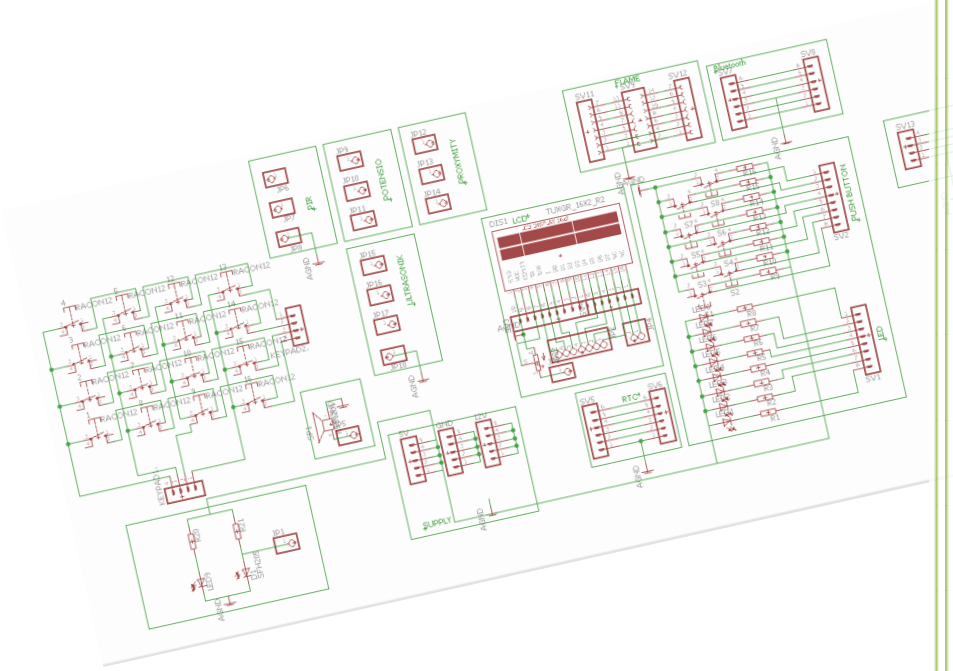
2022

PANDUAN BELAJAR MIKROKONTROLLER ARDUINO



Eko Mardianto

PANDUAN BELAJAR MIKROKONTROLER ARDUINO
(Teori dan Aplikasi)

Eko Mardianto

KATA PENGANTAR

Segala puji bagi Allah Subhanahu wata'ala, Tuhan seru sekalian alam. Penulis panjatkan kehadirat Allah Subhanahu wata'ala yang telah melimpahkan karunia-Nya sehingga penulis diberi kemampuan dan kemudahan dalam menyelesaikan buku Mikrokontroler *Arduino* ini.

Penulis berharap, para pembaca dapat dengan mudah memahami semua yang diberikan dalam tulisan ini. Adapun yang diberikan dalam buku ini diataranya membahas tentang Pengenalan Mikrokontroler Arduino, *Software* dan *Board Arduino*, Dasar Pemograman *Arduino*, Aplikasi Tombol dan Led. Diberikan juga bebarapa aplikasi lainnya diantaranya *LCD*, *ADC*, *PWM*, Sensor Suhu *IR E18-D80NK*, *Limit Switch*, *Proximity*, Sensor *Ultrasonic*, aplikasi *RTC*, Motor DC, Motor *Servo* dan yang terakhir Penggunaan sensor suhu *DHT11*.

Penulis menyadari, tulisan ini tentunya banyak kekurangan di sana sini. Untuk itu kiranya saran dan kritik yang membangun sangat dibutuhkan, agar diperoleh hasil tulisan yang benar-benar mudah dimengerti dan tentunya bermanfaat bagi para pembaca.

Pontianak, Oktober 2022

Penulis

DAFTAR ISI

PELAJARAN 1
PENGENALAN MIKROKONTROLER *ARDUINO*

TUJUAN

- Pengenalan mikrokontroler *Arduino*
- Mengetahui sejarah *Arduino*
- Mengetahui macam-macam *board* dan *shield arduino*
- Pengenalan Macam-macam Perangkat *I/O Arduino*.

PENDAHULUAN

Mikrokontroler adalah sebuah komputer kecil didalam satu *IC* yang berisi *CPU*, memori, dan perangkat antarmuka yang dapat diprogram. Mikrokontroler digunakan untuk suatu tugas dan menjalankan suatu *program*.

Penggunaan mikrokontroler dapat ditemui pada berbagai peralatan, misalnya telepon digital, *microwave oven*, televisi, mesin cuci, sistem keamanan rumah, antrian di *bank*, dll. Mikrokontroler juga dapat digunakan pada berbagai aplikasi seperti otomasi industri, traffic light, running text, jam shalat, akuisisi data, telekomunikasi, pengendali motor, pengendali *robot*, sensor suhu/kelembaban, sistem keamanan, dll. Keuntungan menggunakan mikrokontroler diantaranya adalah harganya murah, dapat di *program* berulang kali, dan dapat di *program* sesuai dengan keinginan kita.

Terdapat berbagai macam dari mikrokontroler, diantaranya keluarga mikrokontroler *MCS-51, AVR, Arduino* dan lain sebagainya. Adapun yang akan dibahas dalam buku ini adalah berkenaan dengan mikrokontroler *Arduino*. Mikrokontroler *Arduino* adalah pengendali mikro *single board* yang bersifat *open source*, yang didalamnya terdapat komponen utama yaitu sebuah *chip* mikrokontroler, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang.

Secara umum *Arduino* terdiri dari dua bagian, yaitu :

1. *Hardware* → perangkat *input/output* (*I/O*)
2. *Software*, yang meliputi :
   - *IDE* untuk menulis program,
   - *driver* untuk koneksi dengan komputer,
   - contoh program dan *library* untuk pengembangan program

Dari sisi *software*, *arduino* dapat dijalankan dimulti platform, yaitu *linux*, *windows*, atau *mac*. Sementara dari *hardware*, mikrokontroler *arduino*

berbasis *AVR* yang didalamnya sudah diberi *bootloader* dan juga sudah terdapat standar pin *I/O* nya.

SEJARAH *ARDUINO*

Arduino merupakan suatu bentuk prototipe elektronika (*electronic prototyping platform*) berbasis *open-source* yang fleksibel dan mudah digunakan baik dari sisi perangkat keras (*hardware*) maupun perangkat lunak (*software*). *Arduino* memiliki banyak kelebihan dibandingkan dengan mikrokontroler lainnya, diataranya adalah memiliki pustaka kode program (*code library*) dan tersedia modul pendukungnya (*hardware support modules*) dalam jumlah yang cukup banyak. Alasan inilah membuat pengguna pemula mudah dalam mengoperasikan *arduino* sehingga pemakainya sangat banyak.

Pertama kali *Arduino* dikembangkan melalui tesis *Hernando Barragam* yang berjudul "*Arduino Revolution Open Hardware*" pada tahun 2004, seorang mahasiswa yang berasal dari kolombia. Penggunaan *Arduino* dimulai di ruang kelas *Interactive Design Institute Ivrea* (*IDDI*), tahun 2005 di *Ivrea*, Italia. Modul *hardware Arduino* dibuat oleh *Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David A. Mellis*, dan *Nicholas Zambetti* (*Arduino*, 2011 dan *Banzi* 2008).

Tujuan awal pengembangan *Arduino* adalah untuk membantu para siswa membuat perangkat desain dan interaksi dengan harga yang murah. *Arduino* merupakan sebuah *board* mikrokontroler yang bersifat *open source*, dengan desain skematik dan pcb bersifat open source, sehingga pengguna dapat menggunakannya secara langung atau dapat pula memodifikasinya. *Software* yang digunakan untuk membuat, mengkompilasi dan meng-*upload* program yaitu *Arduino IDE* (*Integrated Development Environment*) juga bersifar *open source*.

Arduino terdiri dari beberapa *input output* (*I/O*) yang sudah *fix* dan bisa digunakan dengan mudah. *Arduino* dapat digabungkan dengan modul elektro yang lain sehingga proses perakitan jauh lebh efisien. Para desainer hanya tinggal membuat *software* untuk mendayagunakan rancang *hardware* yang ada. *Software* jauh lebih mudah untuk dimodifikasi tanpa harus memindahkan kabel. Saat ini *arduino* sangat mudah dijumpai dan ada beberapa perusahaan yang mengembangkan sistem *hardware open source* ini.

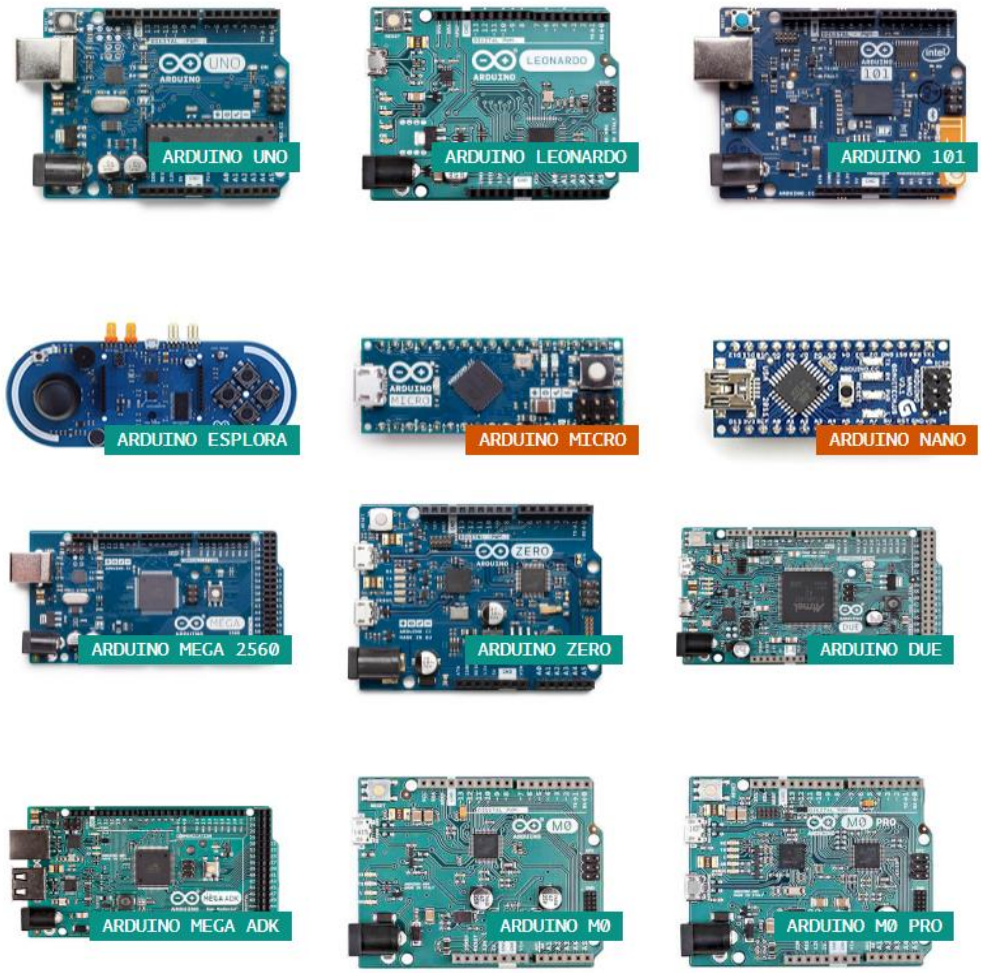
Pada *Arduino board* terdapat *chip* / *IC Boatloader* yang telah diisi program yang dinamakan *Arduino bootloader*. Penggunaan *bootloader* memungkinkan *user* meng-*upload code program* tanpa menggunakan *hardware*

tambahan (tanpa menggunakan programmer dari luar seperti : *AVR-ISP, STK500, parallel programmer, usb programmer*). *Bootloader* akan aktif selama beberapa detik ketika *board* mengalami *reset*.

Hasil kompilasi dari *Arduino Sofware* dapat digunakan dan dijalankan tidak hanya pada *arduino board* tetapi juga dapat dijalankan di sistem mikrokontroler *avr*. Dengan menggunakan *bootloader*, program yang dimasukkan ke *flash* akan semakin kecil.

MACAM-MACAM PAPAN *ARDUINO*

Terdapat berbagai macam papan *board arduino* yang dapat diakses di *website*-nya *https://www.arduino.cc/en/Main/Products*, diantaranya ditunjukkan pada gambar 1.1 berikut ini :



Gambar 1.1. Berbagai macam papan *Arduino*

SHIELD ARDUINO

Shield adalah *board* yang digunakan untuk mempermudah dalam pemasangan berbagai sensor atau aktuator (motor, *LCD*, dll). Dengan penggunaan *shield* ini dapat menambah kapasitas *input* atau *output* dari *arduino* dengan cara dipasangkan diatas papan *Arduino*.

Gambar 1.2. Berbagai macam *shield Arduino*

PERANGKAT *I/O ARDUINO*

SENSOR

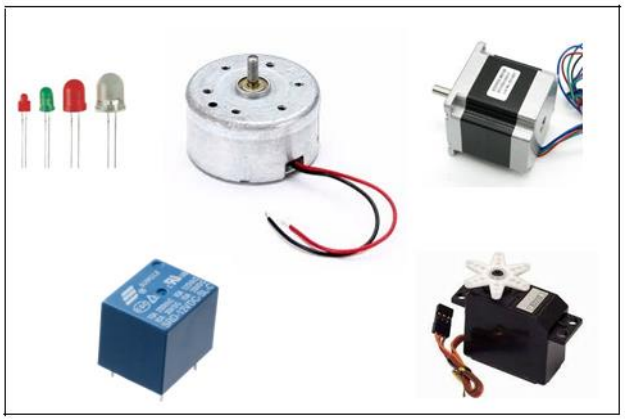
Sensor adalah suatu perangkat yang digunakan untuk memberi masukan data/nilai ke *Arduino* yang selanjutnya diproses. Terdapat beberapa sensor diantaranya adalah sensor kelembaban, suhu, sensor suara, sensor gerak, sensor sentuh, sensor *ultrasonic*, sensor cahaya dan lain sebagainya.

Gambar 1.3 menunjukkan sensor yang dapat digunakan pada *Arduino*.

Gambar 1.3. Berbagai macam sensor *Arduino*

AKTUATOR

Aktuator adalah komponen yang merupakan hasil keluaran dari mikrokontroler. Komponen-komponen yang termasuk dalam aktuator adalah *LED*, motor *DC*, Motor *Servo*, Motor *Stepper*, Lampu, Pemanas, *Relay*, Selenoid, *electric valve*, Pompa Air, dan sebagainya.



Gambar 1.4. Berbagai macam aktuator *Arduino*

MODUL

Modul adalah suatu rangkaian yang memiliki fungsi khusus, yang dapat dihubungkan dengan *arduino* untuk mendukung fungsi-fungsi tertentu sesuai dengan kebutuhan. Contoh modul adalah : modul *sd card*, modul *RTC*, Modul kamera, *Bluetooth*, *GPS*, *MP3* dan lain sebagainya.

Gambar 1.5. Berbagai macam modul *Arduino*

MEKANIK ROBOT

Mekanik Robot adalah komponen pendukung proyek *robotic* yang bisa berupa dudukan servo, tangan *robot*, roda gigi, dan sebagainya. Contoh komponen-komponen *robotic* adalah sebagai berikut :

Gambar 1.6. Berbagai macam mekanik *robotic*

SOAL ESSAY

1. Sebuatkan jenis-jenis mikrokontroler yang pernah ada sebelum *Arduino* ?
2. Siapa dan tahun berapa pertama kali *Arduino* dibuat dalam bentuk *board* mikrokontroler dan digunakan dalam ruang kelas untuk pembelajaran ?
3. Sebutkan macam-macam *board arduino* yang saudara ketahui ?
4. Sebuatkan macam-macam *shield arduino* yang saudara ketahui ?
5. Apa saja perangkat *I/O* yang dapat digunakan sebagai pendukung dalam aplikasi *Arduino* ?
6. Apa yang saudara ketahui tentang *actuator*, sebutkan contohnya ?

PELAJARAN 2
SOFTWARE DAN *BOARD ARDUINO*

TUJUAN

- Pengenalan *Arduino IDE*
- Memahami Pengujian Papan *Arduino*
- Pengenalan mikrokontroler *Arduino*.

ARDUINO IDE

Software arduino yang akan digunakan adalah *driver* dan *IDE*. *IDE arduino* adalah *software* yang canggih, ditulis dengan menggunakan bahasa *java*. *IDE arduino* terdiri atas :

- Editor program, sebuah *windows* memungkinkan pengguna menulis dan mengedit program dalam bahasa *processing*.
- *Compiler*, sebuah modul yang mengubah kode program (bahasa *Processing*) menjadi kode biner.
- *Uploader*, sebuah modul yang memuat kode biner dari komputer ke dalam memori didalam papan *Arduino*.

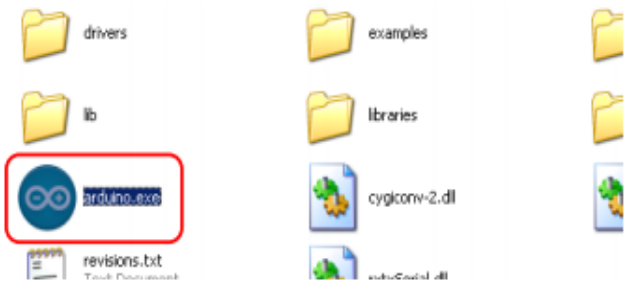
Aplikasi *Arduino IDE* berfungsi untuk menyusun kode program (*sketch*). Selanjutnya *sketch* tersebut di *upload* ke unit *arduino* melalui kabel *USB*. Aplikasi *Arduino IDE* dapat diperoleh pada alamat situs *http://arduino.cc/en/Main/Software*. Beberapa *software* ditulis menggunakan bahasa pemrogramam *java* termasuk *IDE*-nya, sehingga tidak perlu di*install* seperti *software* pada umumya tetapi dapat langsung dijalankan selama komputer telah terinstall *Java runtime*.

MENGUJI KONEKSI PAPAN *ARDUINO*

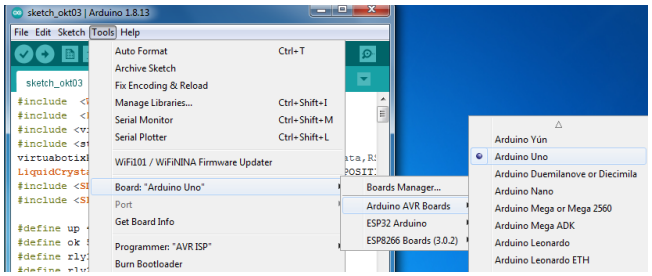
Sebelum melakukan pengujian koneksi dari papan *arduino*, terlebih dahulu pengguna melakukan instalasi *Arduino IDE*. Setelah dilakukan instalasi selanjunya dilakukan uji koneksi papan *arduino* sebagaimana berikut ini.

Langkah-langkah melakukan pengujian koneksi computer dan papan *Arduino*.

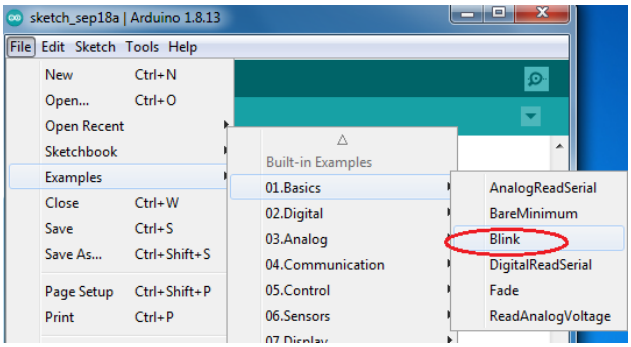
- Jalankan *Arduino IDE* dengan menjalan file bernama *arduino.exe* pada lokasi *software Arduino*.



- Walaupun tampak seperti program *windows* pada umumnya, namun sebetulnya program ini adalah sebuah program *java*. Jika anda menemukan sebuah pesan kesalahan kemungkinan basar pada komputer belum terinstall *Java Runtime Environment* (*JRE*) atau *Java Development Kit* (*JDK*). Untuk mendapatkan dapat dikunjungi situs web http://www.oracle.com.
- Jalankan menu ***Tools → Board*** kemudian pilih tipe papan yang sesuai.



- Jalankan meu ***File → Examples → 1. Basics → Blink***. Program ini sebagai contoh sederhana untuk menguji koneksi *Arduino*. Penjelasan dari program ini akan diberikan pada materi di Pelajaran 4.



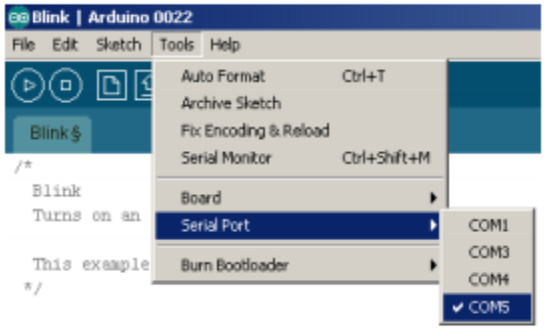
- Pada *toolbar* klik tombol ***Upload*** untuk memuat *sketch* tersebut ke dalam papan *Arduino*.

- Jika program benar maka *sketch* akan dimuat, ditandai dengan pesan berhasil seperti gambar berikut ini.

- Namun jika belum beruntung akan maka akan terdapat pesan kesalahan seperti gambar berikut ini.

Walaupun tidak cukup jelas menjelaskan apa masalahnya, tapi umumnya karena ***IDE*** belum dikonfigurasi dengan benar sehingga komputer dan papan *Arduino* tidak dapat berkomunikasi.

Solusinya adalah dengan mengganti pilihan serial *port* melalui menu ***Tools*** → **Serial *Port***. Jika Anda belum yakin pada *port* nomor berapa papan *Arduino* itu terhubung, coba pilih sebuah nomor *port* lalu jalankan *upload* seperti langkah sebelumnya. Jika pesan kesalahan masih muncul, ganti nomor *port*-nya dan lakukan berulang-ulang sampai *upload* berhasil.



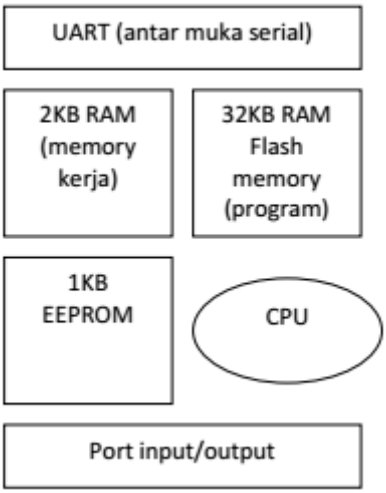
Saat *sketch* yang sudah dimodifikasi tersebut berhasil dimuat ke dalam papan *Arduino* maka tampak lampu *LED* menyala dan padam dengan frekuensiyang lebih cepat. Silahkan lakukan eksperimen sendiri misalnya menambah *delay* dan lihat apa yang terjadi.

PAPAN *ARDUINO UNO*

Terdapat bermacam-macam bentuk papan *Arduino* yang disesuiakan dengan peruntukannya salah satunua adalah *ArduinoUno* seperti diperlihatkan berikut ini.

Gambar 2.1. *Arduino Uno*

Berikut ini merupakan blok sederhana dari mikrokontroler *ATmega328* sebagai pendukung mikrokontroler *Arduino Uno*.



Gambar 2.2. Blok Sederhana mikrokontroler *ATmega328*

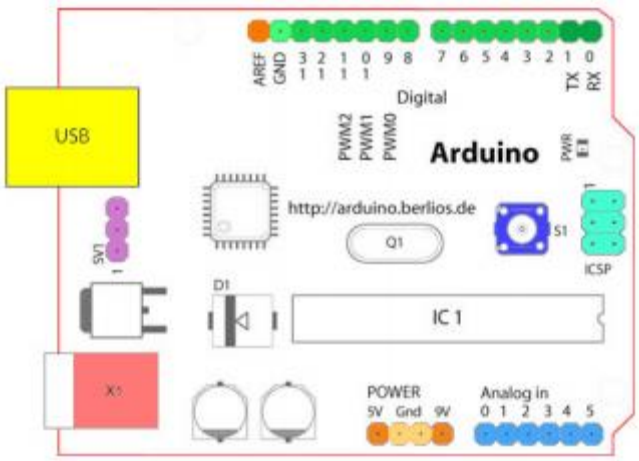
Gambar 2.2 merupakan blok sederhana mikrokontroller *ATmega328* yang dapat dijelaskan sebagai berikut :

- *Universal Asynchronous Receiver/Transmitter* (*UART*) adalah antar muka yang digunakan untuk komunikasi serial seperti pada *RS-232, RS-422* dan *RS-485*.
- 2 kb *RAM* pada memori kerja bersifat *volatile* (hilang saat daya dimatikan), digunakan oleh variable-variabel didalam program.
- 32 kb *RAM flash* memori bersifat *non-volatile*, digunakan untuk menyimpan program yang dimuat dari komputer. Selain program, *flash* memori juga menyimpan *bootloader*.

- *Bootloader* adalah program inisiasi yang ukurannya kecil, dijalankan oleh *CPU* saat daya dihidupkan. Setelah bootloader dijalankan, berikutnya program di dalam *RAM* akan dieksekusi.
- 1 kb *EEPROM* bersifat *non-volatile*, digunakan untuk menyimpan data yang tidak boleh hilang saat daya dimatikan. Tidak digunakan pada papan *Arduino*.
- *Central Processing Unit (CPU)*, bagian dari mikrokontroler untuk menjalankan setiap instruksi dari program
- *Port I/O*, pin-pin untuk menerima data (*input*) *digital* atau *analog*, dan mengeluarkan data (*output*) digital atau *analog*.

BAGIAN-BAGIAN PAPAN *ARDUINO*

Gambar 2.3 menunjukkan bagian dari papan *arduino* tipe *USB*.

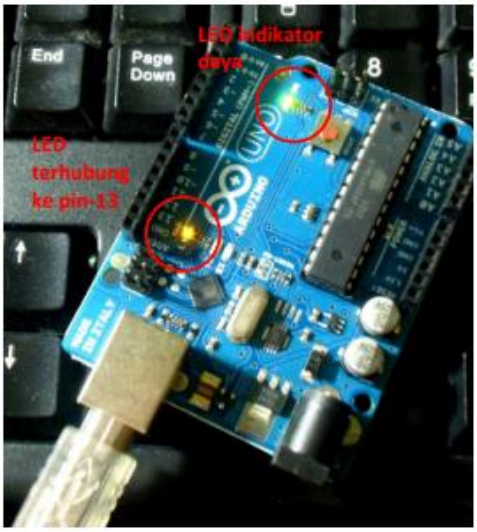


Gambar 2.3. Papan *Arduino*

- 14 pin *Input*/*Output* digital (0 – 13)
  Berfungsi sebagai *input* atau *output*, dapat diatur oleh program menggunakan fungsi *pinMode*(), *digitalWrite*(), dan *digitalRead*(). Fungsi-fungsi tersebut beroperasi di tegangan 5 Volt. Setiap pin dapat memberikan atau menerima atau menerima suatu arus maksimum 40 mA dan mempunyai sebuah *resistor pull-up* 20 – 50 kΩ.
- *PWM* : pin 3, 5, 6, 9, 10, dan 11.
  Memberikan 8 bit *PWM* (*analog output*) yang dapat diatur dengan memanfaatkan fungsi *analogWrite*(). Nilai sebuah pin *output analog* dapat di program antara 0 – 255, dimana hal itu dapat mewakili tegangan 0 – 5 V.

- *USB*
  Berfungsi untuk :
  - memuat program dari komputer ke dalam papan
  - komunikasi serial antara papan dan komputer
  - memberi daya listrik kepada papan
- Sambungan SV1
  Sambungan atau jumper untuk memilih sumber daya papan, apakah dari sumber eksternal atau menggunakan *USB*. Sambungan ini tidak diperlukan lagi pada papan *arduino* versi terakhir karena pemilihan sumber daya eksternal atau *USB* dilakukan secara otomatis.
- Q1 – Kristal (quartz crystal oscillator)
  Jika mikrokontroler dianggap sebagai sebuah otak, maka kristal adalah jantungnya karena komponen ini menghasilkan detak-detak yang dikirim kepada mikrokontroler agar melakukan sebuah operasi untuk setiap detaknya. Kristal ini dipilih yang berdetak 16 juta kali per detik (16 MHz).
- Tombol *Reset* S1
  Untuk mereset papan sehingga program akan mulai lagi dari awal. Perhatikan bahwa tombol *reset* ini bukan untuk menghapus program atau mengosongkan mikrokontroler.
- *In-Circuit Serial Programming* (*ICSP*)
  Port ICSP memungkinkan pengguna untuk memprogram mikrokontroler secara langsung, tanpa melalui *bootloader*. Umumnya pengguna *Arduino* tidak melakukan ini sehingga *ICSP* tidak terlalu dipakai walaupun disediakan.
- IC 1 – *Microcontroller ATmega*
  Komponen utama dari papan *arduino*, didalamnya terdapat *CPU*, *ROM* dan *RAM*
- X1 – Sumber daya eksternal
  Jika hendak di*supply* dengan sumber daya eksternal, papan *Arduino* dapat diberikan tegangan *DC* antara 9 – 12 V
- *ADC* : 6 pin *input analog* (A0 – A5)
  6 buah pin *ADC* ini memberikan 10 bit resolusi (pin *input* antara 0 – 1023) . Secara *default*, keenam pin *ADC* mengukur masukan dari 0 sampai 5 Volt.
- Serial : pin 0 (RX) digunakan untuk menerima dan pin 1 (TX) digunakan untuk memancarkan serial data *TTL*. Kedua pin ini dihubungkan ke pin-pin yang sesuai dari *chip* serial *Atmega8U2 USB* ke *TTL*.

- Eksternal *Interrupts* : pin 2 dan 3. Pin-pin ini dapat dikonfigurasikan untuk dipicu sebuah *interrupt* (gangguan) pada sebuah nilai rendah, suatu kenaikan atau penurunan yang besar, atau suatu perubahan nilai. Biasanya menggunaka fungsi *attachInterrupt*().
- SPL : pin 10 (*SS*), 11 (*MOSI*), 12 (*MISO*), 13 (*SCK*). Pin-pin ini mensupport komunikasi *SPI* menggunakan *SPI library*.
- Led : pin 13. Terdapat sebuah *LED* yang terpasang, terhubung ke pin digital 13, sebagaimana ditunjukkan gambar 2.4. Ketika pin berlogika *High Led* akan menyala dan ketika pin berlogika *Low* Led akan padam,.



Gambar 2.4. *Arduino Uno*

DAYA

Arduino Uno dapat di *supply* menggunakan koneksi *USB* atau dengan sebuah *power supply* eksternal. *Supply* eksternal (*non USB*) dapat diperoleh dari adaptor atau baterai. Adaptor dapat dihubungkan dengan mencolok sebuah *center-positive-plug* yang panjangnya 2,1 mm ke *power jack* dari *board*. Kabel *lead* dari baterai dapat dimasukkan dalam header pin *Ground* dan pin *Vin* dari konektor *Power*.

VIN (tegangan *input* ke *Arduino board*) dapat diberikan sebagai supply eksternal yang besarnya $\pm$ 7 – 12 volt..

SOAL ESSAY

1. Sebuatkan 3 bagian penting dari *Arduino IDE* yang dapat dimanfaatkan oleh *user* ?
2. Bagaimana cara menguji koneksi dari papan *arduino* ?
3. Berapa kapasitas *RAM* dan *Flash* pada papan *Arduino Uno* ?
4. Fasilitas apa saja yang saudara ketahui dari papan *Arduino uno* ?

PELAJARAN 3
DASAR PEMROGRAMAN *ARDUINO*

TUJUAN

- Memahami struktur penulisan pemrograman pada *arduino*
- Mengenal tipe data pada *arduino*
- Memahami Penggunaan operator

STRUKTUR PEMROGRAMAN *ARDUINO*

Program *arduino* dapat dibagi menjadi tiga bagian utama yaitu :

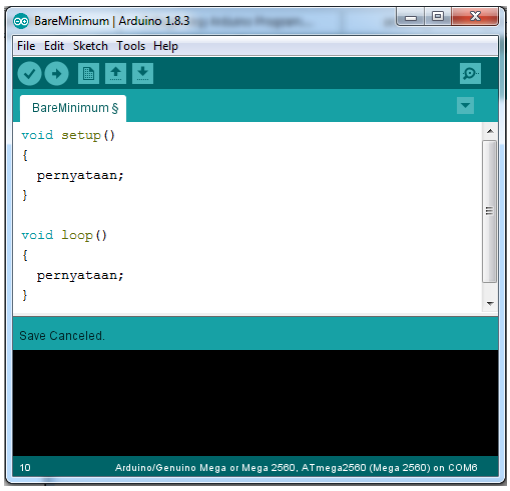
- Struktur
- *Value* (variabel dan kontanta)
- Fungsi

STRUKTUR

Struktur terbagi menjadi dua fungsi utama :

- Fungsi *Setup( )*
  Fungsi *setup( )* hanya dipanggil satu kali saja saat program mulai berjalan. Fungsi *setup()* berfungsi untuk melakukan inisialisasi mode pin atau untuk memulai komunikasi serial. Fungsi *setup()* ini harus ada meskipun tidak ada program yang akan dieksekusi.
- Fungsi *Loop( )*
  Setelah menyiapkan inisialisasi pada *setup(),* berikutnya membuat fungsi *loop().* Seusai dengan namanya, fungsi akan mengulang program secara terus menerus, sehingga program akan berubah dan merespon sesuai *input* yang diberikan. Fungsi *loop()* ini akan secara aktif mengontrol *board Arduino*.

Gambar 3.1 menunjukkan contoh dari program minimum dari *arduino*. Setiap pembuatan program minimum harus ada fungsi *setup()* dan fungsi *loop().*



Gambar 3.1. Struktur program minimum *Arduino*

TIPE DATA

Berikut ini ditunjukkan tipe data yang dapat digunakan dalam program *arduino*.

void	*Boolean*	*char*	*Unsigned char*	*byte*	*int*	*Unsigned int*	*word*
long	*Unsigned log*	*short*	*float*	*double*	*array*	*String-char array*	*String-object*

Beberapa tipe data yang perlu diketahui, ditunjukkan dalam tabel 3.1

Tabel 3.1. Tipe Data

TIPE DATA	UKURAN (BIT)	JANGKAUAN NILAI
boalean	1	0, 1 (tipe data bit hanya digunakan untuk variabel global)
char	8	-128 to 127
unsigned char	8	0 to 255
byte	8	0 to 255
int	16	-32768 to 32767
unsigned int	16	0 to 65535
word	16	0 to 65535
long	32	-2147483648 to 2147483647
unsigned long	32	0 to 4294967295
short	16	-32768 to 32767
float	32	± 1,175e-38 to ± 3,402e38
double	32	± 1,175e-38 to ± 3,402e38

VARIABEL DAN KONSTANTA

Variabel berfungsi untuk menampung nilai angka dan memberikan nama sesuai dengan kebutuhan membuat program. Dengan menggunakan variabel, maka nilai yang ada dapat dirubah. Hal ini berbeda dengan konstanta, dimana nilainya tidak bisa berubah. Pemberian nama untuk variabel seharusnya sesuai dengan deskripsinya, sehingga program yang dibuat mudah dibaca.

Sebuah variabel perlu dideklarasikan terlebih dahulu, dan bisa digunakan sebagai penampung *input* yang akan disimpan atau diberi nilai awal.

```
Int InputVariabel = 0;          // mendeklarasikan nilai variabel dengan nilai 0
InputVariabel = analog(2);      // set variabel ke nilai dari pin 2 analog
```

Keterangan contoh program diatas, *Input*Variabel adalah variabel itu sendiri. Baris pertama mendeklarasikan nilai yang menggunakan tipe *integer*. Baris kedua merupakan *input* variabel sebagai penampung nilai hasil pembacaan pin 2 *analog*. Hal ini memberikan keuntungan bahwa nilai *analog* pin 2 bisa diakses dari manapun di dalam program.

Selain variabel ada juga konstanta yang digunakan untuk mempermudah dalam membaca program. *Integer contant* adalah angka yang secara langsung dapat dituliskan di *sketch*, misalnya 12300. Secara *default*, angka ini diperlakukan sebagai *int*, namun kita juga bisa mengubahnya dengan U dan L.

Secara norma, konstanta *integer* diperlakukan sebagai bilangan basis 10 (desimal) *integer*, tetapi notasi khusus bisa digunakan untuk menyatakan bilangan basis yang berbeda, hal ini bisa dilihat pada tabel 3.2.

Tabel 3.2. Format Bilangan Berbasis

Base	Contoh	Format	Komentar
10 (*decimal*)	12345	-	-
2 (*binary*)	B110011	*Capital* ‘B’	Hanya valid jika 8 bit, karakter yang ada 1 dan 0
8 (*octal*)	0234	Diawali dengan 0	Karakter yang valid 0 sampai 7
16 (*hexadecimal*)	0xB8	Diawali dengan 0x	Karakter yang valid 0 sampai 9, A-F, a-f

Secara *default*, sebuah konstanta *integer* mempunyai keterbatasan dalam hal range jangkauan nilainya. Untuk melakukan spesifikasi pada konstanta *integer* dengan data tipe yang lain, maka perlu dilakukan seperti berikut ini :

- 'u' atau 'U' untuk mengubah *default* konstanta ke bentuk *unsigned* data format. Contoh : 33u
- 'l' atau 'L' untuk mengubah *default* kontanta ke bentuk *long* data format. Contoh : 100000L
- 'ul' atau 'UL' untuk mengubah *default* kontanta ke bentuk *unsigned long constant*. Contoh : 32423533UL

Selain konstanta *integer* ada juga *floating constant* yang digunakan untuk membuat kode lebih mudah dibaca. *Floating point constant* dilakukan saat proses meng-*compile* terjadi.

Contoh :

```
n =   .006;
```

FUNGSI

Fungsi adalah sekumpulan program yang diberi nama khusus, yang dapat dieksekusi dengan cara memanggil fungsi tersebut. *Setup()* dan *loop()* dalam hal ini juga termasuk fungsi.

Sebuah fungsi bisa dibuat sesuai dengan kebutuhannya untuk melakukan suatu pekerjaan yang berulang ataupun bisa membuat program jadi lebih sederhanan.

Langkah-langkah dalam membuat suatu fungsi :

- Deklarasikan terlebih dahulu tipe suatu fungsi.
- Tipe tersebut akan mendapatkan suatu nilai parameter dari fungsi tadi. Misalnya '*int'* untuk tipe fungsi *integer*. Jika tidak ada nilai yang dikembalikan maka tipe fungsi tersebut akan kembali ke *void*.
- Setelah tipe maka diberikan nama fungsinya, dan dalam kurungnya masukkan parameter yang akan diberikan ke dalam fungsi.
- Strukturnya sebagai berikut :

```
tipe functionName(parameter)
{
        Pernyataan;
}
```

Contoh fungsi 1 : (fungsi delay Val())

```
int delay Val( ) {
  int v;                          // membuat variable lokal
  v = analogRead(pot);            // membaca nilai potensiometer
  v /=4;                          // mengkonversi 0 – 1023 ke 0 – 255
  return v;
}
```

Contoh fungsi 2 : (menghitung nilai dari k)

```
/* program utama     */
void loop {
  int i = 2;
  int j = 3;
  int k;
  k = myMultiplyFunction(i, j);     //    k sekarang berisi 6
}
```

```
/* program function-nya beserta parameter x dan y, lalu  dikembalikan
ke program utama */
```

```
int myMultiplyFunction(int x, int y)  {
  int result;
  result = x * y;
  return result;
}
```

Contoh Fungsi 3 : (Fungsi pembacaan sensor)

```
int ReadSens_andCondition()
{
  int I;
  int sval;

for (i = 0; i < 5; i++)
{
  sval = sval  + analogRead(0);          // sensor on analog pin 0
}

  sval = sval / 5;                       // average (rata-rata)
  sval = sval / 4;                       // scale to 8 bits (0 – 255)
  sval = 255 – sval;                     // invert output
  return sval;
}
```

Tabel 3.3 Operator Aritmatika

Operator Aritmatika	Keterangan
+	operasi penjumlahan
-	operasi pengurangan
*	operasi perkalian
/	operasi pembagian
%	operasi sisa pembagian (modulus)

Operator *, / dan % memiliki prioritas yang lebih tinggi dibandingkan dengan operator + dan -.

Contoh : 3 + 4 * 8 = 35 ➔ mengandung arti sama dengan 3 + (4 * 8) = 35

Tabel 3.4. Operator Pembanding

Operator Kondisi	Contoh	Keterangan
==	x == y	bernilai benar bila kedua data sama dan bernilai salah bila sebaliknya
!=	x != y	tidak sama dengan
>	x > y	lebih besar dari
<	x < y	lebih kecil dari
>=	x >= y	lebih besar atau sama dengan
<=	x <= y	lebih kecil atau sama dengan

Operator pembanding digunakan untuk membandingkan dua buah data.

Tabel 3.5. Operator Logika

Operator Logika	Keterangan
&&	Operator untuk logika *AND*
||	Operator untuk logika *OR*
!	Operator untuk logika *NOT*

Operasi logika digunakan untuk membentuk suatu logika atas dua buah kondisi atau lebih.

Tabel 3.6. Operator *Bitwise*

Operator *Bitwise*	Keterangan
&	Operator untuk operasi *AND* level *bit* (*biner*)
|	Operator untuk operasi *OR* level *bit* (*biner*)
^	Operator untuk operasi *XOR* level *bit* (*biner*)
~	Operator untuk operasi *NOT* level *bit* (*biner*)
<<	Operator untuk operasi geser kiri
>>	Operator untuk operasi geser kanan

Operasi *bitwise* hanya bekerja pada operasi logika 0 dan 1 saja.

Contoh :

x = 0x52;
y = 0x49;

Pertanyaan : x & y ➔ x = 0101 0010

y = 0100 1001

x & y = 0100 0000

Sehingga x & y = 0x40

Pertanyaan : (x & y)<<1 ➔ x & y = 0100 000 bila digeser 1 kali ke kiri maka nilai x&y akan menjadi

➔ 1000 0000

Sehingga (x & y)<<1 = 0x80

Tabel 3.7. Operator Tambahan

Asumsi variabel a =10 dan b =20

Nama Operator	Operator Sederhana	Deskripsi	Contoh
Increment	++	Operator increment, menaikkan nilai integer sebanyak satu langkah	a++
Decrement	--	Operator decrement, menurunkan nilai integer sebanyak satu langkah	x--
Compound Addition	+=	Menambahkan operand sisi kanan dengan operand sisi kiri dan hasilnya ke operand sisi kiri	B+= A ekivalen dengan B = B + A
Compound Substraction	-=	Mengurangi operand sisi kanan dengan operand sisi kiri dan hasilnya ke operand sisi kiri	B-=A ekivalen dengan B = B - A

Nama Operator	Operator Sederhana	Deskripsi	Contoh
Compound Addition	*=	Mengali operand sisi kanan dengan operand sisi kiri dan hasilnya ke operand sisi kiri	B*=A ekivalen dengan B = B * A
Compound Substraction	/=	Membagi operand sisi kanan dengan operand sisi kiri dan hasilnya ke operand sisi kiri	B/=A ekivalen dengan B = B / A
Compound Modulo	%=	Menetapkan bentuk modulus menggunakan dua operand dan menetapkan hasilnya ke operand kiri	B %= A ekivalen dengan B = B % A
Compound Bitwise And	&=	Bitwise dan operator penugasan	A &=2 ekivalen dengan A = A & 2
Compound Bitwise Or	|=	Bitwise inclusif OR dan operator penugasan	A |=2 ekivalen dengan A = A | 2

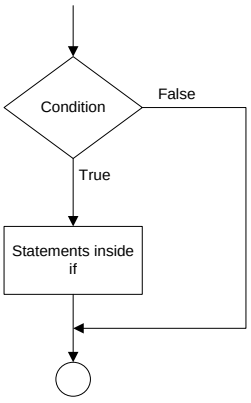
FLOW CONTROL

Bagian ini membicarakan tentang bagaimana menggunakan *syntax* yang sering digunakan untuk melakukan pemilihan atau pengambilan keputusan. Terdapat beberapa *flow control* yang akan dibahas, *yaitu if, if ... else, for, while, dan do ... while.*

PERNYATAAN if

Pernyataan *if* melakukan pengetesan kondisi jika kondisi tersebut telah terpenuhi, seperti *input analog* yang diterima telah berada pada kondisi tertentu. Maka akan mengeksekusi semua *statement* yang berada dalam kurung jika *statement* tersebut *true* , dan akan mengeksekusi semua perintah yang berada dalam kurung { … }. Namun jika tidak terpenuhi akan melewati *statement* tersebut.

Pernyataan *if – execution sequence*



Perbedaan bentuk dari *syntax if*:

Bentuk 1

```
If (expression)
        Statement;
```

Bentuk 2

```
If (expression)
  {
        Block of statements;
  }
```

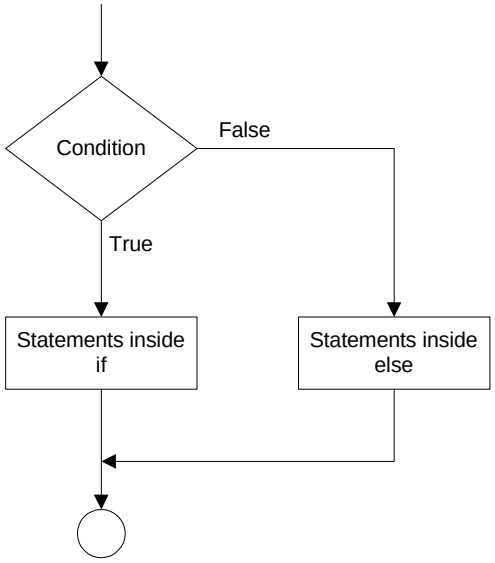
Contoh program :

```
/* global variable definition */
Int A = 5;
Int B = 9;
void setup()
 {
 }
void loop()
  If (expression)
  {
    If(A>B)   /*if condition is true then execute the following statement*/
          A++;
  }
  If ((A>B) && (B!=0))  /* if condition is true then execute the following
                           statement*/
    {
          A+=B;
          B--;
    }
```

PERNYATAAN if … else

Pernyataan *if ... else* memberikan kejelasaan dalam menangani lebih dari satu *statement*. Misalnya jika *input* yang terbaca adalah *High* maka lakukan *action* A, namun jika *Low* lakukan *action* B.

Pernyataan *if – execution sequence*



Bentuk *syntax if .. else*

```
If (expression)
  {
        Block of statements;
  }
else
  {
        Block of statements;
  }
```

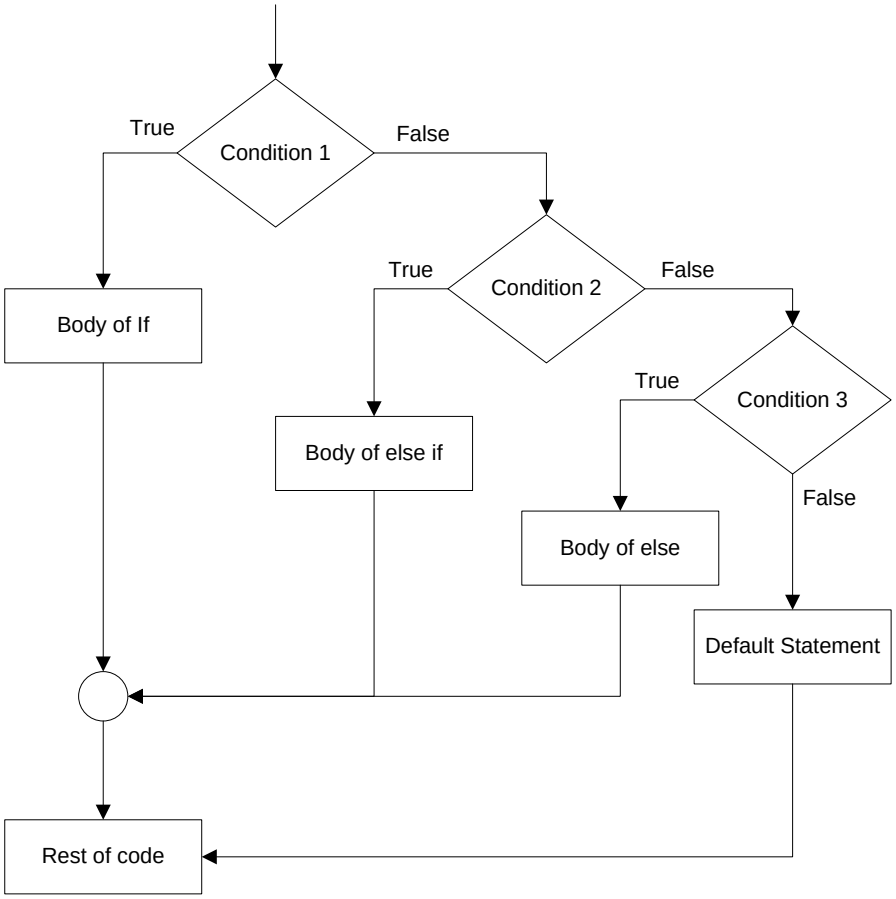
Contoh program :

```
/* global variable definition */
Int A = 5;
Int B = 9;
void setup()
  {
  }
void loop()
{
  If (expression)
  {
    If(A>B)  /*if condition is true then execute the following statement*/
          A++;
  }
  else
    {
          B -= A;
    }
}
```

PERNYATAAN IF BERSARANG

Pernyataan *if* bersarang adalah pernyataan *if* maupun *if..else* dimana di dalam blok pernyataan yang akan dikerjakan terdapat pernyataan *if* atau *if..else* lagi.

Pernyataan *if … else* bersarang – *execution sequence*



Bentuk *syntax if ... else* Bersarang

```
If (expression_1)
  {
        Block of statements;
  }
else If (expression_2)
  {
        Block of statements;
  }
        .
        .
        .
else
  {
        Block of statements;
  }
```

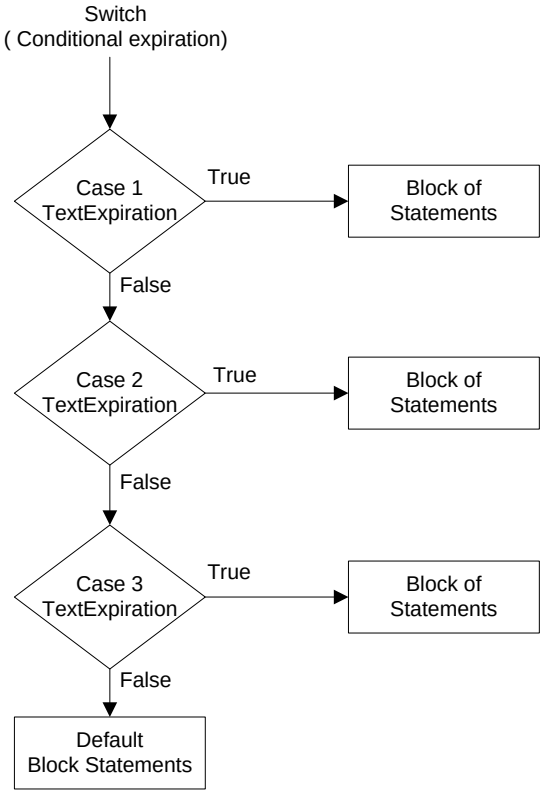
Contoh *code* :

```
/* global variable definition */
Int A = 5;
Int B = 9;
Int C = 15;

void setup()
 {
 }

void loop()
  {
    If(A>B)   /*if condition is true then execute the following statement*/
          {
                A++;
          }
    else if (( A==B) || (B < C))    /*if condition is true then execute the following
                                      statement */
          {
                C = B*A;
          }
    else
    C++;
  }
```

PERNYATAAN SWITCH

Pernyatan *if..else* dapat diganti dengan pernyataan *switch*. Pernyataan *switch* digunakan untuk melakukan pengambilan keputusan terhadap banyak kemungkinan.

Pernyataan *Switch – execution sequence*



Bentuk *Syntax* Pernyataan *Switch*

```
switch (variable)
{
    case label:
   // statements
       break;
    }
  case label:
    {
 // statements
    break;
     }
  default:
    {
     // statements
      break;
```

```
    }
}
```

Contoh code:

```
  switch (phase)
{
  case 0: Lo();  break;
  case 1: Mid(); break;
  case 2: Hi();  break;
  default: Message("Invalid state!");
}
```

Catatan :

- *Switch* hanya dapat memeriksa variabel yang memiliki sebuah konstanta, sedangkan *if* dapat memeriksa persyaratan perbandingan (lebih besar, lebih kecil dan sebagainya).
- Tidak ada konstanta yang sama di dalam sebuah *switch*.
- Perintah *switch* jika dimanfaatkan dengan tepat akan memberikan hasil yang lebih baik bila dibandingkan dengan *perintah if..else*.

LOOPS

Dalam bahasa pemogramman C yang termasuk dalam *loop* adalah sebagai berikut :

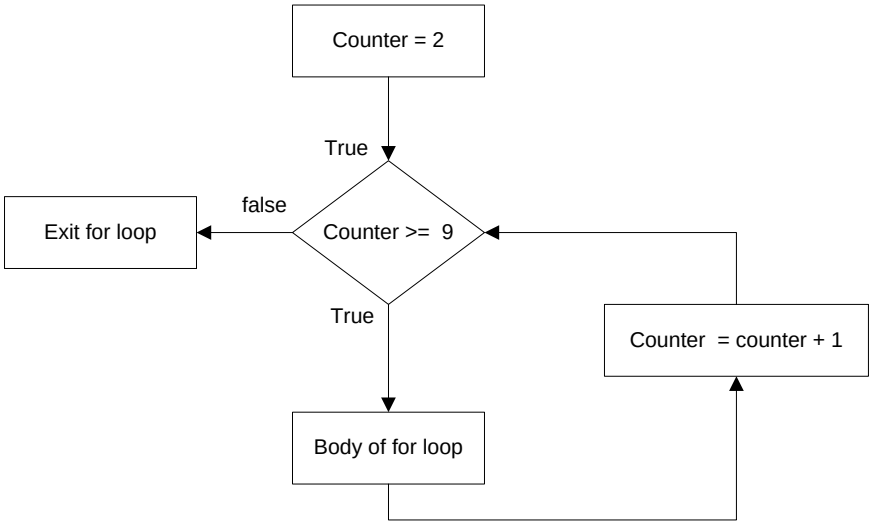
- *For*
- *While*
- *Do .. while*
- *Nested*

For

Pernyataan for digunakan untuk melakukan perulangan yang terdapat pada statement di dalam {}. Untuk bisa melakukan perulangan maka terdapat sebuah counter yang akan menaikkan hitungan secara satu per satu, dan memberikan tanda kapan perulangan akan berakhir. Pernyataan for sangat berguna untuk operasi yang memerlukan perulangan dan sering digunakan bersama dengan

sekelompok *array*, yang biasanya digunakan untuk menyimpan koleksi data atau pin.

Pernyataan for – execution sequence



Bentuk syntax for

```
for ( initialize; control; increment or decrement)
{
// statement block
}
```

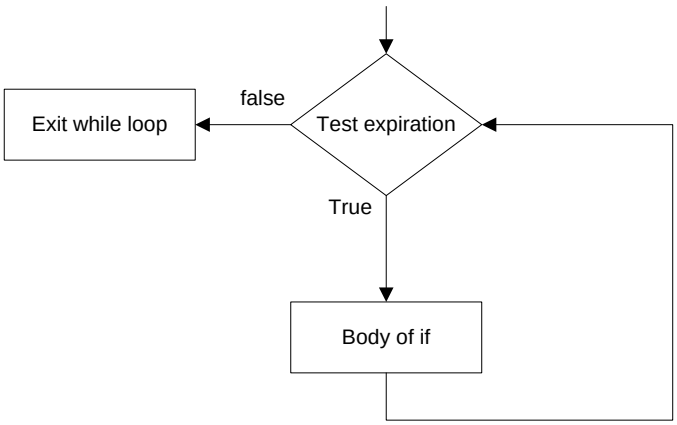
Contoh code :

```
for(counter=2;counter <=9;counter++)
{
   //statements block will executed 10 times
}
```

WHILE

Perulangn menggunakan while akan terus berlangsung terus menerus sampai ekspresi dalam kurung tutup () bernilai false. Agar perulangan bisa berhenti maka harus ada sesuatu yang bisa mengubah variable, atau perulangan while tidak akan pernah berhenti. Sesuatu yang dimaksud bisa berasa dalam kode pemrograman, seperti variable counter menambah satu nilai variable, atau masukan dari luar seperti sensor.

Pernyataan loop execution sequence



Bentuk syntax while

```
while(expression)
  {
    Block of statements;
  }
```

Contoh code :

```
Var = 0;
While (var < 200) {
// do something repetitive 200 times
Var++;
}
```

Do … While

Perulangan do mempunyai cara kerja yang sama dengan perulangan while, dengan catatan pada perulangan do melakukan test kondisi pada bagian terakhir. Karena model pengecekan pada saat terakhir maka perulangan do akan selalu melakukan eksekusi program, setidaknya sekali.

Betuk syntax pernyataan do … while

```
do{
Block of statements;
} while (expression);
```

Contoh code :

```
do
{
delay(50);
x = readSensors();     // check the sensors
}
while (x < 100);      // loops if is less than 100
```

NESTED LOOP

Pemograman bahasa C memyediakan kepada kita untuk dapat menggunakan sebuah loop didalam loop lainnya.

Bentuk syntaxnya :

```
for ( initialize ;control; increment or decrement)
  {
    // statement block
   for ( initialize ;control; increment or decrement)
       {
        // statement block
       }
   }
```

Contoh code :

```
for(counter=0;counter<=9;counter++)
{
   //statements block will executed 10 times
   for(i=0;i<=99;i++)
     {
      //statements block will executed 100 times
     }
}
```

PELAJARAN 4
APLIKASI LED

TUJUAN

- Dapat membuat program sederhana untuk menyalakan led.
- Memahami perangkat trainer yang digunakan, khususnya dalam menghubungkan kabel penghubung dari *arduino* ke *board* trainer.
- Dapat membuat program running led.
- Dapat membuat program aplikasi led displayer.

BLINKING

Perangkat yang dibutuhkan :

- Perangkat Trainer *Arduino*, dengan komponen yang dibutuhkan :
  - 1 buah *Arduino Uno*
  - 8 buah *led*
- *Power Supply*
- *Software Arduino IDE*
- Multimeter
- Laptop
- Kabel penghubung

Persiapan

- *Install* terlebih dahulu *Software Arduino IDE*
- *Install usbasp driver* agar komputer dapat membaca perangkat *usbasp*.
- Memahami perangkat *hardware* yang digunakan

PERCOBAAN LED_1 : Blinking

- Buka *software Arduino IDE* yang sudah di install di laptop.
- Klik menu *File* dan Pilih Examples.
- Dari beberapa examples yang diberikan klik 01. Basics. Lalu pilih Blink.

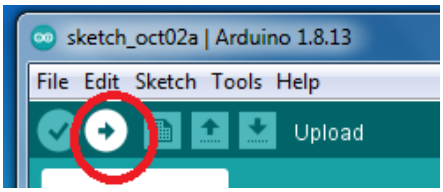
- Hubungkan kabel penghubung dari pin13 *Arduino* ke salah satu pin led yang tersedia di trainer.
- Hubungkan Supply ke perangkat trainer.
- Atur *port* dari *downloader* dengan memilih menu *tools* lalu pilih *port* dan pilih *port* yang tersedia.
- Lakukan verify untuk mengecek apakah program yang dibuat sudah benar atau belum.
- Bila sudah benar program sudah bisa di *Upload* dengan memilih menu upload.
- Komentar dengan tanda double garis miring boleh dihapus untuk membuat program lebih sederhana.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Gambar 4.1. *Listing Program* Percobaan *LED*_1

PERCOBAAN LED_2: RUNNING LED DENGAN 3 BUAH PIN

- Buat *listing program* yang ditunnjukkan pada percobaan *LED*_2.
- Gunaka tiga buah led yang terhubung ke pin 11, 12, 13.
- *Upload* program sebagaimana ditunjukkan gambar 4.2
- Apabila kondisi *led* berkedip secara bergantian maka percobaan yang dilakukan benar. Beri komentar mengapa diperoleh hasil tersebut.



Gambar 4.2. menu apload program

```
void setup() {
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(11, LOW);
  delay(1000);
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  digitalWrite(11, HIGH);
  delay(1000);
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
  delay(1000);
}
```

Gambar 4.3. *Listing Program* Percobaan *LED*_2

RUNNING LED

PERCOBAAN LED_3: RUNNING LED

- Buat *listing program* yang ada pada percobaan *LED*_3.
- Hubungkan *kabel penghubung dari pin arduino ke pin Led* pada trainer.

```
void setup() {
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

}
```

```
void loop() {
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, LOW);
  delay(1000);
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, LOW);
  digitalWrite(6, HIGH);
  delay(1000);
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, LOW);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  delay(1000);
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, LOW);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  delay(1000);
```

```
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(11, HIGH);
  digitalWrite(10, LOW);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  delay(1000);
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(11, LOW);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  delay(1000);

  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  delay(1000);
}
```

Gambar 4.4. *Listing Program* Percobaan *LED*_3

PERCOBAAN LED_4: LED BERGESER DARI KIRI KEKANAN DAN KEMBALI LAGI KE KIRI

- Buat *listing program* yang ada pada percobaan *LED*_4.

```
int ledPin[]= {13,12,11,10,9,8,7,6};
int delayLed = 50;

void setup()
{
 Serial.begin(9600);
 for (int i=0; i<8; i++)
 {
   pinMode(ledPin[i], OUTPUT);
 }
}

void loop()
{
    kananKiri();
    kiriKanan();
}

void kananKiri()
 {
  for (int i=0; i<8; i++)
  {
    digitalWrite(ledPin[i], LOW);
    delay(delayLed);
    digitalWrite(ledPin[i], HIGH);
    delay(delayLed);
  }
 }

 void kiriKanan()
 {
  for (int i=7; i>-1; i--)
  {
    digitalWrite(ledPin[i], LOW);
    delay(delayLed);
    digitalWrite(ledPin[i], HIGH);
    delay(delayLed);
  }
 }
```

Gambar 4.5. *Listing Program* Percobaan *LED*_4

PERCOBAAN LED_5: LED BERGESER DARI TENGAH KE TEPI / LUAR

- Buat *listing program* yang ada pada percobaan *LED*_5.

```
int ledPin[]= {13,12,11,10,9,8,7,6};
int delayLed = 100;

void setup()
{
 Serial.begin(9600);
 for (int i=0; i<8; i++)
 {
   pinMode(ledPin[i], OUTPUT);
 }
}

void loop()
{
   tengahKananKiri();
}

void tengahKananKiri()
{
 int angkaGanjil = 1;
 for (int i=3; i>-1; i--)
 {
   digitalWrite(ledPin[i], LOW);
   digitalWrite(ledPin[i+angkaGanjil],LOW);
   delay(delayLed);
   digitalWrite(ledPin[i], HIGH);
   digitalWrite(ledPin[i+angkaGanjil],HIGH);
   delay(delayLed);
   angkaGanjil += 2;
 }
}
```

Gambar 4.6. *Listing Program* Percobaan *LED*_5

SOAL ESSAY

1. Buatlah program yang mengendalikan led dari tengah ketepi, kemudian lanjut dari tepi ke tengah dan begitu seterusnya ?
2. Buatlah aplikasi lampu displayer sesuai dengan selera anda ?
3. Buat program traffic light 4 simpang, memanfaatkan trainer traffic light yng tersedia ?

PELAJARAN 5
APLIKASI TOMBOL DAN LED

TUJUAN

- Memahami dasar-dasar pemrograman *LED dan Tombol.*
- Mampu membuat aplikasi *input* dan *output* pada *Arduino Uno* menggunakan bahasa pemrograman C.
- Mampu membuat *program tombol dan led.*

PENGGUNAAN PUSH BUTTON

Perangkat yang dibutuhkan :

- Perangkat Trainer *Arduino,* dengan komponen yang dibutuhkan :
  - 8 buah *push button*
  - 8 buah *led*
- *Power Supply*
- *Software Arduino IDE*
- Multimeter
- Laptop
- Kabel penghubung

PERCOBAAN TOMBOL_1 : Led Menyala saat Tombol Ditekan

- Buka *software Arduino IDE*.
- Buat program seperti yang ditunjukkan gambar 5.1.
- Atur tegangan *Power Supply* dengan range 9 sampai 12 Volt *DC*. Selanjutnya hubungkan pada modul praktikum.
- Hubungkan kabel penghubung dari pin13 *Arduino* ke salah satu pin led yang tersedia di trainer.
- Hubungkan kabel penghubung dari pin 2 *arduino* ke salah satu pin tombol.
- Hubungkan Supply yang telah diatur ke perangkat trainer.
- Atur *port* dari *downloader* dengan memilih menu *tools* lalu pilih *port* dan pilih *port* yang tersedia.

- Lakukan verify untuk mengecek apakah program yang dibuat sudah benar atau belum.
- Bila sudah benar program sudah bisa di *Upload* dengan memilih menu upload.

```
const int buttonPin = 2;
const int ledPin =  13;

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) {
    digitalWrite(ledPin, LOW);
  } else {
    digitalWrite(ledPin, HIGH);
  }
}
```

Gambar 5.1. *Listing Program* Percobaan *Tombol*_1

Program yang ditunjukkan gambar 5.1 adalah program untuk menyalakan led dengan ketentuan, bila tombol ditekan maka led akan menyala. Dan bila tombol dilepas led akan padam.

PERCOBAAN TOMBOL_2 : Led Menyala saat Tombol Ditekan

- Buat program seperti yang ditunjukkan gambar 5.2.
- Atur tegangan *Power Supply* dengan range 9 sampai 12 Volt *DC*. Selanjutnya hubungkan pada modul praktikum.

- Hubungkan kabel penghubung dari pin13 *Arduino* ke salah satu pin led yang tersedia di trainer.
- Hubungkan kabel penghubung dari pin 2 *arduino* ke salah satu pin tombol.
- Hubungkan Supply yang telah diatur ke perangkat trainer.
- Lakukan verify untuk mengecek apakah program yang dibuat sudah benar atau belum.
- Bila sudah benar program sudah bisa di *Upload* dengan memilih menu upload.

```
void setup() {
  digitalWrite(2, HIGH);
  pinMode(13, OUTPUT);
  pinMode(2, INPUT);
}

void loop() {
  if (digitalRead(2) == LOW)
  {
    digitalWrite(13, LOW);
    delay(2000);
    digitalWrite(13, HIGH);
    while(digitalRead(2) == LOW)
    {
      delay(50);
    }
  }

}
```

Gambar 5.2. *Listing Program* Percobaan *Tombol_2*

Program yang ditunjukkan gambar 5.2 adalah program untuk menyalakan led dengan ketentuan, bila tombol ditekan maka led akan menyala selama 2 detik. Setelah 2 detik led akan padam walaupun tombol masih ditekan. Untuk menyalakan led dengan melepas tombol lalu lakukan penekan tombol lagi.

PERCOBAAN TOMBOL_3 : Led Menyala Flip-flop saat Tombol Ditekan

- Buat program seperti yang ditunjukkan gambar 5.3.
- Atur tegangan *Power Supply* dengan range 9 sampai 12 Volt *DC*. Selanjutnya hubungkan pada modul praktikum.

```
void setup() {
  digitalWrite(2, HIGH);
  pinMode(13, OUTPUT);
  pinMode(2, INPUT);

  tombol_awal();
}

void loop()
{
  digitalWrite(13,LOW);
  delay(100);
  digitalWrite(13,HIGH);
  delay(100);
}

void tombol_awal()
{
 x:
  if (digitalRead(2) == LOW)
  {
    digitalWrite(13, LOW);
    delay(900);
    digitalWrite(13, HIGH);
    while(digitalRead(2) == LOW)
    {
    }
  }
  else
  {
    goto x;
  }
}
```

Gambar 5.3. *Listing Program* Percobaan *Tombol*_3

PERCOBAAN TOMBOL_4 : Led Menyala Flip-flop saat Tombol Ditekan

- Buat program seperti yang ditunjukkan gambar 5.4.

```
int ledPin[]= {3,4,5,6,7,8,9,10};
int keadaanVariasi = 0;
int delayLed = 50;

void setup()
{
 for (int i=0; i<8; i++)
 {
   pinMode(ledPin[i], OUTPUT);
 }
 attachInterrupt(digitalPinToInterrupt(1), variasi, RISING);
}

void loop() {
  switch(keadaanVariasi)
  {
    case 0:  kananKiri();   break;
    case 1:  kiriKanan();  break;
    default:  break;
  }
}

void variasi()
 {
  delay(100);
  keadaanVariasi++;
  Serial.print("keadaanVariasi=");
  Serial.println(keadaanVariasi);
  if(keadaanVariasi>1) keadaanVariasi =0;
 }

  void kananKiri()
 {
  for (int i=0; i<8; i++)
  {
    digitalWrite(ledPin[i], HIGH);
    delay(delayLed);
    digitalWrite(ledPin[i], LOW);
    delay(delayLed);
    if(keadaanVariasi != 0) break;
  }
 }
```

Gambar 5.4. *Listing Program* Percobaan *Tombol_4*

Program yang ditunjukkan gambar 5.4 akan bekerja sebagai berikut. Bila program ini dijalankan, awalnya led agar running dari kiri ke kanan dan bila tombol di operasikan led akan running dari kanan kekiri dan begitu seterunya.

SOAL ESSAY

1. Tambahkan program running led dengan 3 buah led sebagaimana ditunjukkan percobaan pada gambar 4.2 dengan menggunakan tombol. Ketentuannya adalah running led akan bekerja bila tombol telah ditekan.
2. Lakukan hal yang sama pada percobaan led gambar 4.3. Led bisa bekerja bila tombol ditekan dan led akan pada bila tombol yang sama ditekan ?
3. Buatlah program yang mengendalikan led dari tengah ketepi, kemudian lanjut dari tepi ke tengah dan begitu seterusnya ?
4. Buatlah aplikasi lampu displayer sesuai dengan selera anda ?
5. Buat program traffic light 4 simpang, dengan penekanan tombol terlebih dahulu untuk mengaktifkannya ?
6. Buat program 3 buah led yang bekerja secara mandiri. Led 1 flip-flop setiap 1 detik, led 2 flif-flop setiap 3 detik dan led 3 flip-flop setiap 5 detik ?

PELAJARAN 6
APLIKASI *LCD*

TUJUAN

- Memahami dasar-dasar pemrograman *LCD*
- Mampu memprogram mikrokontroler *Arduino* untuk menampilkan karakter menggunakan bahasa pemrograman C.
- Dapat membuat rancangan skematik *LCD* menggunakan mikrokontroler *Arduino*

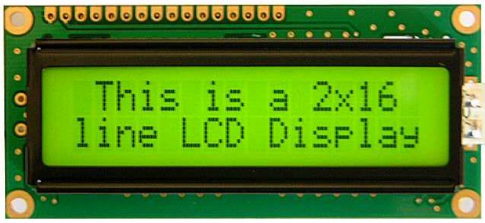
PENGANTAR *LCD*

Perangkat yang dibutuhkan :

- Perangkat Trainer *Arduino*, dengan komponen yang dibutuhkan
  - 1 buah *Arduino Uno*
  - 4 buah *push button*
  - 8 buah *led*
  - 1 buah *LCD* 16 x 2
  - Kabel penghubung secukupnya
- *Power Supply*
- Multimeter
- Laptop
- Kabel penghubung

Perangkat *I/O* berikutnya yang akan dipraktekkan adalah *LCD*. *LCD* berguna sekali untuk membantu menampilkan hasill perhitungan, isi variabel atau keperluan lainnya yang dibutuhkan dalam pemrograman. *LCD* berisi bahan cairan kristal yang pengoperasian-nya menggunakan *dot matrik*.

Bentuk fisik *LCD* yang digunakan ditunjukkan sesuai gambar 6.1 berikut ini :

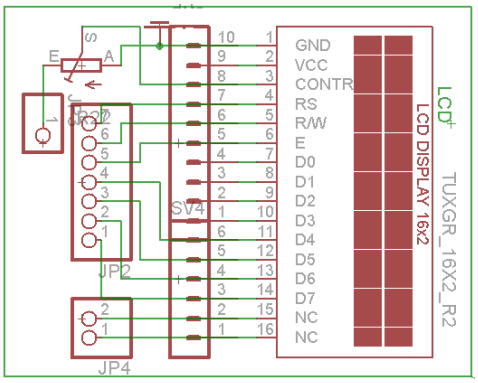


Gambar 6.1. Bentuk Fisik *LCD*

Berikut ditunjukkan konfigurasi pin *LCD*, selain itu juga ditunjukkan rangkaian *interface LCD*.

Tabel 6.1. Konfigurasi Pin *LCD*

PIN	NAMA PIN	FUNGSI
1	VSS	Ground
2	VCC	Catu Daya + 5V
3	VEE	Contrast
4	RS	Register Select 0 = Instruction Register 1 = Data Register
5	R/W	Read/Write, to choose write or read mode 0 = write mode 1 = read mode
6	E	Enable 0 = start to latch data to *LCD* character 1 = disable
7	DB0	Data bit ke 0 (LSB)
8	DB1	Data bit ke 1
9	DB2	Data bit ke 2
10	DB3	Data bit ke 3
11	DB4	Data bit ke 4
12	DB5	Data bit ke 5
13	DB6	Data bit ke 6
14	DB7	Data bit ke 7
15	BPL	Positif Backlight Voltage (4 - 4,2V; 50-200 mA)
16	GND	Negatif Backlight Voltage (0 Volt)



Gambar 6.2. Rangkaian *Interfacing LCD* pada Trainer Mikrokontroler

Untuk memfungsikan *LCD* dapat dilakukan dengan dua cara, yaitu menggunakan metode hubungan paralel dan menggunakan perangkat tambahan yang bernama *I2C*. Kali ini kita akan mempraktekkan aplikasi *LCD* dengan menggunakan perangkat *I2C*. *Arduino Uno* yang kita gunakan sudah mendukung komunikasi *I2C* dengan modul I2C *Lcd*. Untuk memfungsikan *Lcd* dibutuhkan 2 pin *Analog Input*, yaitu Pin 4 (*SDA*) dan Pin 5 (*SCL*).

Modul *I2C* memiliki 4 pin, 2 pin untuk power dan 2 pin untuk komunikasi *I2C*, untuk mengatur kontras disediakan potensiometer pada modul *I2C*.

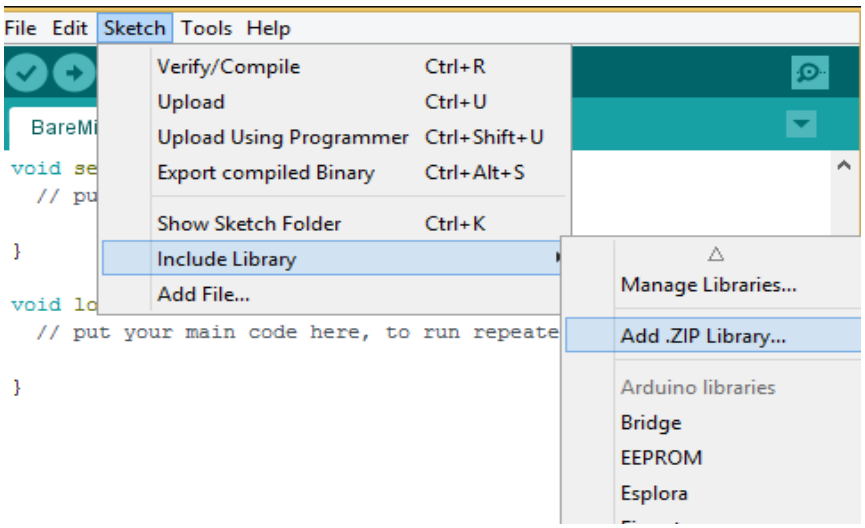
Tabel 6.2. Alokasi Pin *Arduino* :

LCD I2C	***ARDUINO***
SCL	Pin A5
SDA	Pin A4
VCC	+ 5 V
GND	Gnd

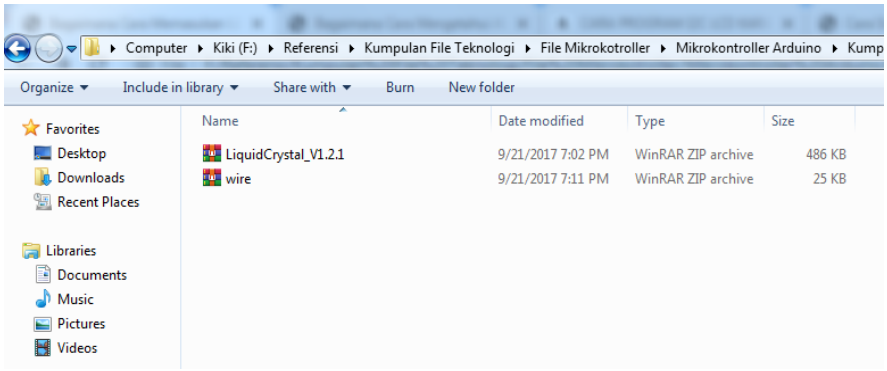
Sebelum membuat program pastikan terlebih dahulu, *library Lcd I2C* sudah didownload. Ada 2 *file* yang harus di download, yaitu *library Lcd I2C* dan *library wire* untuk komunikasi *I2C*.

Setelah 2 file tersebut di download, masukkan library tersebut ke *software Arduino IDE* yaitu dengan cara sebagai berikut :

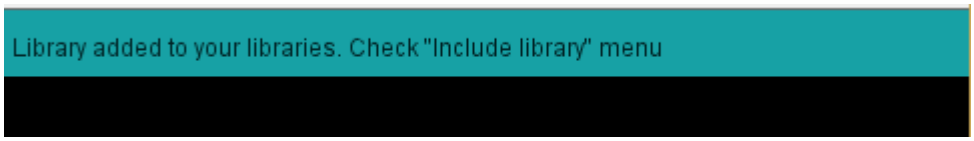
- Buka aplikasi *Arduino*, lalu masuk ke menú ***Sketch***, pilih ***Include Library***, pilih ***ADD.ZIP Library…***



- Cari file Library yang sudah di download,lalu ***Open***



- Jika berhasil, aplikasi *Arduino* akan muncul keterangan seperti berikut ini



PERCOBAAN LCD_1 : TAMPILAN LCD

- Buat *program LCD* dengan memanfaatkan *Sketch Arduino IDE*.
- Lakukan *Verify* untuk memastikan *code* yang dibuat benar.
- Bila tidak ada yang *error*, selanjutnya lakukan *Upload* untuk mengisi *code* yang telah dibuat ke mikrokontroler *Arduino*.

- Bila setelah di *Upload* tidak ada respons pada *Lcd*, maka ada beberapa kemungkinan :
  - 2 *file library* belum dimasukkan ke *software Arduino IDE*
  - Kesalahan pada alamat *Lcd* yaitu pada code :

```
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
```

    Ganti 0x27 dengan 0x3F
  - Lakukan cek Alamat pada *I2C Lcd* dengan menggunakan program cek alamat I2C.
- Amati tampilan pada *Lcd* dan analisa *code* programnya.

```
#include  <Wire.h>
#include  <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
void setup()
{
  Serial.begin(9600);
  lcd.begin(16,2);
  lcd.backlight();  // layar lcd terang
  lcd.setCursor(0,0);
  lcd.print(" Bismillah..... ");
  delay(2500);

  lcd.noBacklight();  // layar gelap
  delay(2500);


  lcd.setCursor(0,1);              // tulisan belum tanpak
  lcd.print(" Alhamdulillah ");  // karena belum di backlight()
  delay(2500);

  // tulisan Alhamdulillah baru timbul
  lcd.backlight();
}

void loop() {
}
```

Gambar 6.3. Bentuk Fisik *LCD*

Keterangan *program* :

- Posisi *LCD* untuk jenis : 16 x 2 → 16 kolom x 2 baris
- ***x,y*** → kolom, baris
- 0, 1 → karakter ditulis dari kolom ke 0 baris ke 1

Instruksinya → *lcd*.setCursor(0, 1);

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

- *lcd*.print(“ Bismillah …… ”); → berfungsi untuk menampilkan karakter dengan tulisan Bismillah ……

PERCOBAAN LCD_2 : PENERAPAN OUTPUT LCD + PUSH BUTTON

- Buat *listing program* yang ada pada gambar percobaan *LCD_2*

```
#include  <Wire.h>
#include  <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
int kondisi = 0;

void setup()
{
  pinMode(13, OUTPUT);
  pinMode(2, INPUT);

  lcd.begin(16,2);
  lcd.setCursor(5,0);
  lcd.print("Bismillah..... ");
  delay(5000);
}
```

```
void loop()
{
  lcd.clear();
  kondisi = digitalRead(2);
  if (kondisi == HIGH)
  {
    lcd.setCursor(2,0);
    lcd.print("Led Menyala");
    digitalWrite(13, LOW);
    delay(1000);
  }
  else
  {
    digitalWrite(13, HIGH);
    lcd.setCursor(2,0);
    lcd.print("Led Padam ");
    delay(1000);
  }
}
```

Gambar 6.4. *Listing Program* Percobaan *LCD_2*

Pada percobaan *LCD*_2 akan diperoleh hasil :

Setelah program di *upload* akan tampil tulisan Bismillah yang menyala kurang lebih 5 detik. Tampilan *lcd* berganti tulisan “Led Menyala” yang ditandai juga dengan nyalanya led dari pin 13. Bila tombol di pin 2 ditekan maka led akan padam, display *lcd* akan menampilkan tulisan “Led Padam”.

PERCOBAAN LCD_3 : Counter

- Buat *listing program* yang ada pada gambar percobaan *LCD*_3

```
#include  <Wire.h>
#include  <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
int  b = 0;
void setup()
{
  lcd.begin(16,2);
  pinMode(3,INPUT);
}

void loop()
{
  if(digitalRead(3) == LOW)
  {
    lcd.setCursor(3,0);
    lcd.print(b++);
    while(digitalRead(3) == LOW);
     {
     }
     delay(500);    // anti bounching
    }
}
```

Gambar 6.5. *Listing Program* Percobaan *LCD_3*

Dalam percobaan *LCD* 3, membuat tampilan di *lcd* melakukan hitungan dari angka ‘1’, ‘2’ dan seterusnya dengan disertai penekanan tombol.

PERCOBAAN LCD_4 : TAMPILAN KARAKTER LCD BERJALAN

- Buat *listing program* yang ada pada gambar percobaan *LCD_4*.
- Atur *delay* sesuai dengan yang diinginkan untuk mendapatkan hasil yang optimum.
- Beri komentar dari hasil praktik yang telah dilakukan.

```
#include  <Wire.h>
#include  <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
int i, j;

void setup()
{
  delay(2000);
  lcd.begin(16,2);
  lcd.setCursor(2,0);
  lcd.print("Bismillah..... ");
  delay(1000);
  lcd.clear();
}

void loop()
{
  lcd.setCursor(2, 0);
  lcd.print("Minal   ");
  delay(1000);
  lcd.setCursor(2, 0);
  lcd.print("Minal Aidin  ");
  delay(1000);
  lcd.setCursor(3, 1);
  lcd.print("Wal ");
  delay(1000);
  lcd.setCursor(3, 1);
  lcd.print("Wal Faizin  ");
  delay(1000);
  lcd.clear();

  for (i = 15 ; i >= 3; i--) {
      lcd.setCursor(i,0);
      lcd.print("MOHON MAAF");
      delay(400);
      lcd.clear();  // untuk menghapus semua karakter
  }
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print("MOHON MAAF");
  delay(1000);
```

```
for (i = 15 ; i >= 2; i--) {
      lcd.setCursor(i,1);
      lcd.print("LAHIR & BATIN");
      delay(400);
      lcd.clear();
      lcd.setCursor(3, 0);
      lcd.print("MOHON MAAF");
  }

lcd.clear();
lcd.setCursor(3, 0);
lcd.print("MOHON MAAF");
lcd.setCursor(2, 1);
lcd.print("LAHIR & BATIN");
delay(1000);

for (i = 3 ; i < 16; i++) {
    lcd.setCursor(i,0);
    lcd.print("MOHON MAAF");
    delay(250);
    lcd.clear();  // untuk menghapus semua karakter
    lcd.setCursor(2, 1);
    lcd.print("LAHIR & BATIN");
}
for (i = 2 ; i < 16; i++) {
    lcd.setCursor(i,1);
    lcd.print("LAHIR & BATIN");
    delay(250);
    lcd.clear();  // untuk menghapus semua karakter
}

 delay(700);
 lcd.setCursor(5,0);
 lcd.print("4");
 delay(400);
 lcd.clear();
 lcd.setCursor(5,0);
 lcd.print("4 J");
 delay(400);
```

```
 lcd.clear();
 lcd.setCursor(5,0);
lcd.print("4 JU");
delay(400);
 lcd.clear();
 lcd.setCursor(5,0);
lcd.print("4 JUN");
delay(400);
 lcd.clear();
 lcd.setCursor(5,0);
lcd.print("4 JUNI");
delay(400);


lcd.setCursor(5,0);
lcd.print("4 JUNI");
lcd.setCursor(3,1);
lcd.print("T");
delay(400);
lcd.clear();
lcd.setCursor(5,0);
lcd.print("4 JUNI");
lcd.setCursor(3,1);
lcd.print("TA");
delay(400);
lcd.clear();
lcd.setCursor(5,0);
lcd.print("4 JUNI");
lcd.setCursor(3,1);
lcd.print("TAH");
delay(400);
lcd.clear();
lcd.setCursor(5,0);
lcd.print("4 JUNI");
lcd.setCursor(3,1);
lcd.print("TAHU");
delay(400);
lcd.clear();
lcd.setCursor(5,0);
lcd.print("4 JUNI");
```

```
  lcd.setCursor(3,1);
  lcd.print("TAHUN");
  delay(400);
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("4 JUNI");
  lcd.setCursor(3,1);
  lcd.print("TAHUN 2");
  delay(400);

  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("4 JUNI");
  lcd.setCursor(3,1);
  lcd.print("TAHUN 20");
  delay(400);
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("4 JUNI");
  lcd.setCursor(3,1);
  lcd.print("TAHUN 202");
  delay(400);
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("4 JUNI");
  lcd.setCursor(3,1);
  lcd.print("TAHUN 2020");
  delay(400);

  i=2;
  while(i=2)
     {};
}
```

Gambar 6.6. *Listing Program* Percobaan *LCD*_4

SOAL ESSAY

1. Buatlah Rangkaian skematik *LCD* menggunakan *I2C* dan rangkaian skematik *lcd* hubungan paralel ?
2. Buat program menggunakan hubungan paralel dengan aplikasi yang telah saudara kuasai ?

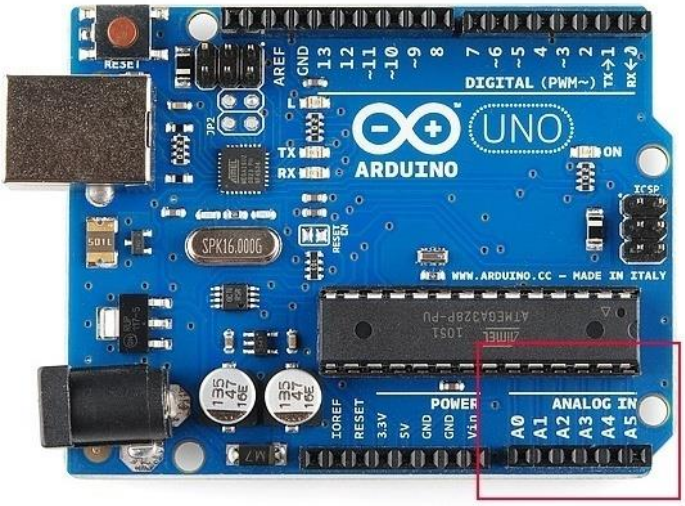
PELAJARAN 7
APLIKASI *ADC*

TUJUAN

- Dapat menjelaskan fungsi *ADC* pada mikrokontroler.
- Dapat membuat *program ADC* menggunakan mikrokontroler *Arduino Uno*.
- Dapat merancang suatu aplikasi yang mengkonversi data *analog* dari suatu sensor ke *display* data *digital*

ADC (*ANALOG TO DIGITAL CONVERTER*)

Analog to Digital Conversion (ADC) adalah fitur yang sangat berguna untuk mengubah tegangan *analog* pada pin ke nomor digital. Dengan mengkonversi dari data *analog* ke data digital, kita dapat menggunakan elektronik untuk berinteraksi dengan data *analog* di sekitar kita.



Gambar 7.1. Posisi pin *ADC* pada *Arduino Uno*

Arduino dapat membaca Tegangan *analog* melalui salah satu pin *analog input* (A0 – A5). Pin *analog input* ini, terhubung dengan *ADC* yang terdapat didalam mikrokontroler *AVR*, dan memiliki resolusi 10 bit (rangenya dari 0 – 1023). Setiap tegangan *analog* yang masuk melalui pin ini akan dikonversikan ke suatu nilai digital yang melalui rumus :

$$\text{Nilai digital} = \frac{V_{masuk}}{V_{ref}}\ x\ 1023$$

Pada *arduino* Vref = 5 V (jika tidak diberi tegangan referensi luar). Dari rumusan yang telah diberi maka dapat diketahui resolusi per bitnya adalah 5 V / 1024 = 0,0049 Volt (4,9 mV). Pada papan berbasis *ATmega* (*Uno, Nano, Mini, Mega*), dibutuhkan sekitar 100 mikro detik (0,0001 detik) untuk membaca *input analog*, sehingga kecepatan membaca maksimum sekitar 10.000 kali per detik.

BOARD	OPERATING VOLTAGE	USABLE PINS	MAX RESOLUTION
Uno	5 Volts	A0 to A5	10 bits
Mini, Nano	5 Volts	A0 to A7	10 bits
Mega, Mega2560, MegaADK	5 Volts	A0 to A14	10 bits
Micro	5 Volts	A0 to A11*	10 bits
Leonardo	5 Volts	A0 to A11*	10 bits
Zero	3.3 Volts	A0 to A5	12 bits**
Due	3.3 Volts	A0 to A11	12 bits**
MKR Family boards	3.3 Volts	A0 to A6	12 bits**

Contoh :

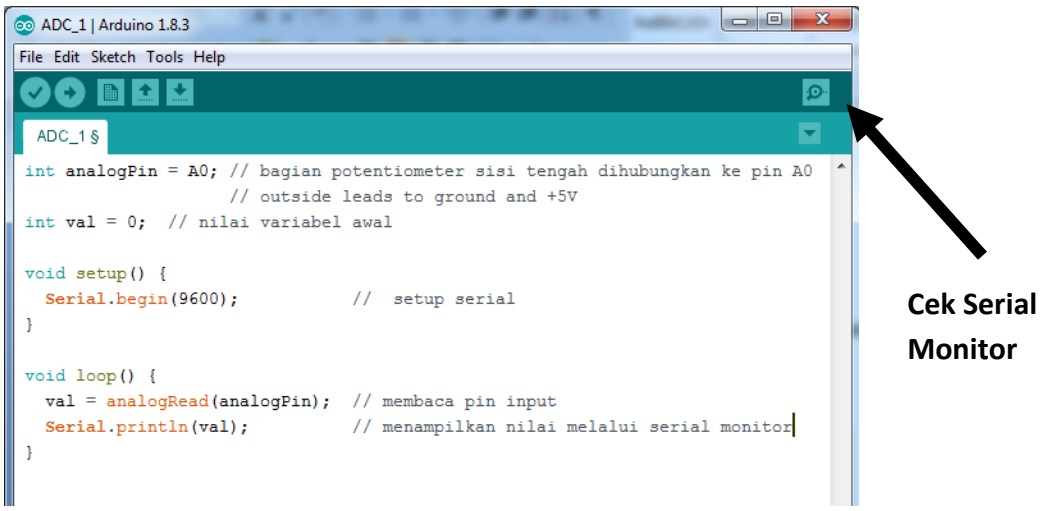
Jika tegangan masuk (V_{in}) 3 V maka akan terbaca pada *arduino* sebagai nilai digital = $\frac{3}{5}\ x\ 1023\ =\ 614$ (dibulatkan).

Instruksi yang diberikan untuk membaca tegangan *analog* adalah :

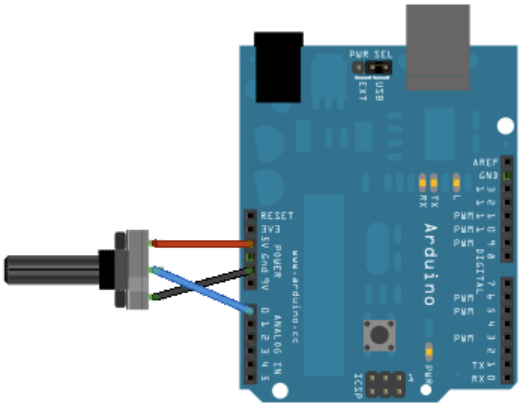
```
analogRead(pinAnalogInput);
```

PERCOBAAN ADC_1 : PROGRAM DASAR PENGIRIMAN DATA ADC

- Buat *program ADC_1* seperti yang ditunjukkan gambar 7.4, menggunakan *software Arduino IDE*.
- Hubungkan kabel dari pin A0 *Arduino* ke pin *potensiometer* sebagaimana ditunjukkan gambar 7.3
- Nyalakan Power Supply dengan tegangan 7 – 12 V ke *Arduino*.
- Lakukan *Upload* program yang telah dibuat.
- Jika program telah sukses (tidak ada yang *error*) selanjutnya cek perubahan data dengan mengatur *potensiometer*. Perubahan data dapat dilihat pada serial monitor yaitu dengan cara mengkliknya yang ditunjukkan gambar 7.2.
- Amati hasil yang diperoleh dari serial monitor.



Gambar 7.2. Cek hasil dari serial monitor

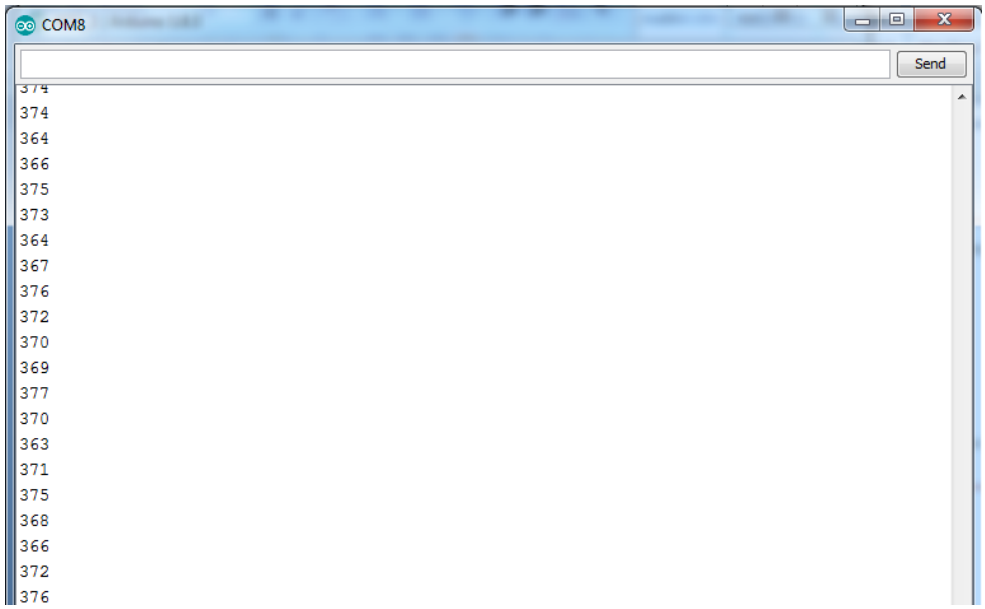


Gambar 7.3. Skematik Rangkaian *ADC*

```
int analogPin = A0; // bagian potentiometer sisi tengah dihubungkan ke pin A0
                    // sisi lainnya pada ground dan +5V
int val = 0;  // nilai variabel awal

void setup() {
  Serial.begin(9600);           //  setup serial
}

void loop() {
  val = analogRead(analogPin);  // membaca pin input
  Serial.println(val);          // menampilkan nilai melalui serial monitor
}
```

Gambar 7.4. Program Aplikasi *analog*

Hasil eksekusi program yang dapat dilihat melalui serial monitor ditunukkan sebagai berikut.



Gambar 7.5. Hasil Eksekusi Program

PERCOBAAN ADC_2 : KONVERSI NILAI ADC KE NILAI TEGANGAN

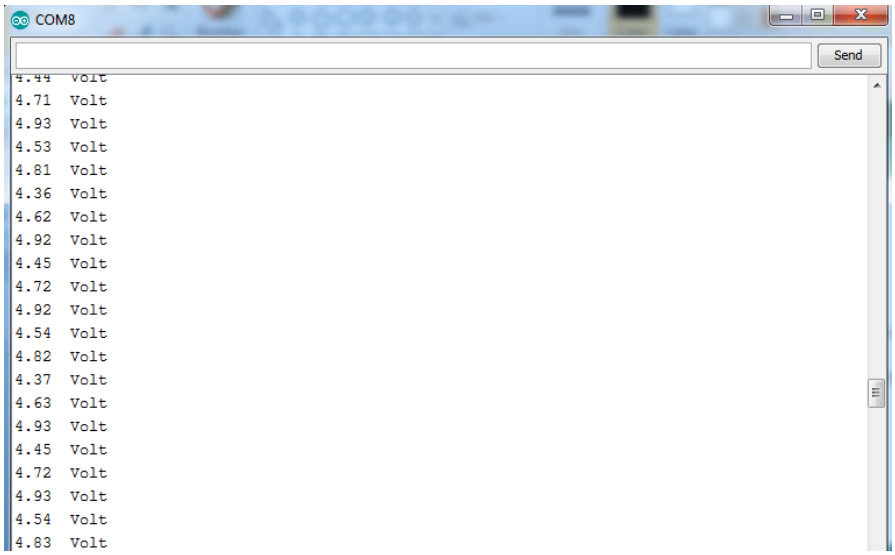
- Buat *program ADC_2* seperti yang ditunjukkan gambar 7.6, menggunakan *software Arduino IDE*.
- Hubungkan kabel dari pin A0 *Arduino* ke pin *potensiometer*.
- Nyalakan *Power Supply* dengan tegangan 7 – 12 V ke *Arduino*.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A1);
  // Convert the analog reading (which goes from 0 - 1023)
  // to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  Serial.print(voltage);
  Serial.println(" Volt");
}
```

Gambar 7.6. Program Konversi Nilai *ADC* ke Tegangan

- Lakukan *Upload* program yang telah dibuat.
- Cek perubahan data dengan mengatur *potensiometer*.

Hasil eksekusi program yang dapat dilihat melalui serial monitor ditunukkan sebagai berikut.



Gambar 7.7. Hasil Eksekusi Program

PERCOBAAN ADC_3 : MEMBERI NILAI ADC PADA LED

- Buat *program ADC_3* seperti yang ditunjukkan gambar 7.8, *menggunakan software Arduino IDE*.
- Hubungkan kabel dari pin A1 *Arduino* ke pin *potensiometer*.
- Nyalakan *Power Supply* dengan tegangan 7 – 12 V ke *Arduino*.
- Lakukan *Upload* program yang telah dibuat.
- Cek perubahan data dengan mengatur *potensiometer*.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
int sensorPin = A1;
int ledPin = 13;
int sensorValue = 0;
```

```
void setup() {
  Serial.begin(9600);
  lcd.begin(16,2);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  lcd.clear();
  sensorValue = analogRead(sensorPin);
  lcd.setCursor(1,0);
  lcd.print("Nilai Delay =");
  lcd.setCursor(5,1);
  lcd.print(sensorValue);
  delay(1000);
  Serial.print("Nilai sensorValue = nilai delay = ");
  Serial.println(sensorValue);

  digitalWrite(ledPin, HIGH);
  delay(sensorValue);
  digitalWrite(ledPin, LOW);
  delay(sensorValue);

}
```

Gambar 7.8. Program Konversi Nilai *ADC* ke Tegangan

Hasil dari eksekusi program dapat dilihat pada layar serial monitor dan atau dapat dilihat pada display *LCD*. Perubahan nilai menyebabkan perubahan delay, ditandai dengan perubahan kedipan led.

SOAL ESSAY

1. Berapa jumlah *ADC* yang dapat digunakan pada Mikrokontroller *Arduino* Uno ?
2. Berapa resolusi *input ADC* dan berapa jangkauan variasi nilainya ?
3. Jika ref *ADC* menggunakan 5 V *DC*. Nilai *ADC* adalah 500, dari nilai *ADC* tersebut tentukan tegangan pada *sensor* yang masuk ke *ADC* ?
4. Buat program ketika nilai potensiometer yang diatur 250 – 350 maka led akan menyala, selain itu led akan padam ?

5. Gunakan 3 buat led untuk menjalankan program, dengan ketentuan :
   - Nilai potensiometer 0 - 100 → semua led akan padam
   - Nilai potensiometer 101 - 200 → led 1 On
   - Nilai potensiometer 201 - 300 → led 2 On
   - Nilai potensiometer 301 - 400 → led 1 & 2 On
   - Nilai potensiometer 401 - 500 → led 3 On
   - Nilai potensiometer 501 - 600 → led 1 & 3 On
   - Nilai potensiometer > 600 → semua led akan On

   Tampilkan di *lcd* dan serial monitor untuk semua keadaan

6. Buat program yang menampilkan tegangan pada solar cell, Tegangan ditampilkan melalui *LCD* dan serial monitor ?

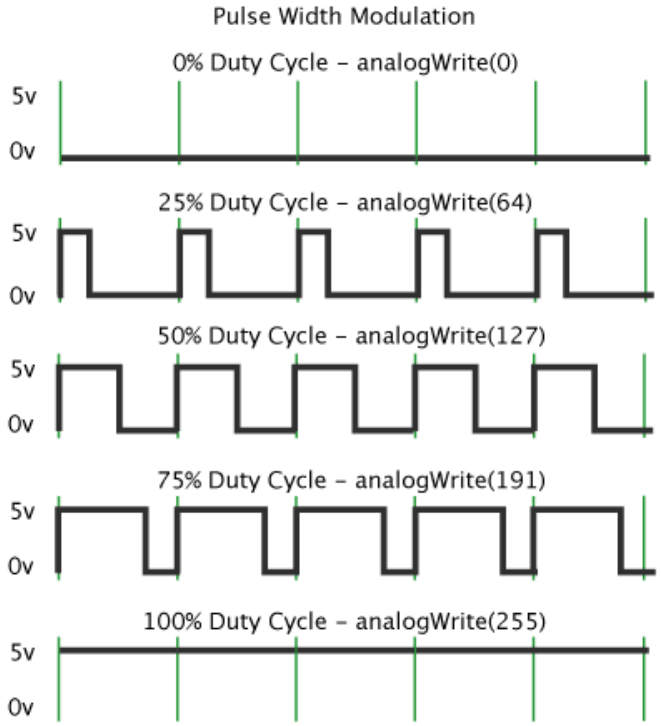
PELAJARAN 8
APLIKASI *PWM*

TUJUAN

- Dapat memahami dasar-dasar pemrograman *PWM*.
- Mampu menjelaskan kegunaan dari pembangkit *PWM*.
- Dapat memprogram *PWM* berbasis *Arduino*

PENGANTAR SINYAL *PWM*

Untuk dapat membangkitkan sebuah tegangan *analog* dari sebuah nilai digital diantarannya adalah menggunakan metoda *Pulse Width Modulation* (*PWM*). Dengan membangkitkan gelombang kotak dan pengaturan *duty cycle* tertentu akan menghasilkan berbagai nilai rata-rata. Teknik *PWM* dilakukan dengan mengontrol gelombang digital menjadi *On/Off* secara bergantian dalam satu perioda gelombang. Perbandingan waktu antara saat gelombang *On* dengan waktu gelombang dalam satu perioda disebut dengan *duty cycle*. Berikut ini salah satu contoh dalam membangkitkan sinyal *PWM*.



Gambar 8.1. Siklus Pembangkitan Sinyal *PWM*

Dalam grafik yang ditunjukkan gambar 8.1, terlihat ilustrasi bagaimana *duty cycle* diubah-ubah tiap gelombang. Pada *Arduino Uno* memiliki jangkauan nilai *PWM* dari 0 – 255, berarti memiliki resolusi 8 bit. *Syntax analogWrite(0)* akan menghasilkan *duty cycle* 0%, sementara *analogWrite*(255) akan menghasilkan *duty cycle* 100 %. Rentang *PWM* 0 hingga 255 akan menghasilkan *duty cycle* dari 0 % sampai dengan 100 %.

Pada dasarnya konsep *PWM* dapat disimulasikan pada semua pin digital. Namun khusus penggunaan fungsi digitalWrite() hanya bisa menggunakan pin-pin khusus (pin *PWM*). Pada *Arduino Uno*, pin yang dapat digunakan untuk untuk *PWM* adalah pin 3, 5, 6, 9, 10, dan 11. Biasanya pin *PWM* disimbolkan dengan karakter '~'.

PERCOBAAN PWM_1 :* MENGUBAH KECERAHAN *LED* MENGGUNAKAN *PWM

- Buat *program PWM_1* sebagaimana yang ditunjukkan gambar 8.2.
- Hubungkan kabel penghubung dari pin 6 ke *led*.
- Nyalakan *Power Supply* dengan tegangan 7 – 12 V ke *Arduino*.
- Pastikan *GND Arduino* sudah dihubungakan dengan *GND* Trainer.
- Lakukan *Upload* program yang telah dibuat.

```
//Initializing LED Pin
int led_pin = 6;
void setup() {
  //Declaring LED pin as output
  pinMode(led_pin, OUTPUT);
}
void loop() {
  //Fading the LED
  for(int i=0; i<255; i++){
    analogWrite(led_pin, i);
    delay(5);
  }
  for(int i=255; i>0; i--){
    analogWrite(led_pin, i);
    delay(5);
  }
}
```

Gambar 8.2. Program *PWM*_1

Program pada gambar 8.2 dapat juga diganti dengan program yang ditunjukkan gambar 8.3. Kedua program ini memiliki fungsi yang sama, yaitu membuat led redup dan terang dalam rentang waktu yang telah ditentukan.

```
int led = 6;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

void setup() {
  pinMode(led, OUTPUT);
}

 void loop() {
   // set the brightness of pin 9:
   analogWrite(led, brightness);

   brightness = brightness + fadeAmount;

   if (brightness <= 0 || brightness >= 255) {
     fadeAmount = -fadeAmount;
   }
   delay(30);
 }
```

Gambar 8.3. Program *PWM*_1

Dari hasil uji coba yang telah dilakukan, jelaskan maksud dari instruksi masing masing coding yang diberikan.

PERCOBAAN PWM_2 : KALIBRASI INPUT ADC

- Buat *program PWM_2* sebagaimana ditunjukkan gambar 8.4.
- Hubungkan Pin A0 ke *output potensiometer* dan pin 9 ke *led*.
- Beri komentar dari hasil eksekusi *program* yang telah dibuat.

```
const int sensorPin = A0;// pin that the sensor is attached to
const int ledPin = 9;     // pin that the LED is attached to
int sensorValue = 0;      // the sensor value
int sensorMin = 1023;     // minimum sensor value
int sensorMax = 0;        // maximum sensor value

void setup() {
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);

  // calibrate during the first five seconds
  while (millis() < 5000) {
    sensorValue = analogRead(sensorPin);

    // record the maximum sensor value
    if (sensorValue > sensorMax) {
      sensorMax = sensorValue;
    }

    // record the minimum sensor value
    if (sensorValue < sensorMin) {
      sensorMin = sensorValue;
    }
  }

  digitalWrite(13, LOW);
}

 void loop() {
   sensorValue = analogRead(sensorPin);
   sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);
   sensorValue = constrain(sensorValue, 0, 255);
   analogWrite(ledPin, sensorValue);
 }
```

Gambar 8.4. *Listing Program* Percobaan *PWM_2*

PERCOBAAN PWM_3 : PENGATURAN TEGANGAN

- Buat *program PWM_3* sebagaimana ditunjukkan gambar 8.5.
- Beri komentar dari hasil eksekusi *program* yang telah dibuat.

```
#define pinSensorTegangan     0
#define pinOutputPWM          9
#define setTegangan           2.5//volt
#define faktorProporsional    0.1

float keluaran;

void setup() {
  pinMode(pinOutputPWM, OUTPUT);

  Serial.begin(9600);
  keluaran = setTegangan;
}

void loop() {
  uint16_t adc = analogRead(pinSensorTegangan);
  float tegangan = map(adc, 0, 1024, 0, 500)/100.0;
  float selisih = setTegangan - tegangan;
  float proporsional = faktorProporsional * selisih;

  keluaran += proporsional;
  keluaran = constrain(keluaran, 0, 5);
  byte keluaranPWM = map(keluaran*100, 0, 5*100, 0, 255);
  analogWrite(pinOutputPWM, keluaranPWM);

  //Plot serial, hapus untuk menambah kecepatan
  Serial.print(tegangan);
  Serial.print(" Volt");
  Serial.print(",       ");
  Serial.print(keluaranPWM);
  Serial.println();
  delay(10);
}
```

Gambar 8.5. *Listing Program* Percobaan *PWM*_3

SOAL ESSAY

1. Buatlah program pengaturan tegangan DC, dimana nilai pengaturannya potensiometer akan mempengaruhi intensitas cahaya led (bila nilai potensiometer minimum membuat led padam dan bila nilai potensiometer maksimum membuat led menyala terang). ?
2. Buat program untuk mengasilkan gelombang kotak dan gelombang dapat di osiloskop?
3. Buat program untuk menghasilkan gelombang segitiga dan gelombang dapat di osiloskop ?
4. Buat program untuk menghasilkan gelombang sinusoida dan gelombang dapat di osiloskop?

PELAJARAN 9

APLIKASI SENSOR IR E18-D80NK/ LIMIT SWITCH/ SENSOR PROXIMITY INDUKTIF

TUJUAN

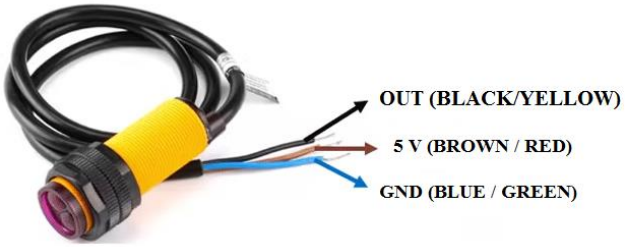
- Dapat memahami dasar-dasar pemrograman sensor digital menggunakan sensor *IR E18-D80NK*, *Limit Switch* dan sensor *Proximity Induktif*.
- Mampu menjelaskan kegunaan dari beberapa sensor yang digunakan.

PENGANTAR *SENSOR IR E18-D80NK, LIMIT SWITCH* DAN *PROXIMITY INDUKTIF*

- ***SENSOR IR E18-D80NK***

Sensor *E18-D80NK* termasuk jenis sensor biner yang digunakan untuk mendeteksi rintangan dan objek. Sensor ini bekerja berdasarkan pancaran sinar infra merah. Langkah pertama, transmitter infra merah mengirimkan sinar secara langsung. Bila sinar ini mengenai rintangan akan kembali ke penerima. Sensor ini memiliki potensiometer yang dapat disesuaikan untuk mengubah jarak deteksi dari 3 sampai 80cm. Pada sensor ini terdapat *LED* yang akan menyala bila mendeteksi adanya halangan. Ketika ada halangan, keluaran sensor *HIGH*, sebaliknya *LOW*. Sensor *E18-D80NK* tidak dapat memberikan jarak terukur sebagaimana sensor jarak HC-SR04 atau ping parallax.

Sensor ini dapat digunakan dalam robotika, sensor parkir, penghitungan *RPM*, mesin cuci tangan otomatis, dan masih banyak lagi.



Gambar 9.1. Bentuk Fisik Sensor *E18-D80NK*

Modul ini memiliki 3 pin, diataranya adalah :

- **VCC** : Catu daya modul – 5V (Coklat)
- **GND** : Ground (Biru)
- **OUT** : Keluaran digital (Hitam)

Spesifikasi dan fitur sensor IR E18-D80NK :

- Tegangan *input* : 5V DC
- Konsumsi arus : >25mA (min) ~ 100mA (max)
- Dimensi : 17cm (diameter) x 4,5 cm(panjang)
- Panjang kabel : 45 cm
- Deteksi objek : transparan atau buram
- Tipe refleksi menyebar (diffuse reflection)
- Jangkauan sensing : 3 cm sam 80 cm (tergantung dari permukaan halangan)
- NPN *output* (*normaly high*)
- Suhu lingkungan : -25^0C ~ 55^0C

- ***LIMIT SWITCH***

Limit switch banyak digunakan pada berbagai aplikasi seperti control motor, otomatisasi. Dikatakan limit switch karena fungsi utamanya digunakan untuk mendeteksi objek bergerak yang mencapai batas.

Limit switch memiliki 3 pin :

- Common → pin common antara kedua pin yaitu Nc atau NO.
- Normally Open (NO) → kondisi normal terbuka
- Normally Closed (NC) → kondisi normal tertutup

Gambar 9.2. Bentuk Fisik Limit Switch

- ***PROXIMITY INDUKTIF***

Proximity inductive merupakan salah satu juga sensor pendeteksi adanya objek. Objek yang dideteksinya berbahan logam dengan tanpa menyentuhnya. Sensor proximity induktif bekerja dengan cara membangkitkan medan magnet yang berfrekuensi tinggi yang dipancarkan pada area sekitar sensor. Bila terdapat objek berbahan logam sensor akan memberikan sinyal diakibatkan perubahan dari medan magnit. Perubahan medan magnet ini mengakibatkan perubahan pada otput sensor.

Gambar 9.3. Bentuk Fisik Proximity induktif

Modul ini memiliki 3 pin, diataranya adalah :

- **VCC** : Catu daya modul 6 V ~ 36 V (Coklat)
- **GND** : Ground (Biru)
- **OUT** : Keluaran digital (Hitam)

Kelebihan Sensor Proximity Induktif, diantaranya adalah :

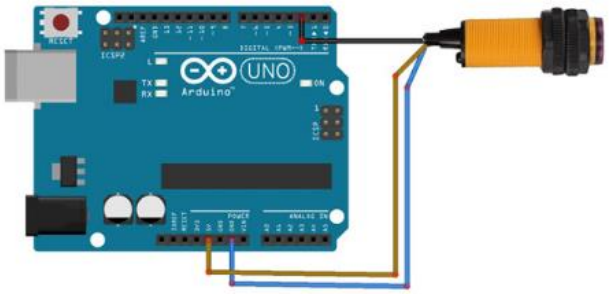
- Akurasi cukup tinggi
- Instalasinya mudah
- Dapat bekerja pada lingkungan yang keras
- Tingkat switching rate tinggi

Kekurangannya adalah :

- Hanya bisa digunakan untuk mendeteksi logam
- Jangkauan deteksi sensor biasanya terbatas.

PERCOBAAN 1. SENSOR IR E18-D80NK

- Buat *program Sensor IR E18-D80NK* sebagaimana yang ditunjukkan gambar 9.5.
- Hubungkan kabel penghubung dari pin 2 ke *output sensor*.
- Rangkailah sensor *IR E18-D80NK* ke *Arduino* ditunjukkan gambar 9.4.
- Hubungkan Supply 5 V ke Vcc sensor dan Gnd ke Gnd sensor.
- Pastikan *Vcc Arduino* sudah dihubungakan dengan *Vcc* Trainer.
- Lakukan *Upload* program yang telah dibuat.
- Bila program sukses mendeteksi objek ditandai dengan nyalanya indikator pada sensor, tambahkan programnya dengan nyalanya buzzer secara flip-flop bila sensor mendeteksi objek.



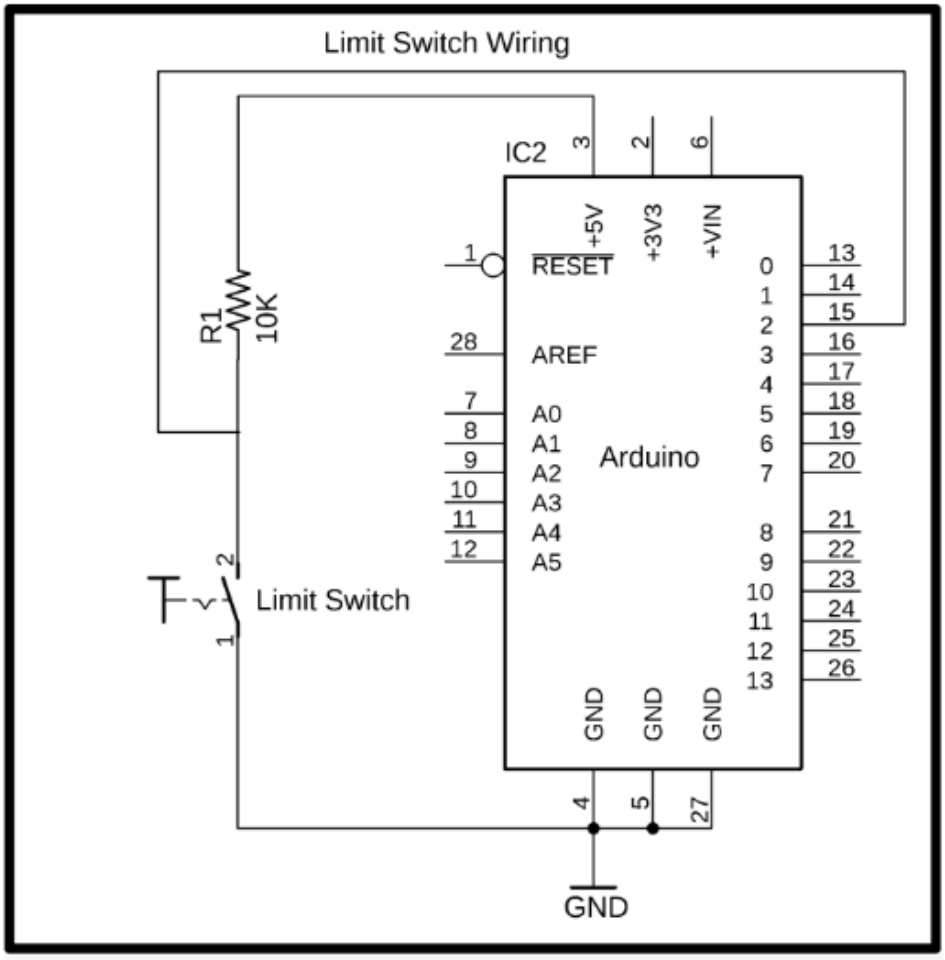
Gambar 9.4. Pengkawatan *Sensor IR E18-D80NK* ke *Arduino*

```
#define Sensor_IR 2

void setup() {
  Serial.begin(9600);
  pinMode(Sensor_IR, INPUT);
}

void loop() {
  if (digitalRead(Sensor_IR) == HIGH)
  {
    Serial.println("Objek terdeteksi!");
  }
  else
  {
    Serial.println("Tidak ada objek");
  }
  delay(100);
}
```

Gambar 9.5. Listing Program *Sensor IR E18-D80NK*

PERCOBAAN 2. LIMIT SWITCH

- Buat *program Limit Switch* sebagaimana yang ditunjukkan gambar 9.7.
- Hubungkan kabel penghubung dari pin 2 ke *output limit switch*.
- Rangkailah sensor *Limit Switch* ke *Arduino* ditunjukkan gambar 9.6.
- Hubungkan Supply 5 V ke resistor 10K yang diserikan dengan limit switch dan Gnd ke salah satu kaki dari limit switch.
- Pastikan *Vcc Arduino* sudah dihubungakan dengan *Vcc* Trainer.
- Lakukan *Upload* program yang telah dibuat.
- Bila program sukses mendeteksi objek ditandai dengan pembacaan di serial monitor, tambahkan programnya dengan nyalanya buzzer sesaat bila limit switch mendeteksi objek.



Gambar 9.6. Pengkawatan *Limit Switch* ke *Arduino*

```
#define ls 2
int state = LOW;
int value;

void setup() {
  Serial.begin(9600);
  pinMode(ls, INPUT);
}

void loop() {
  value = digitalRead(ls);
  Serial.println("ls value = ");
  if (value == LOW)
  {
    Serial.println("Objek terdeteksi!");
  }
  else
  {
    Serial.println("Tidak ada objek");
  }
  delay(100);
}
```

Gambar 9.7. Listing Program *Sensor IR E18-D80NK*

PERCOBAAN 3. SENSOR PROXIMITY INDUKTIF

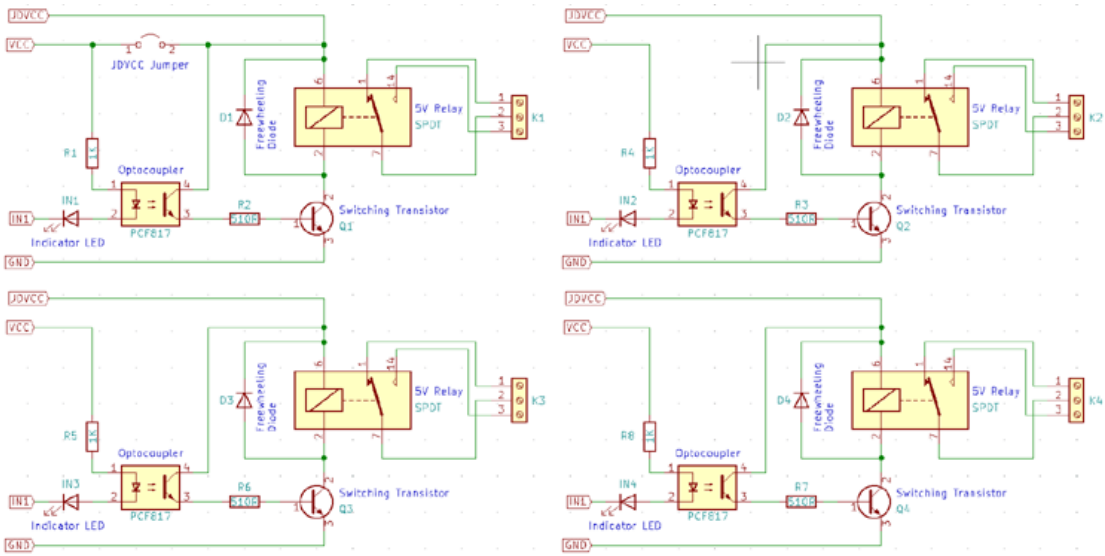
- Buat *program Sensor proximity induktif* sebagaimana yang ditunjukkan gambar 9.9.
- Hubungkan kabel penghubung dari pin 13 ke *output sensor*.
- Rangkailah skematik relay sebagai referensi pengkawatan ke *arduino* ditunjukkan gambar 9.8.
- Supply sensor beroperasi dari 6 V ~ 36 V dan hubungkan Gnd ke Gnd sensor.
- Pastikan *Vcc Arduino* sudah dihubungakan dengan *Vcc* Trainer.
- Lakukan *Upload* program yang telah dibuat.
- Bila proximity sukses mendeteksi objek ditandai dengan pembacaan di serial monitor, tambahkan programnya dengan nyalanya buzzer. Tampilkan juga respon logika pada sensor disaat mendeteksi objek.

Sebagai catatan :
Sensor Proximity induktif memiliki 3 pin :

- VCC : Catu daya modul 6 V ~ 36 V (Coklat)
- GND : Ground (Biru)
- OUT : Keluaran digital (Hitam).

Tegangan *output* = 11,4 V (saat tidak mendeteksi logam)

Tegangan *output* = 0 V (saat mendeteksi logam)



Gambar 9.8. Rangkaian skematik relay 4 channel

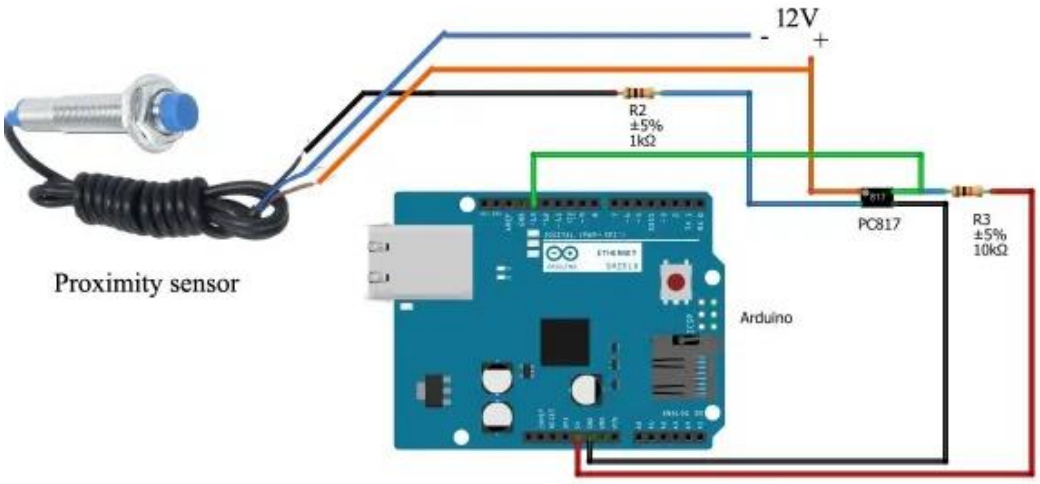
```
#define proximity 13
int state = LOW;
int value;

void setup() {
  Serial.begin(9600);
  pinMode(proximity, INPUT);
}
```

```
void loop() {
  value = digitalRead(proximity);
  Serial.println("sensor value = ");
  if (value == LOW)
  {
    Serial.println("Objek terdeteksi!");
  }
  else
  {
    Serial.println("Tidak ada objek");
  }
  delay(100);
}
```

Gambar 9.9. Listing Program *Sensor Proximity induktif*

SOAL ESSAY

1. Buatlah program counter menggunakan sensor proximity induktif yang menghitung adanya objek logam. Jumlah hitungan ditampilkan pada display *LCD* dan siapkan juga tombol untuk mereset jumlah hitungan ?



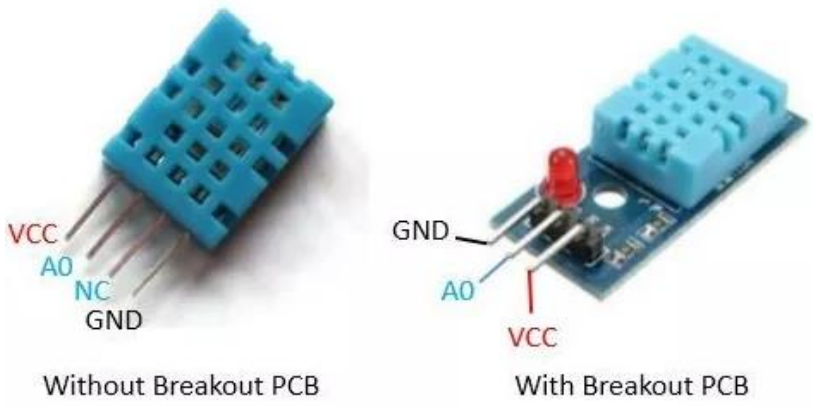
PELAJARAN 9
APLIKASI SENSOR SUHU *DHT 11*

TUJUAN

- Dapat memahami dasar-dasar pemrograman suhu menggunakan sensor *DHT 11*.
- Mampu menjelaskan kegunaan dari pembangkit Sensor *DHT11*.

PENGANTAR *DHT11*

Sensor *DHT11* adalah modul sensor yang berfungsi untuk mensensing objek suhu dan kelembaban yang memiliki *output* tegangan *analog* yang dapat diolah lebih lanjut menggunakan mikrokontroller. Modul sensor ini tergolong kedalam elemen resistif seperti perangkat pengukur suhu, contohnya *NTC*. Sensor ini *DHT11* memiliki 4 kaki pin dan terdapat juga sensor *DHT11* dengan *breakout PCB* yang terdapat hanya memiliki 3 kaki pin, sebagaimana ditunjukkan gambar 9.1.



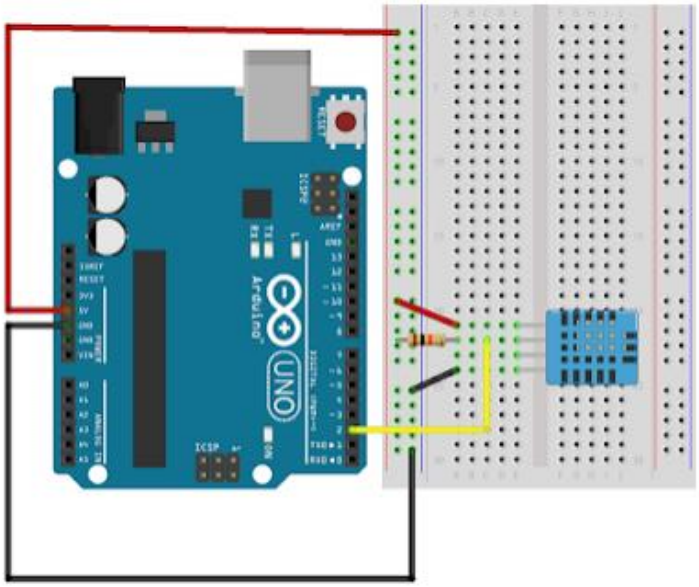
Gambr 9.1. Bentuk Fisik *DHT11*

Spesifiksi *DHT11* :

- Tegangan *input* : 5 Vdc
- *Range* temperatur : 0 – 50^0 C *error* ± 2^0 C
- Kelembaban : 20 – 90 % RH ± 5 % RH *error*

PERCOBAAN SUHU_1 : **TAMPILAN SUHU**

- Buat *program SUHU* sebagaimana yang ditunjukkan gambar 9.3.
- Hubungkan kabel penghubung dari pin 13 ke *led*.
- Rangkailah sensor *DHT11* ke *Arduino* sebagainama ditunjukkan gambar 9.2.
- Gunakan Resistor 10 k Ω.
- Nyalakan *Power Supply* dengan tegangan 7 – 12 V ke *Arduino*.
- Pastikan *GND Arduino* sudah dihubungakan dengan *GND* Trainer.
- Lakukan *Upload* program yang telah dibuat.



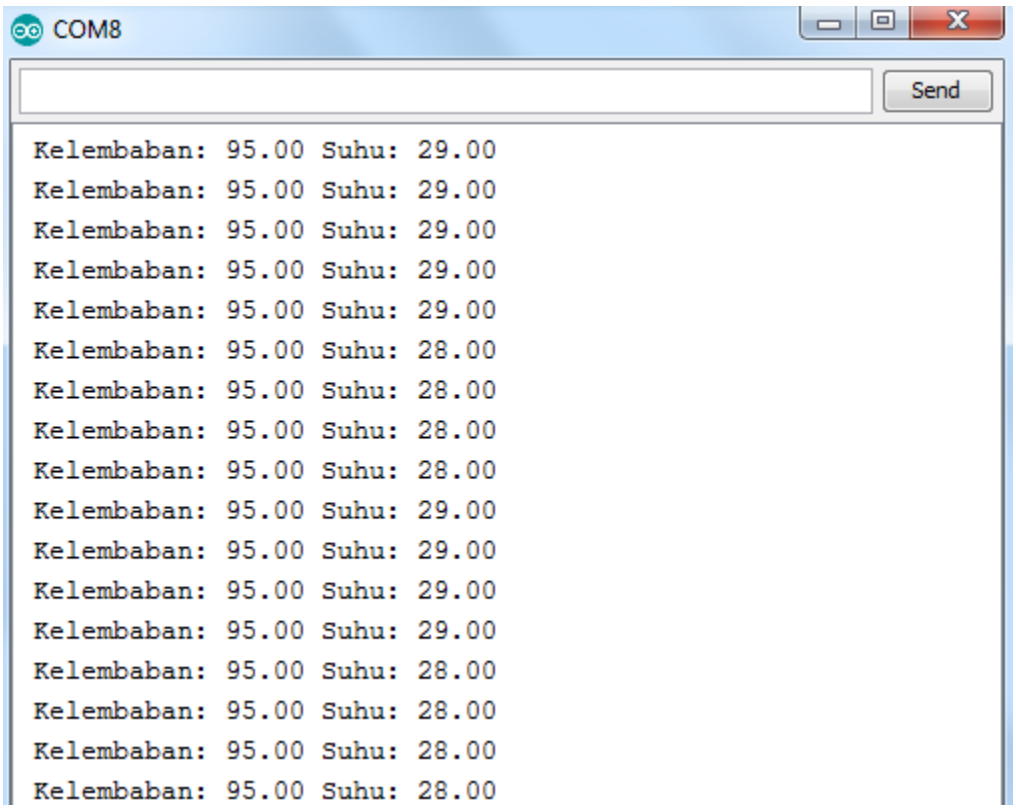
Gambar 9.2. Listing Program *SUHU*_1

```
#include "DHT.h"  //Memasukan Library DHT ke Program
//menggunakan pin 2 untuk pemasangan sensornya
#define DHTPIN 2
//memilih tipe DHT11, bisa diubah menjadi DHT22, DHT21
#define DHTTYPE DHT11

//setting pin yang dipilih dan tipe DHT
DHT dht(DHTPIN, DHTTYPE);
int led= 13;
```

```
void setup() {
Serial.begin(9600);
dht.begin(); //Komunikasi DHT dengan Arduino
pinMode(led, OUTPUT);
}

void loop() {
//menyimpan nilai Humidity pada variabel kelembaban
float kelembaban = dht.readHumidity();
  //menyimpan nilai Temperature pada variabel suhu
  float suhu = dht.readTemperature();

 Serial.print(" Kelembaban: ");
   Serial.print(kelembaban);
 Serial.print(" Suhu: ");
    Serial.println(suhu);

   delay(500);

 if ( kelembaban > 65.00){
         digitalWrite(led, HIGH);
 }
 if ( suhu > 32.00){
 digitalWrite(led, HIGH);
 }
 if ( kelembaban < 66.00){
 digitalWrite(led, LOW);
 }

  if ( suhu < 33.00){
  digitalWrite(led, LOW);
  }
  }
```

Gambar 9.3. Listing Program *SUHU_1*

Program pada gambar 9.3 digunakan untuk membaca suhu dan kelembaban. Hasil pembacaan nilai dari suhu dan kelembaban dapat dilihat pada serial monitor, sebagaimana ditunjukkan gambar 9.4. Led akan menyala bila kelembaban kurang dari 66 dan suhu kurang dari 33 derajat.



Gambar 9.4. Tampilan suhu dan kelembaban pada serial monitor

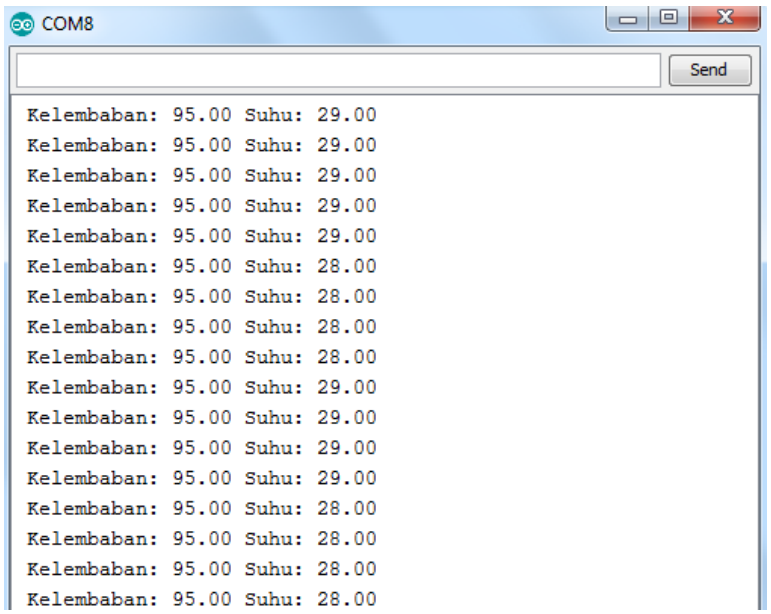
PERCOBAAN SUHU_2 :

- Buat *program SUHU_2* sebagaimana ditunjukkan gambar 9.5.
- Beri komentar dari hasil eksekusi *program* yang telah dibuat.

```
#include "DHT.h"
#define DHTPIN A2
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("Pengujian DHT11!");
  //prosedur memulai pembacaan module sensor
  dht.begin();
}
```

```
void loop() {
  delay(2000);
  float humidity_1 = dht.readHumidity();
  float celcius_1 = dht.readTemperature();
  float fahrenheit_1 = dht.readTemperature(true);

//mengecek pembacaan apakah terjadi kegagalan atau tidak
if (isnan(humidity_1) || isnan(celcius_1) || isnan(fahrenheit_1)) {
  Serial.println("Pembacaan data dari module sensor gagal!");
  return;
}

float htof = dht.computeHeatIndex(fahrenheit_1, humidity_1);
float htoc = dht.computeHeatIndex(celcius_1, humidity_1, false);

Serial.print("Kelembaban: ");
Serial.print(humidity_1);
Serial.print(" %\t");

     Serial.print("Suhu : ");
     Serial.print(celcius_1); //format derajat celcius
     Serial.print(" 'C ");
     Serial.print(fahrenheit_1); //format derajat fahrenheit
     Serial.print(" 'F\t");




     Serial.print("Indeks Panas: ");
     Serial.print(htof);
     Serial.println(" *F");

     Serial.print(htoc);
     Serial.print(" *C ");
   }
```

Gambar 9.5. *Listing Program* Percobaan *SUHU_2*



Gambar 9.6. Tampilan suhu dan kelembaban pada serial monitor

PELAJARAN 10
APLIKASI SENSOR *ULTRASONIC*

TUJUAN

- Dapat memahami dasar-dasar pemrograman *Ultrasonic.*
- Mampu menjelaskan kegunaan dari sensor *Ultrasonic.*
- Dapat memahami prinsip kerja *Ultrasonic*.
- Dapat membuat program sederhana menggunakan *Ultrasonic*.

PENGANTAR *ULTRASONIC*

Sensor *ultrasonic* adalah sensor yang berfungsi untuk merubah besaran fisis (suara) menjadi besaran listrik maupun sebaliknya yang dikonversi menjadi jarak.

Gambar 10.1. Sensor *Ultrasonic* HC-SR04

Tabel 10.1. Hubungan pin di sensor *Ultrasonic HC-SR04*

Pin	Keterangan
Pin 1	*VCC* (dihubungkan ke tegangan +5 V)
Pin 2	*Trig* (untuk mengirim gelombang suara)
Pin 3	*Echo* (untuk menerima gelombang suara)
Pin 4	*Gnd* (dihubungkan ke *ground*)

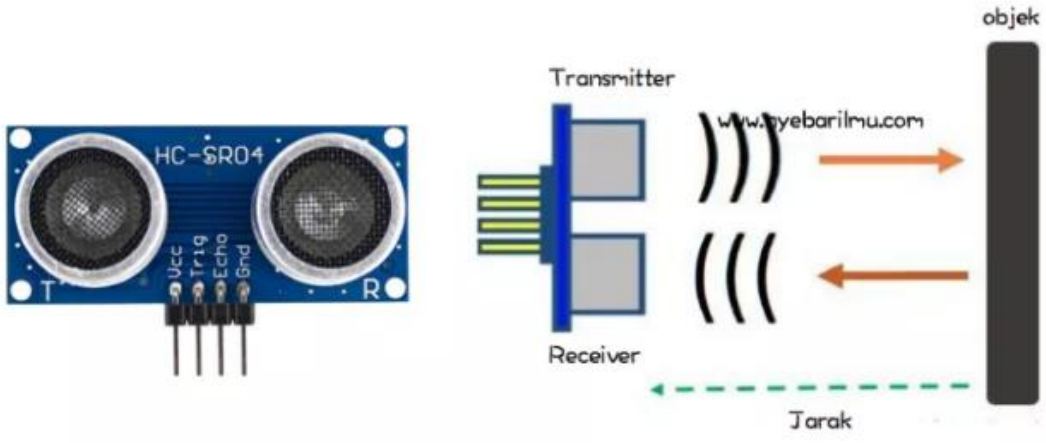
Konsep dasar dari sensor ini yaitu memanfaatkan prinsip pemantulan gelombang suara yang dapat diaplikasikan untuk menghitung jarak benda dengan frekuensi yang ditentukan sesuai dengan sumber *oscillator*. Disebut sebagai sensor *ultrasonic* dikarenakan sensor ini mengaplikasikan gelombang *ultrasonic* sebagai *tranducer*-nya. Gelombang *ultrasonic* merupakan gelombang suara yang memiliki frekuensi tinggi yaitu pada kisaran 20 kHz. Bunyi ini tidak bisa didengar dengan telinga normal manusia, hanya bisa didengar oleh sistem pendengaran pada kelelawar, anjing, lumba-lumba, dan kucing.

Sifat gelombang *ultrasonic* hanya bisa merambat melalui zat cair, padat dan gas. Reflektivitas gelombang *ultrasonic* pada permukaan benda padat hampir sama dengan reflektivitas suara *ultrasonic* dengan permukaan benda cair. Meskipun begitu pada gelombang bunyi *ultrasonic* akan mudah diserap oleh bahan-bahan tertentu seperti bahan dari busa maupun tekstil.

Cara kerja

Sensor ini dimulai dari gelombang *ultrasonic* dengan frekuensi tertentu yang dibangkitkan melewati alat yang disebut juga dengan nama piezoelektrik sebagai *transmitter*. Alat ini akan menghasilkan gelombang *ultrasonic* yang berfrekuensi 40 kHz (sesuai dengan osilator yang terpasang pada sensor). Biasanya alat ini akan memancarkan gelombang pada suatu target dan jika sudah mengenai permukaan target, maka gelombang tersebut akan terpantulkan kembali.

Pantulan gelombang *ultrasonic* akan diterima oleh piezoelektrik (*receiver*) dan kemudian sensor akan mengkalkulasi perbedaan antara waktu pengiriman dan waktu gelombang pantul yang diterima.



Gambr 10.2. Ilustrasi Prinsip Kerja Sensor *Ultrasonic*

Keterangan :

Pemancar *ultrasonic* akan memancarkan gelombang dengan frekuensi 40 kHz dengan jeda waktu tertentu. Kecepatan rambat gelombang bunyi yaitu kisaran 340 m/s. Setelah gelombang pantulan mengenai alat penerima, gelombang tersebut akan diolah untuk dihitung jarak benda tersebut.

Rumus jarak benda dapat dihitung dengan rumus segai berikut :

$$S = (340 . t) / 2$$

S = jarak
t = selisih waktu dipancarkan dan waktu diterima gelombang

Bagian-bagian dari Sensor *Ultrasonic* Antara Lain :

1. Piezoelektrik
   Berfungsi sebagai alat pengubah energi listrik menjadi energi mekanik. Materi dasar yang terdapat pada piezoelektrik yang menghasilkan medan listrik saat terjadi tekanan mekanis dan sebaliknya. Misalnya rangkaian pengukur dioperasikan pada mode pulsa dengan unsur piezoelektrik yang sama, sehingga bisa digunakan sebagai mode *receiver* dan *transmitter*.
   Frekwensi dihasilkan tergantung dari osilator yang terpasang dan itu akan disesuaikan dengan frekwensi kerja dari *tranducer*.
2. *Transmitter*
   Merupakan alat yang mempunyai peran sebagai pemancar gelombang dengan frekwensi 40 kHz yang bersumber dari osilator. Frekwensi tersebut dihasilkan dari rangkaian osilator serta *amplifier* sinyal / penguat sinyal.
   Pada *amplifier* sinyal akan menghasilkan sinyal listrik yang diumpankan ke piezoelektrik dan terjadilah reaksi mekanik.
3. *Receiver*
   Terdiri dari *tranducer ultrasonic* yang memakai piezoelektrik juga yang difungsikan sebagai penerima gelombang pantulan. Bahan piezoelektrik mempunyai reaksi yang *reversible*, terdapat elemen keramik yang berfungsi sebagai pembangkit tegangan listrik.
   Pada waktu gelombang datang dengan kriteria frekwensi yang resonansi dan pada saat itu akan menggetarkan bahan pezoelektrik.

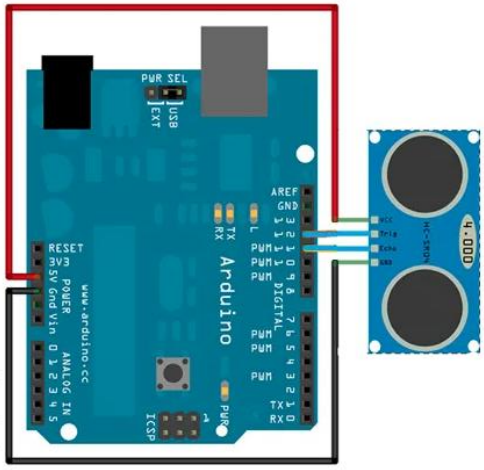
Spesifikasi

Modul yang ada dipasaran yang sering digunakan untuk *arduino* diantaranya adalah tipe *HC-SR04*. Spesifikasi sensor *ultrasonic HC-SR04* adalah sebagai berikut :

- Jarak deteksi antara 2 cm – 300 cm
- Tingkat kepresisian pengukuran jarak ± 3 mm
- Tegangan operasional 5 V Dc
- Sudut sensor < 15 cm
- Konsumsi arus berkisar 2 mA
- Dimensi modul 45 mm x 20 mm

PERCOBAAN ULTRASONIC_1 : BASIC ULTRASONIC

- Buat *program ULTRASONIC* sebagaimana yang ditunjukkan gambar 19.4.
- Hubungkan kabel penghubung dari *arduino* ke *ultrasonic* sebagaimana ditunjukkan gambar 19.3. (Pin 11 ke pin *Echo* dan Pin 12 ke pin *Trig*)
- Nyalakan *Power Supply* dengan tegangan 7 – 12 V ke *Arduino*.
- Pastikan *GND Arduino* sudah dihubungakan dengan *GND* Trainer.
- Lakukan *Upload* program yang telah dibuat.



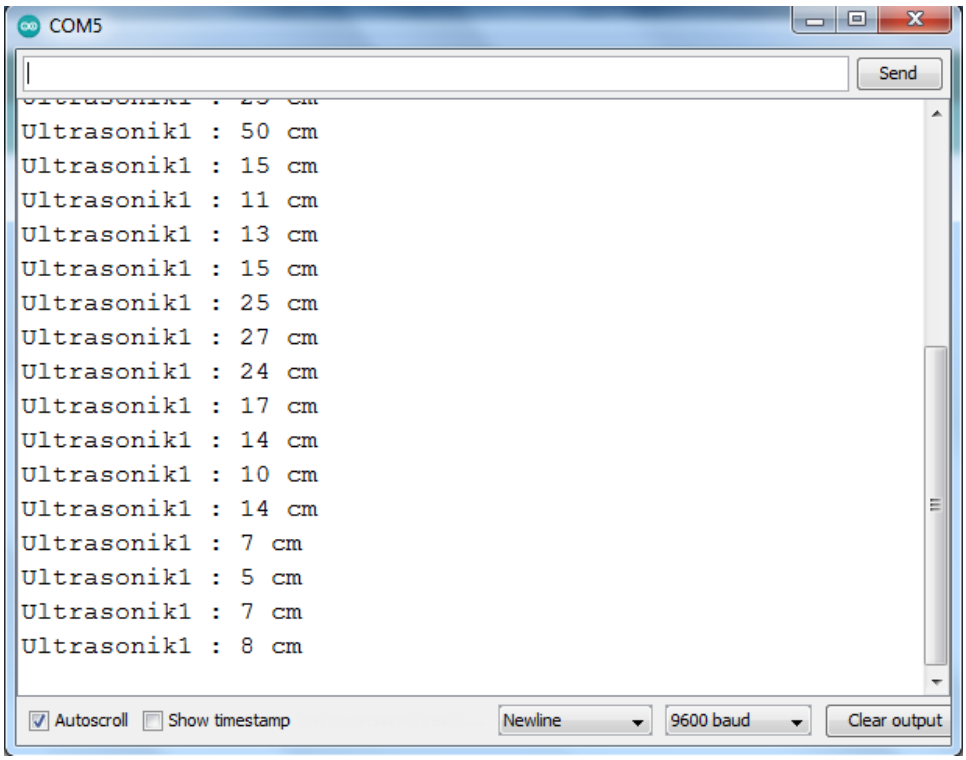
Gambar 10.3. Rangkaian Pecobaan *Ultrasonic Arduino*

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

LiquidCrystal_I2C lcd(0x27, 16, 2);

const int Trigger = 12;
const int Echo = 11;

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();  // layar lcd terang

  lcd.setCursor(0,0);
  lcd.print("Tes Ultrasonic");
  delay(1000);
  pinMode(Trigger, OUTPUT);
  pinMode(Echo, INPUT);
}
```

```
long duration1;
long centi;

void loop()
{
  digitalWrite(Trigger, LOW);
  delayMicroseconds(10);
  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);

  digitalWrite(Trigger, LOW);
  duration1 = pulseIn(Echo, HIGH);

  centi = microsecondsKeCenti(duration1);
  lcd.clear();
  Serial.print("Ultrasonik1 : ");
  lcd.print("Jarak : ");
  lcd.setCursor(9,0);
  lcd.print(centi);
  lcd.setCursor(14,0);
  lcd.print("cm");
  Serial.print(centi);
  Serial.println(" cm");

  delay(500);
}

long microsecondsKeCenti(long microseconds)
{
  return microseconds / 29 / 2;
}
```

Gambar 10.4. Listing Program *ULTRASONIC*_1

Program pada gambar 10.4 digunakan untuk membaca Jarak suatu objek. Hasil pembacaan nilai dari objek dapat ditampilkan melalai layar serial monitor ataupun dapat juga ditampilakan melalui layar *lcd*. Gambar 10.5 tampilan jarak dari layar serial monitor. Sedangkan gambar 10.6 tampilan jarak dari layar *lcd*.



Gambar 10.5. Tampilan jarak pada serial monitor

Gambar 10.6. Tampilan jarak pada layar *LCD*

SOAL ESSAY

1. Buatlah program menggunakan sensor *ultrasonic* dengan ketentuan, bila jarak yang terbaca lebih dari 5 cm mak buzzer akan menyal secara *flip-flop* dan bila jarak melebihi 49 cm maka buzzer akan padam. *Buzzer* juga akan padam bila jarak dalam membaca objek kurang dari sama dengan 5 cm ?

PELAJARAN 11
APLIKASI *RTC*

TUJUAN

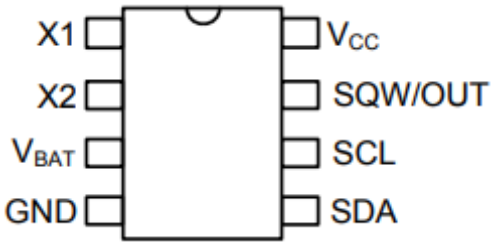
- Memahami fungsi dari penggunaan *RTC*
- Dapat memprogram penggunaan jam digital sederhana dengan tampilan RTC

PENGANTAR *RTC*

RTC (*Real Time Clock*) adalah perangkat yang memungkinkan untuk menghasilkan waktu yang tepat karena dilengkapi pembangkit waktu dan baterai. *RTC* sangat terkenal sebagai modul interface untuk jam digital, bentuknya cukup kecil dan sangat bermanfaat sekali bagi siapa saja yang ingin membuat jam digital, karena sangat praktis.

DS1307 RTC Pin Diagram

Gambar 11.1 menunjukkan pin *IC RTC DC1307*. Untuk mengurangi konsumsi daya, jumlah pin pada *IC* harus dikurangi. Oleh karena itu, *DS1307 RTC* menggunakan komunikasi *I2C*.



Gambar 11.1. Pin *DS1307 RTC*

Keterangan pin *DS1307 RTC* :

- X1 dan X2 : pin konektor *IC RTC* dengan frekwensi *crystal* 32768 Hz pada oscilator internal modul. Jika menggunakan oscillator eksternal maka pin ini dibiarkan terbuka.
- VBAT : Pin sumber tegangan baterai dihubungkan dengan 3 V *Lithium cell* sebagai *backup* dayanya.
- *GND* : *Ground* pin

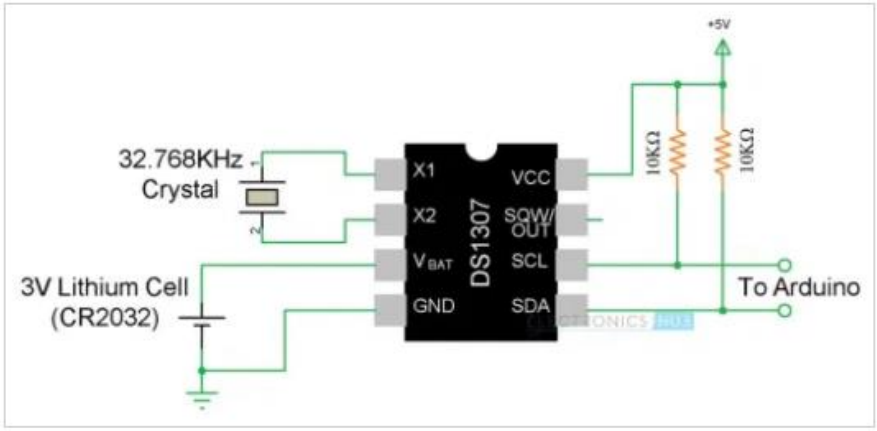
- *SDA* : *Serial data pin*. Ini adalah pin khusus data *input/output* dari *interface I2C*. Tegangan 5 V diberikan pada resistor 10 kΩ.
- *SCL* : *Serial Clock Input Pin*. Pin *input clock* dari *I2C interface*. Pin ini juga sebagai penarikan tegangan 5 V juga dibutuhkan resistor 10 KΩ.
- *SQW/OUT* : Pin *Out* gelombang persegi (*Square wave*). Bila tidak dibutuhkan abaikan saja.
- *VCC* : Pin sumber tegangan utama.

Modul *RTC DS1307* harga yang murah rendah konsumsi daya dan mantap sebagai *IC* pewaktu dan penanggalan secara lengkap dan akurat (detik, menit, jam, hari, tanggal, bulan, dan tahun.

Sekilas beberapa fitur *DS1307* yaitu:

- Fungsi pewaktu yang akurat
- Validasi penanggalan hingga 2100
- Rendah konsumsi daya hanya 500nA saat beroperasi dengan *battery li-ion CR1032*
- Otomatis *switching power supply* saat tegangan utama bermasalah.
- 24 jam atau 12 jam dengan Indikator *AM/PM*
- Bentuk mungil seukuran *battery* komponen yang sedikiti sehingga cukup praktis

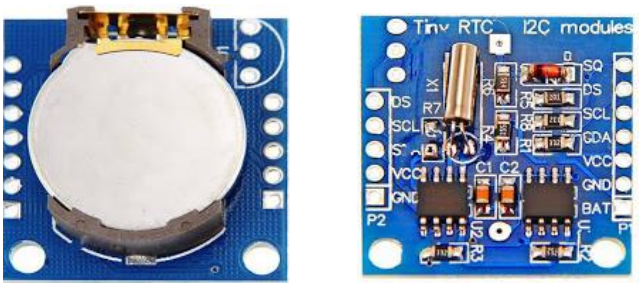
Berikut ditunjukkan modul *RTC DS1307*, yang berisi rangkaian modul jam pewaktu.



Gambar 11.2. Rangkaian *RTC DS1307*

Baterai internal 3V digunakan untuk menyimpan tanggal dan waktu, berupa detik, menit, jam, tanggal, bulan, dan tahun. Module RTC yang umum di gunakan di Mikrokontroler memakai *IC* DS3231, DS1307 dan DS1302.

Arduino secara internal tidak di lengkapi denga *RTC* sehingga untuk membuatan *project* yang membutuhkan waktu *Real Time* di perlukan module *RTC* yang sudah terpasang baterai, sehingga saat power supply mati, *RTC* tetap dapat *power supply* dari baterai.
Mikrokontroler *Arduino* berkomunikasi dengan *RTC DS1307* Menggunakan komunikasi *I2C* (*Inter-Integrated Circuit*). Sehingga pin yang di gunakan ada 4 yaitu *VCC* untuk supply tegangan 5V *DC*, *GND* , *SDA* (*Serial Data*) dan *SCL* (*Serial Clock*).



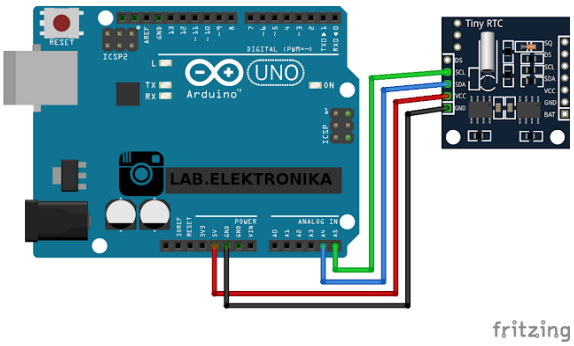
Gambar 11.3. RTC DS1307 tampak atas dan tampak bawah

PERCOBAAN RTC_1 : **JAM DIGITAL DI MONITORING DARI SERIAL MONITOR**

- Buat *program RTC* sebagaimana yang ditunjukkan gambar 11.5.
- Rangkailah MODUL *RTC DS1307* ke *Arduino* sebagainama ditunjukkan gambar 11.4.
  Alokasi pin RTC dan *Arduino* sebagai berikut :

RTC DS1307	***Arduino***
SCL	A5
SDA	A4
VCC	+5V
GND	Gnd

- Nyalakan *Power Supply* dengan tegangan 7 – 12 V ke *Arduino*.
- Pastikan *GND Arduino* sudah dihubungakan dengan *GND Trainer*.
- Lakukan *Upload* program yang telah dibuat.
- Beri komentar dari hasil eksekusi program yang telah dibut.



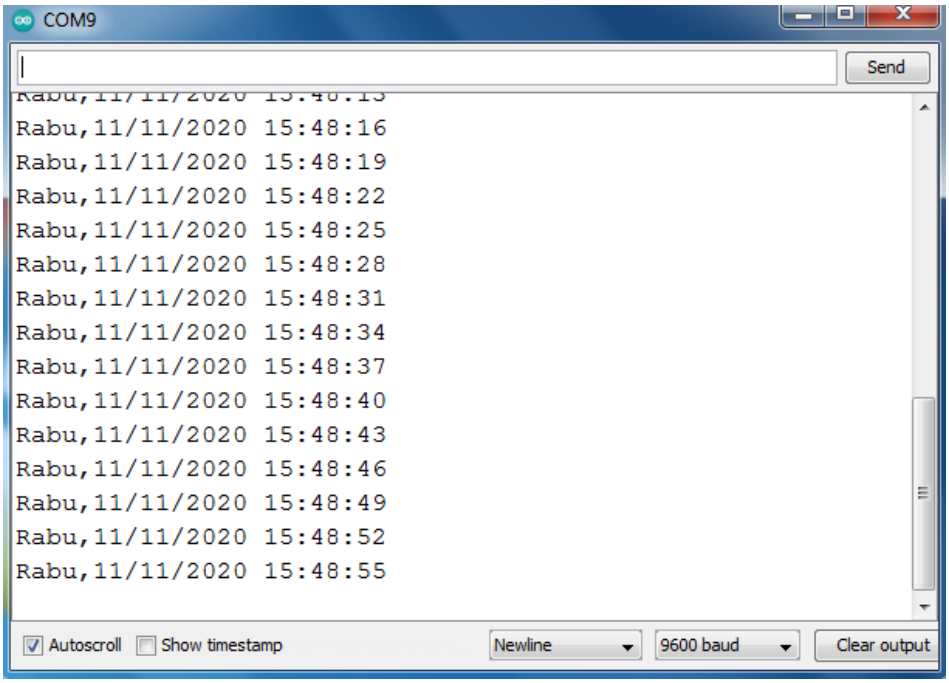
Gambar 11.4. Wiring Diagam RTC DS1307 ke *Arduino*

```
#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 rtc;

char namaHari[7][12] = {"Minggu", "Senin", "Selasa",
                        "Rabu", "Kamis", "Jumat", "Sabtu"};
void setup () {
  Serial.begin(9600);
  if (! rtc.begin()) {
    Serial.println("RTC TIDAK TERBACA");
    while (1);
  }

  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    //rtc.adjust(DateTime(2020, 8, 29, 19, 28, 0));
  }
}
void loop () {
    DateTime now = rtc.now();
    Serial.print(namaHari[now.dayOfTheWeek()]);
    Serial.print(',');
    Serial.print(now.day(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.year(), DEC);
    Serial.print(" ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

    delay(3000);
}
```

Gambar 11.5. Listing Program Jam Digital

Program pada gambar 11.5 digunakan untuk menampilkan jam digital. Jam digital akan tampil di layar serial monitor dengan tampilan Hari, tanngal, bulan, tahun, Jam, Menit dan detik. Instruksi rtc.adjust(DateTime(2020, 8, 29, 19, 28, 0)); digunakan untuk setting tanggal dan waktu secara manual. Sedangkan instruksi rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); digunakan untuk setting waktu dari waktu komputer.

Berikut tampilan tanggal dan waktu yang dihasilkan dari layar serial montir yang ditunjukkan gambar 11.6.



Gambar 11.6. Hasil dari tampilan jam digital dari serial monitor

PERCOBAAN RTC_2 : **JAM DIGITAL YANG DIMPILKAN *VIA LCD***

- Buat *program RTC* sebagaimana yang ditunjukkan gambar 11.7.
- Alokasi pin RTC dan *Arduino* :

RTC DS1307	***Arduino***
SCL	A5
SDA	A4
VCC	+5V
GND	Gnd

- Nyalakan *Power Supply* dengan tegangan 7 – 12 V ke *Arduino*.
- Pastikan *GND Arduino* sudah dihubungakan dengan *GND Trainer*.
- Lakukan *Upload* program yang telah dibuat.
- Beri komentar dari hasil eksekusi *program* yang telah dibuat.

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

  RTC_DS1307 rtc;
  LiquidCrystal_I2C lcd(0x27, 16, 2);

  char daysOfTheWeek[7][12] = {"Sun", "Mon", "Tue",
  "Wed", "Thu", "Fri", "Sat"};

void setup ()
{
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();  // layar lcd terang

  lcd.setCursor(0,0);
  lcd.print(" Tes  ..... ");
  delay(1000);

  if (! rtc.begin())
  {
    lcd.print("Couldn't find RTC");
    while (1);
  }

  if (! rtc.isrunning())
  {
    lcd.print("RTC is NOT running!");
  }
    //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    //rtc.adjust(DateTime(2020, 8, 29, 19, 28, 0));
}

void loop ()
{
    DateTime now = rtc.now();
```

```
    lcd.setCursor(0, 1);
    lcd.print(now.hour());
    lcd.print(':');
    lcd.print(now.minute());
    lcd.print(':');
    lcd.print(now.second());
    lcd.print("    ");

    lcd.setCursor(0, 0);
    lcd.print(daysOfTheWeek[now.dayOfTheWeek()]);
    lcd.print(" ,");
    lcd.print(now.day());
    lcd.print('/');
    lcd.print(now.month());
    lcd.print('/');
    lcd.print(now.year());

}
```

Gambar 11.7. Listing Program Jam Digital

Program yang ditunjukkan gambar 11.7 merupakan program jam digital yang hasil dari pewaktuannya akan ditampilkan pada layar *LCD*, sebagaimana ditunjukkan gambar 11.8.

Gambar 11.8. Tampilan Jam digital dari layar *LCD*

PELAJARAN 12
APLIKASI MOTOR DC

TUJUAN

- Memahami dasar-dasar pemrograman untuk menggerakkan *motor DC.*
- Mampu membuat aplikasi *mobole robot* sederhana menggunakan *Arduino*.

PENGGUNAAN MOTOR DC

Motor DC adalah suatu *motor* penggerak yang dikendalikan dengan arus searah (*DC*). Pengontrolan *motor DC* dapat dilakukan dengan menggunakan resistor variabel, *auto transformator*, *transistor* dan lain-lain. Namun cara tersebut akan mengkonsumsi daya yang relatif besar. Untuk mengatasi permasalahan tersebut, terdapat 2 cara lain yang sering digunakan yaitu menggunakan :

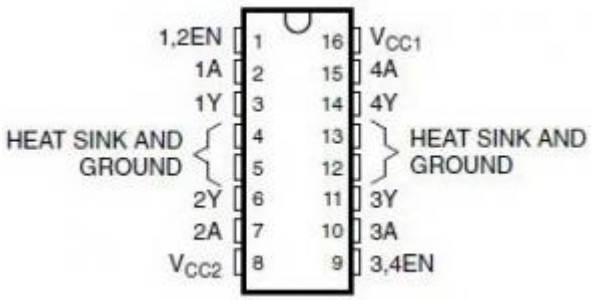
1. *Pulse Frequency Modulation*

   Cara ini memanfaatkan lebar pulsa yang konstan. Untuk mendapatkan tegangan rata-rata yang dapat diatur yaitu dengan memberikan perubahan frekuensi pada *switch on/off.*

2. *Pulse Width Modulation*

   Cara ini menggunakan frekuensi yang konstan. Daya *DC* diberikan ke *motor DC* dengan pengaturan modulasi waktu *switching on/off* (lebar pulsa) yang berubah-ubah. Pengaturan lebar pulsa ini untuk mendapatkan tegangan rata-rata yang dapat di atur.

Pada contoh kasus modul ini, kita akan mempraktekkan pengaturan kecepatan dan arah putar *motor DC*. Untuk mengatur kecepatan *motor DC* menggunakan metode *PWM,* sedangkan pengaturan arah gerak *motor DC* menggunakan rangkaian *H-Bridge* sebagai *driver motor*. Untuk komponen rangkaian *driver motor* yang menggunakan *H-Bridge* sudah ada tersedia dipasaran yaitu diantaranya menggunakan *IC L293D*. Gambar 12.1 merupakan gambaran dari konstruksi *IC L293D*.



Gambar 12.1. Koneksi Pin *IC L293D*

IC L293D yang ditunjukkan gambar 12.1 dan 12.2 dapat digunakan untuk mengatur kecepatan dan arah putar 2 *motor DC* sekaligus. Sehingga komponen ini sering sekali digunakan bagi penggiat *mobile robot line follower*. Sisi kiri *IC* dapat digunakan sebagai *motor* 1 dan sisi kanan IC dapat digunakan sebagai *motor* 2.

Motor 1:

1A
2A → **Input M1**

1-2EN → **digunakan untuk meng-enable-kan (On/Off) motor 1**

1Y
2Y → **Output M1**

VCC1 →**Supply Voltage**

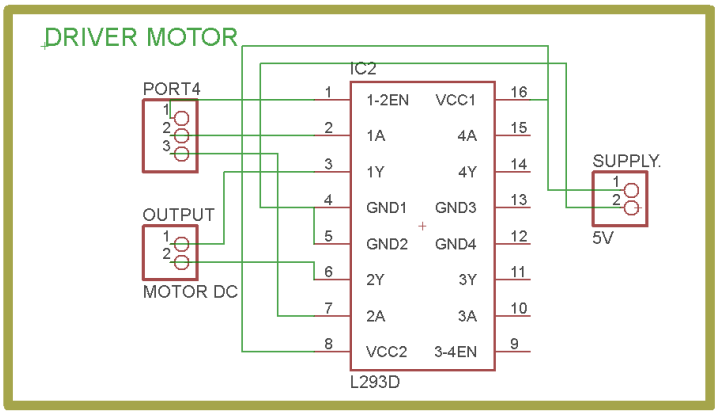
VCC2 →**Logic Supply Voltage**

Motor 2:

3A
4A → **Input M2**

3-4EN → **digunakan untuk meng-enable-kan (On/Off) motor 2**

3Y
4Y → **Output M2**



Gambar 12.2. Skematik rangkaian *driver motor* menggunakan *IC L293D*

Tabel 12.1. Pengaturan kecepatan dan arah putar *motor* menggunakan *IC L293D*

1A	*2A*	*Kondisi Motor 1*
0	0	motor 1 *stop*
0	1	putar searah jarum jam
1	0	putar berlawanan jarum jam
1	1	motor *stop*

Untuk melakukan pengaturan *motor DC* menggunakan *IC L293D* harus mengikuti ketentuan-ketentuan yang diberikan oleh *data sheed IC L293D*. Tabel 12.1 menunjukkan persyaratan yang harus dipenuhi oleh *user* untuk melakukan pengaturan kecepatan dan arah putar *motor*. Untuk membuat *motor* berputar searah atau berlawanan arah jarum jam, *Pin 1A* dan *1B* harus diberi logika yang berbeda. Selain diberi logika yang berbeda *Pin 1-2 En* juga harus diberi logika "1". Untuk membuat *motor* dapat perputar dengan kecepatan yang dapat diatur maka *Pin 1-2 En* diberikan sinyal *PWM*, yaitu memberikan modulasi sinyal *On/Off* dengan pengaturan lebar pulsa yang bervariasi dan dengan frekwensi yang tetap.

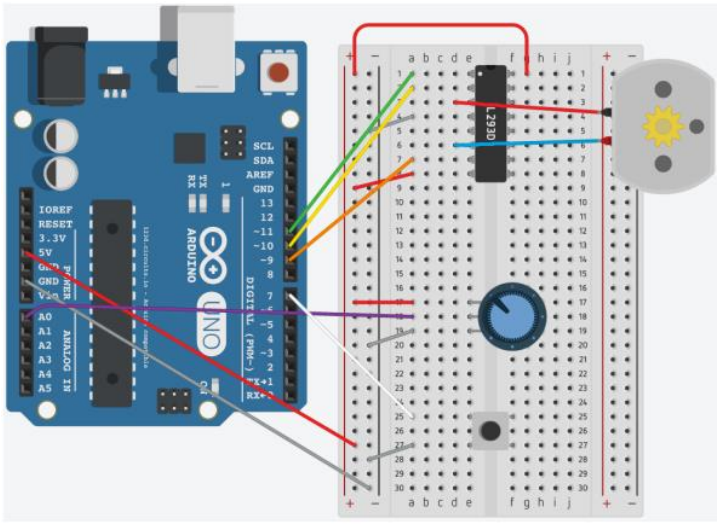
PERCOBAAN MOTOR_DC_1 : PENGATURAN KECEPATAN MOTOR

- Buat *program Motor_Dc_1* sebagaimana ditunjukkan gambar 12.3.
- Perhatikan kabel koneksi antara pin-pin mikrokontroler *Arduino* dengan pin yang ada pada *driver motor*.
- Pastikan bahwa *program* yang telah saudara buat benar.
- Hubungan pengkabelan, ditunjukkan gambar 12.4

```
int enablePin = 11;
int forwardPin = 10;
int reversePin = 9;
int buttonPin = 3;
int potPin = 0;

void setup() {
  pinMode(forwardPin,OUTPUT);
  pinMode(reversePin,OUTPUT);
  pinMode(enablePin,OUTPUT);
  pinMode(buttonPin,INPUT);
}
```

```
void loop() {
  int speed = analogRead(potPin)/4; // utk mendapatkan ra
  boolean buttonPressed = !digitalRead(buttonPin);
  setMotor(speed, buttonPressed); // kirim setting ke mot
}

void setMotor(int speed, boolean buttonPressed) {
  analogWrite(enablePin,speed);

  if(buttonPressed) {
    digitalWrite(forwardPin, HIGH);
    digitalWrite(reversePin, LOW);
  }
  else {
    digitalWrite(forwardPin, LOW);
    digitalWrite(reversePin, HIGH);
  }
}
```

Gambar 12.3. *Listing Program* Percobaan *DriverMotor*_1



Gambar 12.4. Rangkaian Pengkabelan *Motor DC_1*

Bila *push button* ditekan menyebabkan *motor* berputar berbalik arah dan bila *push button* dilepas *motor* akan berputar seperti putaran semula. Potensiometer digunakan untuk mengatur kecepatan putar dari motor.

PERCOBAAN MOTOR_DC_2 : PENGATURAN KECEPATAN MOTOR

- Buat *program Motor_Dc_2*.
- Perhatikan kabel koneksi antara pin-pin mikrokontroler *Arduino* dengan pin yang ada pada *driver motor*.

```
const int POT_input = A1;    /* assign ADC Channel */
bool d1 = HIGH;
bool d2 = LOW;

void setup() {
  pinMode(10, OUTPUT);  /* Motor control pin 1 */
  pinMode(9, OUTPUT);  /* Motor control pin 2 */
  pinMode(11, OUTPUT);  /* PWM pin for Speed Control */
  /* Interrupt pin for direction control */
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), motor, FALLING);
}

void loop() {
  int pwm_adc;
  /* Input from Potentiometer for speed control */
  pwm_adc = analogRead(POT_input);
  digitalWrite(10,d1);
  digitalWrite(9,d2);
  analogWrite(11, pwm_adc / 4);
}

void motor(){
  d1 = !d1;
  d2 = !d2;
  _delay_ms(200);
}
```

Gambar 12.5. *Listing Program* Percobaan *DriverMotor*_2

Bila *push button* ditekan menyebabkan *motor* berputar berbalik arah dan bila *push button* dilepas *motor* akan tetap berputar. Bila diinginkan perubahan putaran tinggal menekan tombol lagi dan begitu seterusnya. Potensiometer digunakan untuk mengatur kecepatan putar dari motor DC.

SOAL ESSAY

1. Buatlah program pengaturan putaran motor dc dengan memanfaatkan sensor ultrasonik dengan ketentuan :
   - jarak sensor kurang dari 6 cm maka motor tidak berputar
   - jarak sensor dari 6 sampai 12 cm motor berputar kekanan dengan duty cycle pwmnya 50 %
   - jarak sensor dari 13 sampai 20 cm motor berputar kekanan dengan duty cycle pwmnya 100 %
   - jarak sensor dari 21 sampai 30 cm motor berputar kekiri dengan duty cycle pwmnya 50 %
   - jarak sensor dari 30 sampai 40 cm motor berputar kekiri dengan duty cycle pwmnya 100 %

PELAJARAN 13
APLIKASI MOTOR SERVO

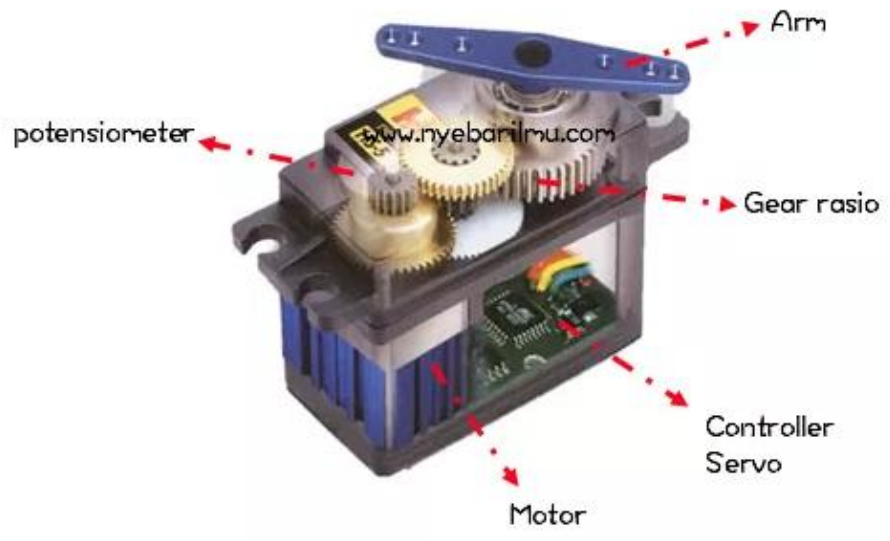
TUJUAN

- Memahami dasar-dasar pemrograman untuk menggerakkan *motor Servo.*
- Memahami pengkabelan dari motor Servo

PENGGUNAAN MOTOR SERVO

Motor Servo adalah komponen elektronika yang terdiri dari motor yang memiliki umpan balik sistem guna memberikan informasi posisi putaran motor yang diteruskan pada rangkaian kontrol mikrokontroller. Penggunaan motor servo banyak digunakan sebagai actuator yang menbutuhkan putaran motor yang presisi. Biasanya servo memiliki torsi yang besar. Motor *servo* lebih mudah untuk dikontrol sudutnya karena menggunakan *input PWM.*

Jika pada motor DC biasanya hanya mengatur antara motor DC, rasio, potensiometer serta pengontrol servo sebagaimana ditunjukkan gambar 13.1



Gambar 13.1. Komponen Utama Penyusun Motor Servo

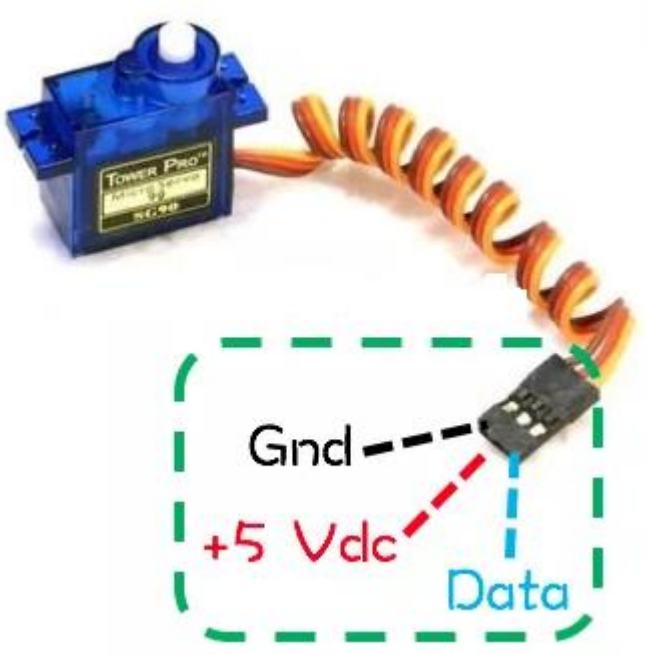
Komponen potensiometer berfungsi sebagai umpan balik nilai yang akan diolah menjadi data posisi aktual. Sedangkan fungsi dari controller servo adalah untuk memberikan sinyal-sinyal PWM agar motor dapat berputar.

Macam-macam tipe dari motor servo ada 2 yaitu tipe standard dan tipe kontinyu.

- Tipe standar berputarnya sebanyak 180^0 dan tipe ini lebih banyak digunakan pada sistem mekatronika atau robotika.
- Tipe kontinyu memiliki criteria perputaran motornya sebesar 360^0 contoh pada aplikasi mobil robot.

Secara standar, motor servo terdiri dari 3 kabel yaitu kabel :

- Power (merah)
- GND (hitam)
- Sinyal (kuning)



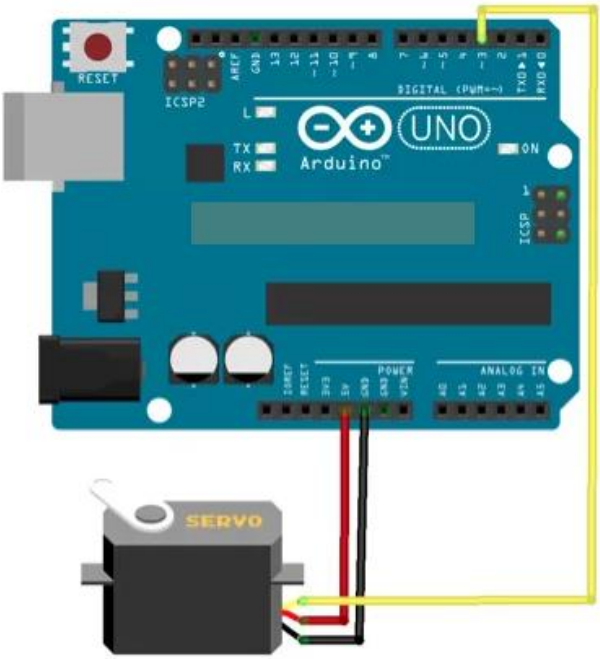
Gambar 13.2. Pewarnaan Kabel Standar Motor Servo

Spesifikasi motor servo gambar 13.2. :

- Tegangan kerja : 4,6 – 6 V dc
- Torsi : 1,6 kg / cm
- Arus : < 500 mA
- Dimensi : 22 x 12,5 x 29,5 cm
- Berat : 9 gr
- Kecepatan putaran : 0,12 detik / 60 derajat

PERCOBAAN MOTOR_SERV0_1 : PENGATURAN SUDUT

- Buat *program Motor_Servo_1* sebagaimana ditunjukkan gambar 13.4.
- Perhatikan kabel koneksi antara pin-pin mikrokontroler *Arduino* dengan pin yang ada pada *motor servo sebagaimana ditunjukkan gambar 13.3*.
- Pastikan bahwa *program* yang telah saudara buat benar.
- Gunakan tegangan 5 V untuk Supply Motor Servo



Gambar 13.3. Hubungan Motor Servo ke *Arduino Uno*

```
#include <Servo.h>
Servo motorServo;

void setup()
{
   motorServo.attach(3);
}
```

```
void loop()
{
   motorServo.write(45);  // Turn Servo ke kiri 45 degrees
   delay(500);
   motorServo.write(0);   // Turn Servo ke kiri to 0 degrees
   delay(500);
   // Turn Servo ke posisi center position (90 degrees)
   motorServo.write(90);
   delay(2000);
   motorServo.write(135); // Turn Servo Ke kanan 135 degrees
   delay(500);
   motorServo.write(180); // Turn Servo ke kanan 180 degrees
   delay(500);
   // Turn Servo ke posisi center position (90 degrees)
   motorServo.write(90);
  delay(2000);
}
```

Gambar 13.4. *Listing Program* Percobaan *MotorServo*_1

Keterangan :

- *#include<Servo.h>* digunakan untuk menyertakan library Servo pada program
- *Servo motorServo;* digunakan untuk membuat variabel *motorServo*.
- *motorServo.attach(3)* memilih atau mengatur pin 3 digital sebagai pin yang digunakan untuk *servo* pada variabel *motorServo*.
- *motorServo.write(90)* megatur posisi motor *servo* pada posisi tengah.

Program ini menggerakkan *servo* dengan *range* 0 sampai 180^0. Sebagai posisi tengah diberi sudut sebesar 90^0. Bila diinginkan *servo* bergerak kekiri sebesar 90^0 maka atur *servo* dengan sudut 0^0. Dan bila diinginkan *servo* bergerak kekanan dari titik tengan maka atur *servo* dengan sudut 180^0.

PERCOBAAN MOTOR_SERV0_2 : PENGATURAN SUDUT

- Buat *program Motor_Servo_2* sebagaimana ditunjukkan gambar 13.5.
- Perhatikan kabel koneksi antara pin-pin mikrokontroler *Arduino* dengan pin yang ada pada *motor servo* sebagaimana ditunjukkan gambar 13.3.

- Pastikan bahwa *program* yang telah saudara buat benar.
- Gunakan tegangan 5 V untuk *Supply Motor Servo*.

```
#include <Servo.h>
Servo myservo;      // variable untuk menyimpan posisi data
int pos = 00;
void setup()
{
 myservo.attach(3);   //penggunaan data pada pin 3 sebagai keluaran PWM
}

void loop()
{
 // dimulai dari nilai data 0 derajat sampai 180 derajat
 for(pos = 00; pos < 180; pos += 1)
 {
  // pada posisi 1 derajat
  myservo.write(pos); // memberitahu servo untuk pergi ke posisi 'pos'
  // waktu tunda 15ms untuk pencapaian  posisi
  delay(15);      // lambat
 }
 // mulai awal dari 180 derajat ke 0 derajat
 for(pos = 180; pos>=0; pos-=1)
 {
  // memberitahukan bahwa servo telah pindah ke posisi  'pos'
  myservo.write(pos);
  // waktu tunggu 15ms untuk pencapaian  posisi
  delay(5);      // cepat
 }
}
```

Gambar 13.5. *Listing Program* Percobaan *MotorServo_2*

Untuk program gambar 13.5 ini, sudut putaran yang dapat dituju maksimum 180^0. Dengan menggunakan *servo* ini, tidak lagi membicarakan putaran searah atau berlawanan jarum jam tetapi bicara tentang sudut yaitu 0^0, 45^0, 90^0 dan seterusnya sampai 180^0.

Bila diinginkan untuk memperoleh sudut 90^0 dan bergerak berlawanan jarum jam maka ditulis perintah *myservo.write(90);*. Setelah itu ditulis *myservo.write(0);*.

Akan tetapi jika ingin mendapatkan posisi sudut 90^0 dan searah jarum jam maka dituliskan perintah *myservo.write(90);*. Setelah itu dituliskan perintah *myservo(180);*.

Jadi posisi 0 sampai dengan 180 derajat sudah ditentukan oleh kontroller internal motor *servo*, dan cukup dengan memberikan perintah pada sudut mana motor akan memutar melalui perintah *myservo.write(detajat)*.

PERCOBAAN MOTOR_SERV0_3 : PENGATURAN SUDUT

- Buat *program Motor_Servo_6* sebagaimana ditunjukkan gambar 13.6.
- Perhatikan kabel koneksi antara pin-pin mikrokontroler *Arduino* dengan pin yang ada pada *motor servo sebagaimana ditunjukkan gambar 12.3*.
- Pastikan bahwa *program* yang telah saudara buat benar.
- Gunakan tegangan 5 V untuk Supply Motor Servo

```
#include <Servo.h>
Servo myservo;
const int button1 = 2;
const int button2 = 3;
const int button3 = 4;
const int button4 = 5;
const int button5 = 6;

int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;
int buttonState5 = 0;
int target=0;
int sekarang=0;

void setup() {
  myservo.attach(9);
  myservo.write(90);
  pinMode(button1, INPUT);
  pinMode(button2, INPUT);
  pinMode(button3, INPUT);
  pinMode(button4, INPUT);
  pinMode(button5, INPUT);
}
```

```
void loop() {
  buttonState1 = digitalRead(button1);
  buttonState2 = digitalRead(button2);
  buttonState3 = digitalRead(button3);
  buttonState4 = digitalRead(button4);
  buttonState5 = digitalRead(button5);

  if(buttonState1 == HIGH) target=0;
  if(buttonState2 == HIGH) target=45;
  if(buttonState3 == HIGH) target=90;
  if(buttonState4 == HIGH) target=135;
  if(buttonState5 == HIGH) target=180;

  if(target>sekarang){
      for(int i=sekarang;i<=target;i++){
      myservo.write(i);
      delay(5);}
      sekarang=target;}

   if(sekarang>target){
      for(int i=sekarang;i>=target;i--){
      myservo.write(i);
      delay(5);}
      sekarang=target;}
}
```

Gambar 13.6. *Listing Program* Percobaan *MotorServo*_3

PERCOBAAN MOTOR_SERV0_4 : PENGATURAN SUDUT

- Buat *program Motor_Servo_4* sebagaimana ditunjukkan gambar 13.7.
- Perhatikan kabel koneksi antara pin-pin mikrokontroler *Arduino* dengan pin yang ada pada *motor servo sebagaimana ditunjukkan gambar 13.3*.
- Pastikan bahwa *program* yang telah saudara buat benar.
- Gunakan tegangan 5 V untuk Supply Motor Servo

```
#include <Servo.h>
Servo myservo;
const int button1 = 2;
const int button2 = 3;
const int button3 = 4;
const int button4 = 5;
const int button5 = 6;

int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;
int buttonState5 = 0;
int target=0;
int sekarang=0;

void setup() {
  myservo.attach(9);
  myservo.write(90);
  pinMode(button1, INPUT);
  pinMode(button2, INPUT);
  pinMode(button3, INPUT);
  pinMode(button4, INPUT);
  pinMode(button5, INPUT);
}
void loop() {
  buttonState1 = digitalRead(button1);
  buttonState2 = digitalRead(button2);
  buttonState3 = digitalRead(button3);
  buttonState4 = digitalRead(button4);
  buttonState5 = digitalRead(button5);
  if(buttonState1 == HIGH) target=0;
  if(buttonState2 == HIGH) target=45;
  if(buttonState3 == HIGH) target=90;
  if(buttonState4 == HIGH) target=135;
  if(buttonState5 == HIGH) target=180;

  if(target>sekarang){
      for(int i=sekarang;i<=target;i++){
      myservo.write(i);
      delay(5);}
      sekarang=target;}
```

```
    if(sekarang>target){
        for(int i=sekarang;i>=target;i--){
        myservo.write(i);
        delay(5);}
        sekarang=target;}
}
```

Gambar 13.7. *Listing Program* Percobaan *MotorServo*_4

DAFTAR PUSTAKA

1. Ardianto Heri & Darmawan Aan, 2016, "*ARDUINO* Belajar Cepat dan Pemograman", Informatika Bandung.
2. …, 2016,"ARDUINO TUTORIAL", Tutorial Point(I) Pvt.Ltd.
3. Djuandi Feri, 2011, "Pengenalan *Arduino*".
4. M.D. Yuwono, 2016, " *Arduino* itu Pintar", PT Elex Media Komputindo, Jakarta.
5. Saftari Firmansyah, 2015, " Proyek Robotika Keren dengan *Arduino*", PT Elex Media Komputindo, Jakarta
6. Setiawan Deny, 2008, "*Arduino Uno*", Ilmu Teknologi Informasi.
7. 2016, "*ARDUINO* Tutorial Points Simply Easy Learning", Pvt. Ltd, *www.tutorialspoint.com*
8. *https://www.arduino.cc*