan output display LCD terhubung I2C. Empat baris keypad terhubung ke PORTD dan 3 jalur kolom terhubung ke PORTB. Program untuk sistem ini terdapat pada kode 7.1. Scan baris diperlihatkan pada instruksi

```
PORTD=~(1<<0);
```

yang dilanjutkan dengan pembacaan kolom yang disimpan pada variable kolom. Bit pada kolom dicek dengan instruksi

```
if(!bitRead(k,n)); //k adalah varibel dan n a
```

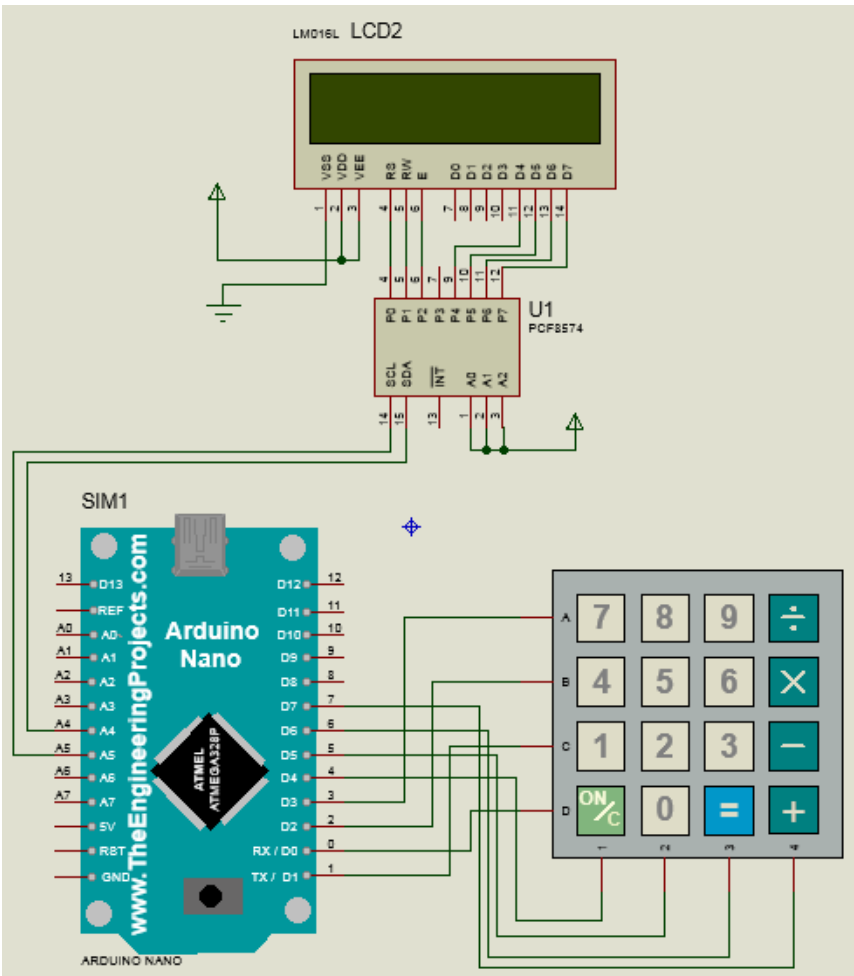
Nilai decoding angka yang ditekan disimpan pada variable angka. Yang akan dimunculkan pada LCD setelah semua baris ter-scan.

7.3. Sistem Arduino dengan Input Keypad 4x4

Suatu sistem yang memerlukan keypad dengan input key yang lebih besar (4x4) maka akan lebih baik jika program dapat dipersingkat dengan menggunakan for loop yang dapat dilihat pada kode 7.2 berikut ini

Kode 7.2

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

LiquidCrystal_I2C lcd(0x27,16,2);
char angka;
byte baris,kolom,i;
char key[4][4]={{'C','0','=','+'},
                {'1','2','3','-'},
                {'4','5','6','X'},
                {'7','8','9','/'}};
void setup()
{
```



Gambar 7.4 Sistem arduino dengan input keypad 4x4

```
DDRD=0x0f;
PORTD=0xff;
lcd.init();
lcd.backlight();
lcd.setCursor(0,0);
```

```
  lcd.print("Hello, world!");
  lcd.setCursor(0,1);
  lcd.print("Antarmuka Mikro");
  delay(1000);
  lcd.clear();
  baris=0;
}

void loop()
{
    PORTD=~(1<<baris);
    kolom=(PIND & 0xf0);
    for(int i=0;i<4;i++)
    {
      if(!bitRead(kolom,i+4))   angka=key[baris][i]
;
    }
  baris++;
  if(baris==4)baris=0;

  lcd.setCursor(0,0);
  lcd.print("tekan keypad:");
  lcd.setCursor(0,1);
  lcd.print(angka);
  delay(20);
}
```

Soal:

1. Coba sederhanakan program 7.3 dengan cara
   a. Mengganti if … else dengan perintah switch … case
   b. dengan menggunakan perintah for loop.
2. Ubahlah rangkaian 7.3 rangkaian sehingga keypad hanya menggu-nakan satu buah port yaitu PORTD! Buat juga programnya.
3. Bagaimana caranya agar karakter “*” dan “#” dapat tercetak ke LCD.

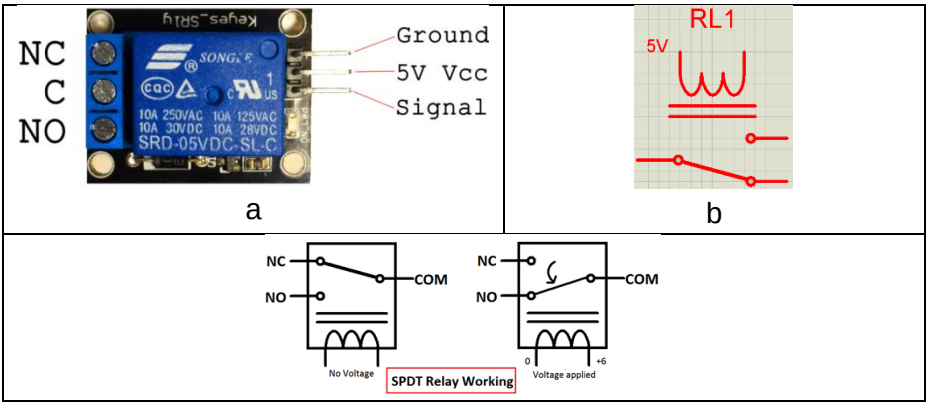
8. ANTARMUKA DENGAN RELAY

Sistem arduino untuk dapat mengendalikan sistem dengan tegangan tinggi maupun tegangan AC memerlukan alat bantu berupa relay.

8.1. Relay

Relay ada dua jenis, yaitu relay magnetic dan relay solid state (SSR). Gambar 6.1 memperlihatkan tampilan fisik salah satu relay, skematik dan cara kerja relay magnetik. Lihat gambar 6.c, ketika kumparan tidak mendapatkan input tegangan maka saklar output yang terhubung adalah com dengan NC, sedangkan bila diberikan input maka saklar terhubung com dengan NO.

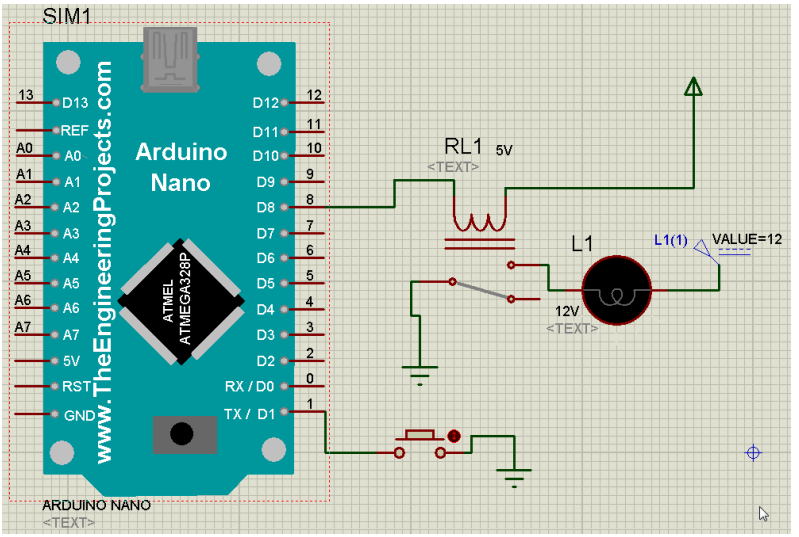
Secara umum relay digunakan untuk mengatur on-off suatu beban dari tegangan input yang lebih rendah. Biasanya tegangan output adalah tegangan jala-jala AC (220V) dan input adalah tegangan DC 5V. Namun output bias juga tegangan DC yang lebih tinggi dari tegangan input.



Gambar 8.1 Relay

a. Tampilan fisik relay pada board yang dilengkapi optokopler dan led
b. Skematik relay pada program proteus
c. Cara kerja relay magnetik

Contoh aplikasi relay adalah untuk mengendalikan lampu 12 v dengan input 5 volt, seperti yang ditunjukkan gambar 6.2. Input relay terhubung dengan pin 8 arduino. Relay akan aktif jika pin 8 diberikan nilai logika 0 (low). Sedangkan saklar terhubung dengan pin 1 arduino yang diatur sebagai input pull up.



Gambar 8.2 Rangkaian aplikasi relay mengendalikan lampu 12v dari input 5 v

8.2. Relay Digunakan pada Sistem Lampu Lalu Lintas

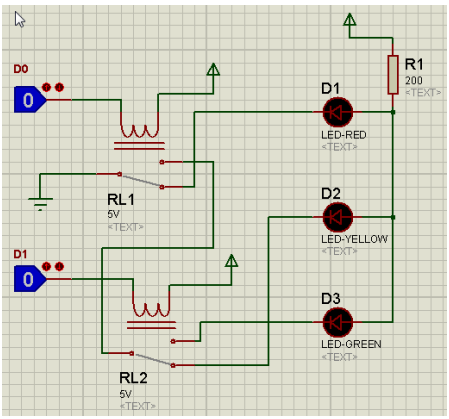
Salah satu sistem real yang menggunakan relay adalah sistem lampu lalu lintas. Gambar 6.3 memperlihatkan skematik sistem lampu lalu lintas yang dikontrol oleh arduino. Pada sistem ini ada hal yang patut dicermati, yaitu lampu lalu lintas semuanya ada 12 buah,

namun relay yang digunakan untuk mengontrolnya cuma 8 buah saja. Hal ini bisa terjadi karena relay selain sebagai pengontrol yang digunakan sebagai decoder. Setiap 2 relay digunakan untuk mengontrol 3 buah lampu (merah, kuning dan hijau) yang terdapat dalam satu persimpangan.

Gambar 6.2 memperlihatkan bagaimana 2 buah relay berpasangan mengendalikan 3 buah output. Pasangan relay tersebut membentuk suatu decoder 2 ke 3. Untuk ringkasnya sistem tersebut dapat dirangkum dalam tabel berikut.

D0	D1	Output yang aktif
0	0	RED
0	1	RED
1	0	YELLOW
1	1	GREEN

Keuntungan yang didapat dengan membuat pasangan relay menjadi decoder adalah bit input yang lebih sedikit, dimana 2 bit mampu mengontrol 3 output. Sehingga untuk 3 output hanya 2 relay yang digunakan.

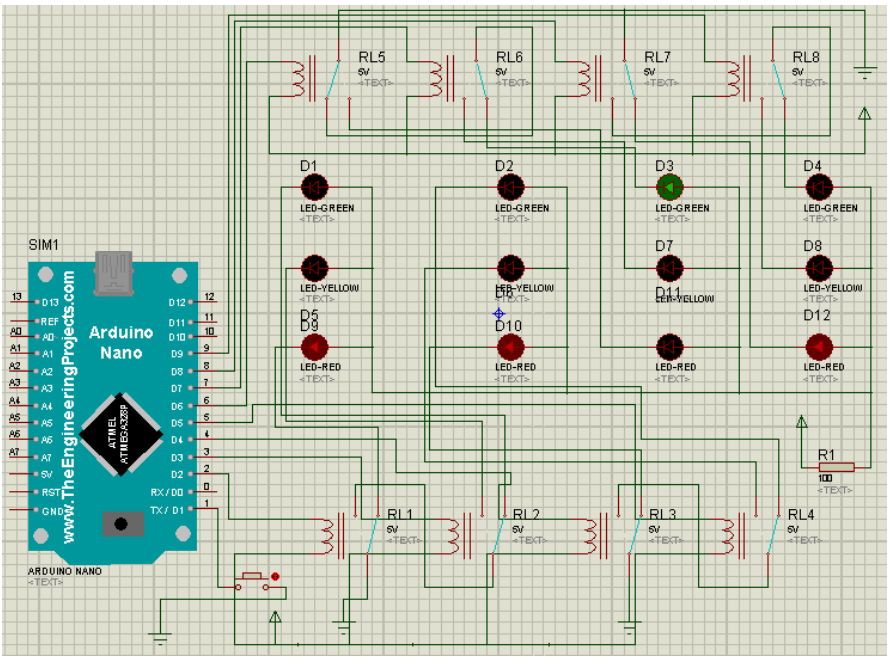


Gambar 8.3 Rangkaian 2 buah relay mengendalikan 3 buah output

Keuntungan lainnya adalah mencegah terjadinya kesalahan pada output, dimana tidak mungkin dua output akan aktif bersamaan. Dimana hal ini sering menjadi masalah dalam sistem lalu lintas. Mungkin pembaca pernah menemukan lampu merah dan kuning nyala bersamaan, atau bahkan lampu merah dan hijau nyala bersama. Hal tersebut menimbulkan kekacauan tafsir pengguna jalan yang dapat mengakibatkan kecelakaan. Sambungan output relay ke lampu pada gambar dibuat sedemikian rupa sehingga apabila arduino mati maka lampu merah yang akan aktif. Ini berguna untuk mencegah kecelakaan. Jika terjadi error pada sistem, lebih baik semua lampu pada jalur simpangan berwarna merah daripada semuanya berwarna hijau yang akan menyebabkan tabrakan arus lalu lintas.

Rangkaian komplet dari sistem lalu lintas empat simpangan dengan 8 buah relay dapat dilihat pada gambar 6.3. Sedangkan program untuk sistem tersebut ditunjukkan oleh program 6.1

Sebuah sistem lalu lintas merupakan suatu program sekuensial yang berubah dari satu state (keadaan) ke state yang lain. Setiap state berada dalam delay waktu tertentu. Kondisi output dalam setiap state dapat dirangkum dalam tabel berikut



Gambar 8.4 Rangkaian sistem lampu lalu lintas 4 simpangan dengan relay sebagai decoder

Program 8.1 Program lampu lalu lintas 4 simpangan

```
byte i=0;
void setup() {
  pinMode(1,INPUT_PULLUP);
  for(i=2;i<10;i++)
  pinMode(i,OUTPUT);
}

void loop() {
  //state 0
  digitalWrite(2,0);digitalWrite(3,0);
  digitalWrite(4,0);digitalWrite(5,0);
  digitalWrite(6,0);digitalWrite(7,0);
```

```
  digitalWrite(8,0);digitalWrite(9,0);
  delay(2000);
  //state 1
  digitalWrite(2,0);digitalWrite(3,0);
  digitalWrite(4,0);digitalWrite(5,0);
  digitalWrite(6,0);digitalWrite(7,0);
  digitalWrite(8,0);digitalWrite(9,0);
  delay(10000);
  //state 2
  digitalWrite(2,0);digitalWrite(3,0);
  digitalWrite(4,0);digitalWrite(5,0);
  digitalWrite(6,0);digitalWrite(7,0);
  digitalWrite(8,0);digitalWrite(9,0);
  delay(2000);
  //state 3
  digitalWrite(2,0);digitalWrite(3,0);
  digitalWrite(4,0);digitalWrite(5,0);
  digitalWrite(6,0);digitalWrite(7,0);
  digitalWrite(8,0);digitalWrite(9,0);
  delay(10000);

  .
  .
  .
  //state 10
  digitalWrite(2,0);digitalWrite(3,0);
  digitalWrite(4,0);digitalWrite(5,0);
  digitalWrite(6,0);digitalWrite(7,0);
  digitalWrite(8,0);digitalWrite(9,0);
  delay(10000);
}
```

Soal:

1. Kenapa sistem lampu lalu lintas ini menggunakan input tombol pull-up? Jelaskan keuntungannya!
2. Modifikasilah sistem seperti pada gambar 9.4, dengan tambahan satu tombol input untuk fasilitas pejalan kaki untuk menyeberang. Jika tombol ini ditekan maka akan masuk ke sistem state baru, dimana pada kondisi ini semua simpangan akan menyala merah. Namun state ini tidak sertamerta terjadi ketika penekanan tombol, melainkan menunggu giliran seperti lampu pada simpangan lainnya. Jadi kondisi menyeberang dianggap seolah-olah sebagai simpangan kelima.

9. INTERUPSI

Interupsi adalah proses menyela program utama yang sedang berjalan. Pada IDE Arduino program utama yang selalu dijalankan sistem adalah fungsi loop. Ketika terjadi interupsi, program utama akan berhenti sejenak untuk menjalankan program interupsi sampai selesai, setelah itu kembali lagi menjalankan program utama.

Pada low level (Bahasa assembly) proses pemanggilan fungsi interupsi sebenarnya adalah proses program lompat ke alamat vektor interupsi. Vektor interupsi adalah alamat awal program interupsi pada ROM.

Berikut adalah vektor interupsi untuk mikrokontroler AVR ATMega 328p yang terdapat pada Arduino Uno, Arduino Nano dan Arduino Pro mini.

Tabel vektor interupsi (low level)

No	Alamat Program	Sumber Interupsi	Definisi
1	0x0000	RESET	External Pin, Power-on Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2_COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2_COMPB	Timer/Coutner2 Compare Match B
10	0x0012	TIMER2_OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1_CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1_COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1_COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1_OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0_COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0_COMPB	Timer/Coutner0 Compare Match B
17	0x0020	TIMER0_OVF	Timer/Counter0 Overflow

No	Alamat Program	Sumber Interupsi	Definisi
18	0x0022	SPI STC SPI	Serial Transfer Complete
19	0x0024	USART_RX	USART Rx Complete
20	0x0026	USART_UDRE	USART Data Register Empty
21	0x0028	USART_TX	USART Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface (I2C)
26	0x0032	SPM READY	Store Program Memory Ready

Dari tabel terlihat ada 26 sumber interupsi. INT0 dan INT1 adalah interupsi eksternal pada Arduino (Uno, nano dan Pro Mini) yaitu pin 2 dan pin3 (PORTD.2 dan PORTD.3).

Pada high level bila terjadi interupsi maka program akan memanggil fungsi interupsi yang terhubung dengan sumber interupsi tersebut. Berikut adalah bentuk instruksi mengaktifkan interupsi attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);

Pada sketch yang lama yang menggunakan instruksi yang sudah tidak disarankan lagi attachInterrupt(interrupt, ISR, mode);

Contoh program Arduino untuk interupsi eksternal 0 (INT0) yang terletak pada pin 2 adalah sebagai berikut dan fungsi interupsi bernama myInterupt adalah sebagai berikut:

attachInterrupt (digital Pin ToInterrupt (2), myInterupt, RISING);

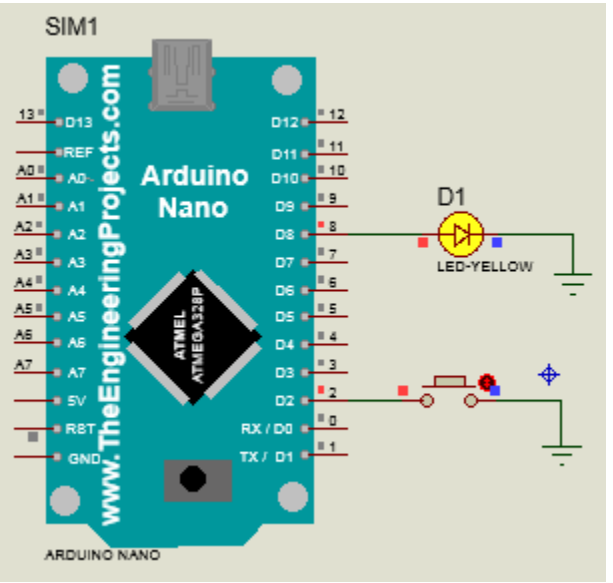
Mode menyatakan bentuk trigger dari sinyal yang akan mengaktifkan interupsi. Ada 4 nilai konstanta yaitu:

LOW akan men-trigger interupsi ketika nilai logika pin 0.

CHANGE akan men-trigger interupsi ketika nilai logika pin berubah dari 0 ke 1 atau sebaliknya.

RISING akan men-trigger interupsi ketika nilai logika pin berubah dari 0 ke 1.

FALLING akan men-trigger interupsi ketika nilai logika pin berubah dari 1 ke 0.



Gambar 9.1 Rangkaian satu led yang dikendalikan dengan interupt eksternal 0

Program 9.1

```
volatile bool led=false, button;
void setup() {
  // put your setup code here, to run once:
  pinMode(2,INPUT_PULLUP);
  pinMode(8,OUTPUT);
}

void loop() {
  button= !digitalRead(2);
  if(button)
     digitalWrite(8, led);
}
```

```
void toggle()
{
  led= not led;
}
```

Program 9.2

```
volatile bool led=false, button;
void setup() {
  // put your setup code here, to run once:
  pinMode(2,INPUT_PULLUP);
  pinMode(8,OUTPUT);
  attachInterrupt(digitalPinToInterrupt(2), toggle,
FALLING);
}

void loop() {
  // put your main code here, to run repeatedly:
  //button= !digitalRead(2);
  digitalWrite(8, led);
}

void toggle()
{
  led= not led;
}
```

Program 9.1 dan 9.2 memperlihatkan perbandingan bagaimana menyalakan dan mematikan led dengan tanpa interupsi dan menggunakan interupsi. Pada kedua program terdapat fungsi toggle yang berfungsi membuat led hidup atau mati. Pada program 9.1 fungsi toggle harus dipanggil dari fung loop dengan instruksi pemanggilan fungsi toggle. Sedangkan pada program 9.2 fungsi

toggle tidak dipanggil secara software. Tidak ada instruksi pemanggilan fungsi toggle. Fungsi toggle akan aktif jika button ditekan. Button terhubung dengan pin 2 dimana pin tersebut merupakan sumber interupsi 0. Pada fungsi setup, harus dituliskan instruksi pengaktifan interupsi yaitu attachInterrupt().

Dalam membuat fungsi ISR (interrupt service Routine) antara lain adalah :

- Usahakan fungsi dibuat singkat.
- Jangan gunakan instruksi delay dalam fungsi ISR
- Jangan gunakan instruksi serial print
- Buatlah variable yang juga digunakan pada fungsi loop menjadi volatile
- Jangan ada iinstruksi me-“enable” atau men-“disable” interupsi dalam fungsi ISR.

Soal

1. Berdasarkan tabel vector interupsi, berapa besar ruang alamat untuk setiap interupsi?
2. Apakah fungsi delay dapat di interupsi?
3. Mengapa dalam fungsi interupsi dilarang menggunakan fungsi serial print?
4. Apakah perintah untuk me-enable dan me-disable interupsi?
5. Cobalah buat sebuah sistem lampu lalu lintas dengan menambahkan tombol interupsi untuk kondisi gawat darurat, yang menyebabkan satu simpangan selalu hijau sedang simpangan lainnya merah. Kondisi ini dimulai ketika tombol interupsi ditekan dan berakhir jika tombol interupsi itu ditekan sekali lagi.

Daftar Pustaka

https://www.arduino.cc/
https://www.adafruit.com/
datasheet ATmega328P
datasheet I2C PCF8574

TENTANG PENULIS

Penulis adalah dosen Politeknik Negeri Banjarmasin dari tahun 2001 sampai sekarang. Pendidikan S-1 di Universitas Gadjah Mada Yogyakarta diselesaikan pada tahun 2001 dan Master of Science dalam bidang Computer Engineering di TU Delft Belanda pada tahun 2010. Spesialisasi penulis adalah Mikroprosesor/ Mikrokontroler, Sistem Digital dan Pemrograman.

BELAJAR ANTARMUKA ARDUINO

Secara Cepat dari Contoh

ZAIYAN AHYADI

Buku ini dapat dipergunakan luas bagi mahasiswa Jurusan Elektro secara umum. Untuk kalangan umum yang ingin belajar tentang Arduino dan antarmukanya melalui buku ini, mewajibkan telah mempunyai pengetahuan dasar tentang pemrograman, terutama Bahasa C.

Buku hanya membahas sebagian kecil dari antarmuka Arduinio dengan komponen elektronik. Pembahasan menitik beratkan konsep dasar tentang bagaimana Arduino dapat mengendalikan, membaca status logika dan berkomunikasi dengan komponen dasar elektronika. Dari metode dasar ini, diharapkan mahasiswa dapat mengembangkan untuk komponen dan peralatan elektronika lainnya.

Penerbit Poliban Press
Redaksi :
Politeknik Negeri Banjarmasin, Jl. Brigjen H. Hasan Basry, Pangeran, Komp. Kampus ULM, Banjarmasin Utara
Telp : (0511)3305052
Email : press@poliban.ac.id

ISBN 978-602-53809-4-5