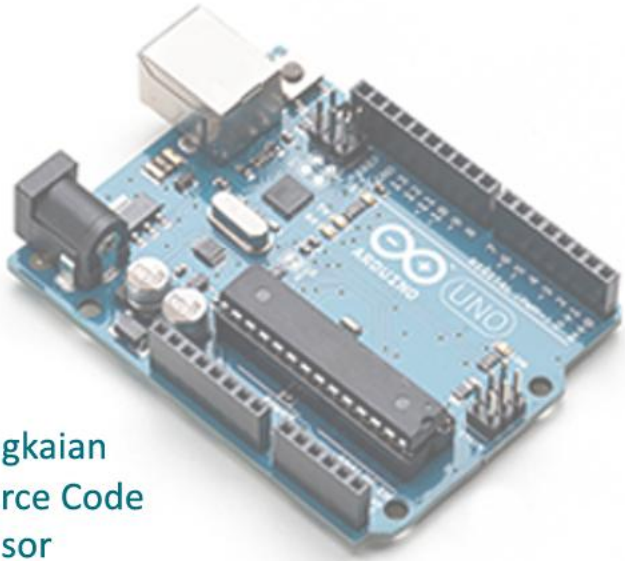
ebook gratis

Panduan Praktis

Arduino untuk Pemula



Rangkaian
Source Code
Sensor
Teori Pendukung

Hack Your Skills!

Hari Santoso
www.elangsakti.com

V.1

Panduan Praktis

Arduino untuk Pemula

Hari Santoso

www.elangsakti.com

Juli 2015

Direvisi, Juni 2016

Kami menerima jasa pembuatan program berbasis arduino untuk berbagai keperluan. Untuk informasi lebih lanjut, silakan klik http://hire.elangsakti.com/ dan isi form penawaran yang kami sediakan.
Lebih cepat lebih baik. ☺

Catatan Pembuka

Ebook ini terinspirasi dari buku *Introduction to Arduino* karangan Alan G. Smith. Oleh sebab itu, sebagian isi dari ebook ini mengikuti alur pembahasan dari buku tersebut, tapi dengan berbagai improvisasi baik dari segi rangkaian dan program. Gambar rangkaian pada ebook ini dibuat dengan aplikasi Fritzing.org dan SnagIt. Arduino yang digunakan dalam ebook ini adalah **Arduino Uno.**

Persembahan

Puji dan syukur kepada Allah SWT yang masih memberi kesempatan hidup dan waktu luang sehingga penulisan buku ini selesai lebih cepat dari yang diperkirakan.

Salawat dan salam semoga tetap tercurahkan kepada Rasulullah dan keluarga Belia, kepada para Nabi, keluarga, dan para penerusnya.

Secara spesial, kasih dan sayang penulis untuk istri tercinta (Aprillia D. Kreswanti) yang telah merelakan waktu, pikiran, dan tenaganya dalam mendukung penulisan buku ini.

Terima kasih pula untuk rekan-rekan SKI-C2 Brawijaya angkatan 2014 serta semua pihak yang secara tidak langsung "*dikatutkan*" dalam skenario Allah sehingga penulis benar-benar terjerumus ke dalam dunia elektronika. :p

Tentunya buku ini masih belum sempurna baik dari segi tata bahasa dan penyampaiannya. Semua kritik dan saran silakan kirimkan ke penulis melalui **hari/at/elangsakti.com**.

Salam,

Trenggalek, 15 Juli 2015

Untukmu

Generasi Muda Indonesia

Sebelumnya, saya bersyukur kepada Allah SWT yang telah memelihara otak dan tubuh ini, menjaga dan membuatnya mudah dalam mempelajari sebagian ciptaannya. Bersyukur karena masih diberikan kesempatan untuk berbagi tentang apa yang penulis bisa, untuk negeri ini, untuk kalian semua.

Ebook ini dipersembahkan untuk generasi muda Indonesia. Generasi yang akan mewarnai Indonesia 3–10 tahun mendatang dengan teknologi dan karya terbaik mereka. Karena salah satu faktor kemajuan suatu negeri ditentukan dengan teknologi yang berkembang di masyarakat negeri tersebut.

Ebook ini dipersembahkan untuk para pendidik, siswa, dan mahasiswa yang berkecimpung dalam elektronika dan komputer. Sehingga mereka bisa membuat prototype dari teknologi impian yang ingin mereka buat. Diharapkan akan muncul ide-ide brilian sebagai solusi dari masalah-masalah kehidupan sehari-hari yang kita alami.

Ebook ini ditujukan bagi mereka yang ingin mulai belajar tentang robotika, membuat mesin-mesin otomatis, dan yang ingin mendalami Internet of Things (IoT). Arduino bisa dikembangkan dan dipadukan dengan berbagai sensor dan kecerdasan buatan, mengintegrasikan dengan website, dan banyak hal lainnya yang memungkinkan untuk membuat rumah cerdas (*smart house*), *smart gardening*, *smart farming*, hingga *smart city*.

Daftar Isi

Daftar Rangkaian

Daftar Program

Daftar Gambar

Bagian 1

Pengenalan Arduino

Ebook ini dimaksudkan agar Anda yang masih pemula dalam dunia mikrokontroller dapat mengikuti dan mempelajari Arduino dengan mudah dan segera mempraktekkannya. Oleh sabab itu, kami akan membahas tentang konsep elektronik, sensor, dan bahasa pemrograman secukupnya dengan harapan Anda bisa segera praktek tanpa memikirkan konsep elektronika yang relatif rumit.

Dalam setiap bahasan, kami Akan menyajikan konsep singkat dari apa yang sedang dibahas dalam bahasan tersebut. Misal, ketika kita membahas tentang rangkaian sensor suhu, maka akan dibahas pula tentang komponen sensor suhu, cara kerja, cara merangkai, dan cara memprogramnya. Hal tersebut dimaksudkan agar pembaca bisa dengan mudah mengikuti dan memahami apa yang kami sajikan serta sedikit demi sedikit mempelajari tentang elektronika (mikrokontroller) dan bahasa pemrograman.

1.1 Apa itu Mikrokontroller?

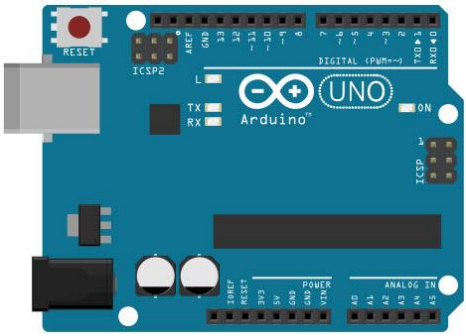
Menurut wikipedia[1]:

> *A microcontroller (sometimes abbreviated μC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.*

Dalam diskusi sehari-hari dan di forum internet, mikrokontroller sering dikenal dengan sebut μC, uC, atau MCU. Terjemahan bebas dari pengertian tersebut, bisa dikatakan bahwa mikrokontroller adalah komputer yang berukuran mikro dalam satu chip IC (integrated circuit) yang terdiri dari processor, memory, dan antarmuka yang bisa diprogram. Jadi disebut komputer mikro karena dalam IC atau chip mikrokontroller terdiri dari CPU, memory, dan I/O yang bisa kita kontrol dengan memprogramnya. I/O juga sering disebut dengan

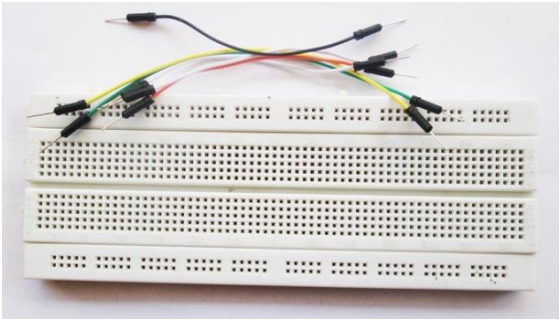
[1] *https://en.wikipedia.org/wiki/Microcontroller*
[2] *https://www.arduino.cc/en/Main/Software*

GPIO (*General Purpose Input Output Pins*) yang berarti : pin yang bisa kita program sebagai input atau output sesuai kebutuhan.



Gambar 1.1 Board Arduino Uno

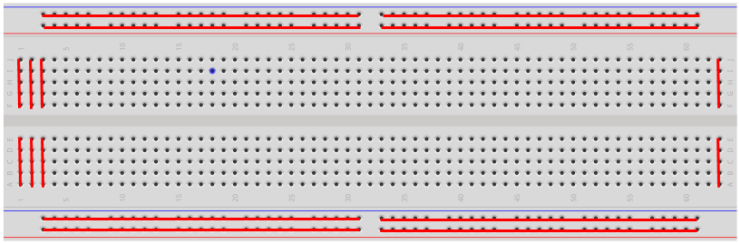
Dalam ebook ini kita akan menggunakan *board* Arduino Uno (Gambar 1.1). Board Arduino terdiri dari hardware / modul mikrokontroller yang siap pakai dan *software* IDE yang digunakan untuk memprogram sehingga kita bisa belajar dengan mudah. Kelebihan dari Arduino yaitu kita tidak direpotkan dengan rangkaian minimum sistem dan *programmer* karena sudah *built in* dalam satu *board*. Oleh sebab itu kita bisa fokus ke pengembangan sistem.



Gambar 1.2 Project board dan Kabel Jumper

Untuk praktek, kita akan menggunakan *project board* (ada yang menyebutnya dengan istilah *bread board*) dan beberapa kabel jumper untuk menghubungkan antara komponen dan Arduino (Gambar 1.2). Dengan *project board* kita tidak perlu menyolder rangkaian sehingga relatif mudah dan cepat dalam merangkai. *Project board* memungkinkan kita untuk membangun dan membongkar rangkaian dengan cepat

sehingga sangat cocok untuk eksperimen. Tapi jika kita ingin membuat rangkaian yang permanen, maka kita harus menggunakan PCB.



Gambar 1.3 Peta jalur pada project board

Yang terpenting adalah, kita harus memahami jalur-jalur pada *project board*. *Project board* terdiri dari jalur vertikal dan jalur horisontal. Jalur vertikal ada di bagian tengah yang terdiri dari 2 x 64 jalur. Masing-masing jalur terdiri dari 5 titik vertikal, misal jalur 1A-1B-1C-1D-1E dan jalur 1F-1G-1H-1I-1J yang kedua tidak saling tersambung. Jalur horisontal sebanyak 8 jalur, 4 jalur ada di bagian atas dan 4 jalur lagi di bagian bawah. Jalur ini bisa digunakan untuk power supply (VCC dan GND) untuk rangkaian. Untuk lebih jelasnya, silakan perhatikan Gambar 1.3. Garis **merah** menunjukkan bahwa lubang tersebut terhubung secara fisik.

1.2 Instalasi Arduino IDE

Anda bisa mendownload Arduino IDE di website Arduino[2]. Pada saat tulisan ini dibuat (30/06/2015), Arduino IDE sudah versi 1.6.5. Software Arduino ada yang versi installer (hanya untuk Windows) dan versi terkompres dalam zip. Jika memilih versi tanpa install (format .zip), maka Anda hanya perlu mengekstraknya di folder mana saja dan Anda bisa langsung menjalankannya.

Jika Anda pengguna Linux, maka sedikit tantangan untuk Anda karena proses instalasi tidak semudah instalasi di Windows dan Mac. Panduan untuk menginstall di Linux bisa Anda pelajari di bagian

[2] *https://www.arduino.cc/en/Main/Software*

instalasi Linux[3]. Sedangkan untuk pengguna Windows dan Mac, Anda bisa menginstall dengan mengikuti instruksi dalam ebook ini.

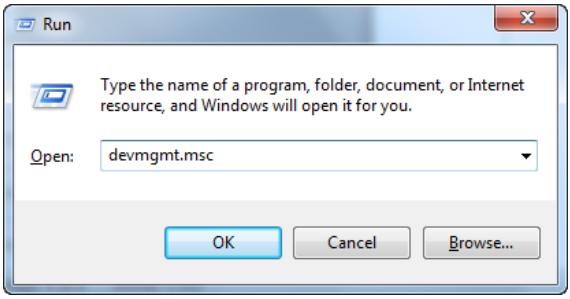
1.2.1 Instalasi di Windows

1. Pasang board Arduino Anda ke port USB pada komputer atau laptop, kemudian tunggu hingga Windows mencoba untuk menginstall driver sendiri. Biasanya dia gagal menginstall *driver* jika belum memiliki *driver* tersebut. (Silakan lanjutkan ke step berikutnya)
2. Jika berhasil, berarti instalasi selesai. Tapi jika gagal, lanjutkan ke step selanjutnya.
3. Anda harus install dari *device manager*. Untuk masuk ke *device manager*, Anda bisa melakukannya dengan dua cara:

Gambar 1.4 Posisi tombol Windows

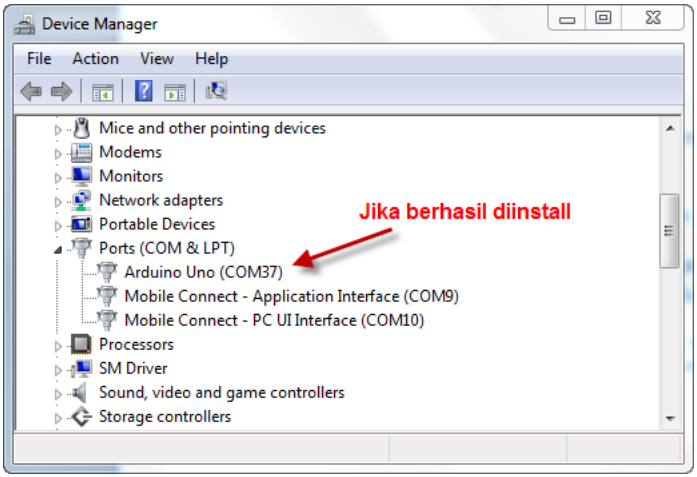
- Tekan tombol ("Windows" + R) secara bersamaan. Tombol "Windows" adalah tombol pada keyboard dengan logo Windows (gambar logo windows, biasanya terletak di sebelah kiri atau kanan spasi, lihat Gambar 1.4). Setelah Anda menekan tombol "Windows" + R, maka akan muncul "Run", ketikkan "*devmgmt.msc*" (tanpa tanda petik), kemudian tekan tombol ENTER. Jika benar, maka akan muncul window *Device Manager*.

[3] *http://playground.arduino.cc/Learning/Linux*



Gambar 1.5 Window yang muncul setelah menekan (Windows + R)

- Jika *Device Manager* Anda sudah keluar, Anda bisa lanjut ke point 4, jika tidak, coba cara berikut untuk menampilkan *device manager*
- Klik Start - pilih *Control Panel*. Di dalam Control Panel, pilih *System and Security*, lalu pilih *System*. Selanjutnya pilih *Device Manager*.



Gambar 1.6 Tampilan Device Manager

4. Pada *Device Manager*, perhatikan bagian Ports (COM & LPT), akan muncul device baru dengan nama "Arduino UNO (COMxx)"
5. Klik kanan pada "Arduino UNO (COMxx)", kemudian pilih "*Update Driver Software*".
6. Selanjutnya pilih "*Browse my computer for Driver software*".

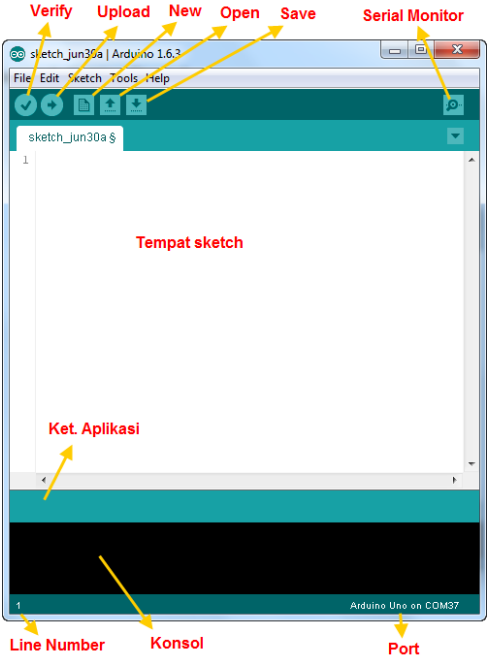
7. Cara folder software Arduino Anda, kemudian cari file *arduino.inf* (khusus untuk Arduino UNO REF.3) pada folder Drivers. Jika Anda menggunakan versi IDE di bawah 1.0.3, Anda bisa memilih *driver* dengan nama file *ArduinoUNO.inf*
8. Jika berhasil, berarti instalasi *driver* sudah selesai. Jika belum, silakan Anda tanya-tanya di web *http://www.elangsakti.com*
9. Selanjut mari kita coba untuk mengupload sampel code yang ada pada *software* Arduino
10. Jalankan Aplikasi Arduino (arduino.exe), pada pojok kanan bawah akan ada tulisan "Arduino UNO on COMxx". Berarti port yang digunakan Arduino adalah COMxx, jika tulisan tersebut tidak muncul, berarti instalasi *driver* belum berhasil atau board Arduino belum disambungkan ke komputer. Selanjutnya, silakan buka sampel led flip-flop dengan cara Klik menu *File > Examples > 1.Basic > Blink*
11. Setting board Arduino dengan cara : Klik menu *Tools > Board > Arduino UNO*
12. Pilih port yang digunakan Arduino dengan cara mengklik menu *Tools > Ports > (pilih yang ada Arduino-nya)*
13. Klik tombol upload (tombol denga panah ke kanan)
14. Setelah berhasil diupload, akan muncul tulisan "*Done uploading*" di bagian bawah. Jika berhasil, maka LED dengan tulisan "L" pada board Arduino akan kelap-kelip.

1.2.2 Instalasi pada Mac

1. Pasang *board* Arduino melalui USB
2. Ekstrak aplikasi ke hardisk
3. Ketika muncul *Network Preferences*, klik "*Apply*" (ingat /dev/tty/usb)
4. Jalankan aplikasi
5. Bukan sampel aplikasi pada menu *File > Example > 1.Basics > Blink*
6. Pilih Arduino Uno pada menu *Tools > Board*
7. Pilih Port yang yang digunakan Arduino pada menu *Tools > Ports*
8. Klik tombol upload (tombol dengan panah ke kanan)
9. Setelah muncul pesan "*Done Uploading*", maka led "L" pada board Arduino akan berkedip.

1.3 Arduino IDE

Untuk memprogram board Arduino, kita butuh aplikasi IDE (*Integrated Development Environment*) bawaan dari Arduino. Aplikasi ini berguna untuk membuat, membuka, dan mengedit *source code* Arduino (*Sketches*, para *programmer* menyebut *source code* arduino dengan istilah "*sketches*"). Selanjutnya, jika kita menyebut *source code* yang ditulis untuk Arduino, kita sebut "*sketch*" juga ya :). Sketch merupakan *source code* yang berisi logika dan algoritma yang akan diupload ke dalam IC mikrokontroller (Arduino).



Gambar 1.7 Interface Arduino IDE

Interface Arduino IDE tampak seperti gambar 1.7. Dari kiri ke kanan dan atas ke bawah, bagian-bagian IDE Arduino terdiri dari:

- ✓ **Verify** : pada versi sebelumnya dikenal dengan istilah *Compile*. Sebelum aplikasi diupload ke *board* Arduino, biasakan untuk memverifikasi terlebih dahulu *sketch* yang dibuat. Jika ada kesalahan pada *sketch*, nanti akan muncul error. Proses Verify / *Compile* mengubah *sketch* ke *binary code* untuk diupload ke mikrokontroller.

- ✓ **Upload** : tombol ini berfungsi untuk mengupload *sketch* ke *board* Arduino. Walaupun kita tidak mengklik tombol *verify*, maka *sketch* akan di-*compile*, kemudian langsung diupload ke *board*. Berbeda dengan tombol *verify* yang hanya berfungsi untuk memverifikasi *source code* saja.
- ✓ **New Sketch** : Membuka window dan membuat *sketch* baru
- ✓ **Open Sketch** : Membuka *sketch* yang sudah pernah dibuat. *Sketch* yang dibuat dengan IDE Arduino akan disimpan dengan ekstensi file **.ino**
- ✓ **Save Sketch** : menyimpan *sketch*, tapi tidak disertai mengcompile.
- ✓ **Serial Monitor** : Membuka *interface* untuk komunikasi serial, nanti akan kita diskusikan lebih lanjut pada bagian selanjutnya
- ✓ **Keterangan Aplikasi** : pesan-pesan yang dilakukan aplikasi akan muncul di sini, misal "*Compiling*" dan "*Done Uploading*" ketika kita mengcompile dan mengupload *sketch* ke *board* Arduino
- ✓ **Konsol** : Pesan-pesan yang dikerjakan aplikasi dan pesan-pesan tentang *sketch* akan muncul pada bagian ini. Misal, ketika aplikasi mengcompile atau ketika ada kesalahan pada *sketch* yang kita buat, maka informasi *error* dan baris akan diinformasikan di bagian ini.
- ✓ **Baris Sketch** : bagian ini akan menunjukkan posisi baris kursor yang sedang aktif pada *sketch*.
- ✓ **Informasi Port** : bagian ini menginformasikan *port* yang dipakah oleh *board* Arduino.

1.4 Rangkaian LED Pertama

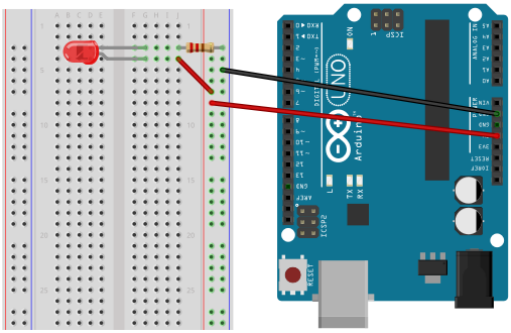
Ketika belajar pemrogaman, program pertama yang harus dicoba pertama kali adalah memunculkan pesan "*Hello World!*". Dalam belalajar mikrokontroller ternyata juga ada, yang pertama kali harus dibuat adalah membuat lampu LED berkedip, LED berkedip maksudnya adalah *flip-flop*, hehe. ☺

LED merupakan kependekan dari *Light Emiting Diode*, yaitu diode yang mampu mengubah listrik menjadi cahaya. Sebagaimana sifat diode, lampu LED memiliki kaki positif dan negatif. Sehingga

pemasangannya tidak boleh terbaik, jika dipasang terbalik maka tidak akan ada arus yang mengalir dan LED pun tidak akan menyala.

Arduino bekerja pada tegangan 5-12 volt dengan arus yang relatif besar yang sanggup memutuskan LED. Sehingga jika kita ingin menyambungkan LED, maka kita butuh tahanan (resistor) untuk membatasi arus yang masuk ke LED. LED memiliki tegangan kerja yang disebut dengan *forward voltage* (*fv*) yang mana tegangan ini adalah tegangan yang dibutuhkan LED untuk bisa menyala dengan baik.

Ukuran resistor yang bisa dipakai adalah 100Ω hingga 1KΩ (Ω dibaca ohm, satuan dari resistansi/hambatan), makin besar nilai resistor maka nyala LED akan semakin redup. Pada Arduino, tegangan yang keluar dari pin-pinnya adalah 0-5 volt. Sementara catu daya untuk Arduino antara 5-12 volt. Oleh sebab itu, pemilihan resistor tergantung tegangan mana yang akan kita gunakan.



Rangkaian 1.1 Percobaan LED

Silakan buatlah Rangkaian 1.1 dengan langkah-langkah sebagai berikut:

1. Pasang kaki positif LED di G4 dan kaki negatifnya di G3,
2. Setelah itu, hubungkan salah satu kaki resistor pada lubang J3, kemudian kaki satunya ke lubang di kolom pertama dari kanan,
3. Ambil kabel *jumper*, lalu sambungkan lubang J4 ke lubang di kolom kedua dari kanan,
4. Ambil kabel *jumper*, sambungkan salah satu lubang di kolom pertama ke *socket* GND pada board Arduino,

5. Ambil kabel *jumper*, sambungkan salah satu lubang di kolom kedua ke *socket* 5V di *board* Arduino
6. Hubungkan USB Arduino ke komputer/laptop.
7. Selamat! Anda sudah membuat rangkaian pertama Anda dengan Arduino. ☺

Jika Anda memasangnya dengan benar, maka LED akan menyala. Jika tidak, mungkin kaki dipasang terbalik, silakan diperbaiki.

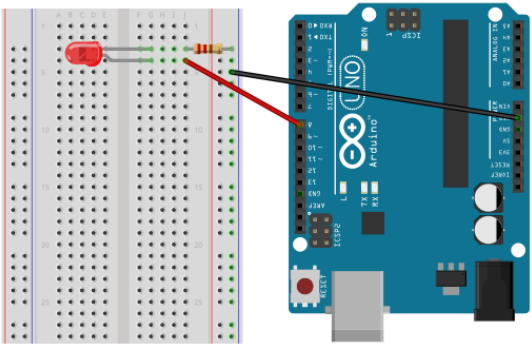
1.6 Program Pertama Anda

Program pertama Anda adalah membuat kedipan LED. Untuk itu, yang pertama harus Anda lakukan adalah mengubah Rangkaian 1.1, setelah itu Anda bisa membuat program pada Arduino IDE. Logika untuk program pertama seperti berikut:

> *Kedipan LED (hidup-matinya LED) akan dikontrol oleh salah satu kaki Arduino (kita pilih kaki/pin 8). Dengan demikian, logika pada pin 8 akan menjadi* ***output*** *untuk mengontrol LED. Ingat, logika 1 berararti LED akan nyala, logika 0 berarti LED mati.*

1.6.1 Update Rangkaian Anda

Rubahlah rangkaian Anda sehingga tampak seperti Rangkaian 1.2. Kabel jumper dari 5V ke kolom pertama dilepas. Kemudian kabel di kolom kedua dilepas dan disambungkan ke pin 8 pada board Arduino.



Rangkaian 1.2 Rangkaian LED Berkedip

1.6.2 Program untuk LED Berkedip

Ketika Anda pertama kali membuka Arduino IDE, maka secara otomatis akan muncul sketch seperti berikut:

Sketch 1.1 Sketch minimal Arduino

```
1  void setup() {
2    // put your setup code here, to run once:
3
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8
9  }
```

Fungsi *setup()* dan fungsi *loop()* merupakan fungsi wajib dan harus ada. Fungsi *setup()* digunakan untuk inisialisasi program, fungsi ini hanya dijalankan sekali yaitu ketika program pertama kali dijalankan (ketika arduino pertama kali dihidupkan). Sedangkan fungsi *loop()* akan dijalankan terus-menerus (*looping forever*) hingga Arduino dimatikan.

Program di atas sudah bisa diupload ke Arduino dengan cara meng-klik tombol Upload. Hanya saja, ketika program tersebut diupload, Arduino tidak akan melakukan apa-apa sebab dalam *sketch* tersebut memang tidak ada perintah yang harus dikerjakan.

Dalam ebook ini, jika ada *sketch* yang harus Anda coba, maka kami akan menampilkan *sketch* terlebih dahulu, setelah itu akan kami jelaskan fungsi dan logika dari *sketch* tersebut. Untuk mengawali primordial dalam belajar mikrokontroller, maka Anda harus mencoba *sketch* ini. Berikut ini adalah *sketch* untuk membuat kedipan LED.

Sketch 1.2 LED Berkedip :p

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  const int pinLED = 8;
6
7  void setup() {
8     pinMode(pinLED, OUTPUT);
9  }
10
```

```
11  void loop() {
12    digitalWrite(pinLED, HIGH);
13    delay(500);
14    digitalWrite(pinLED, LOW);
15    delay(500);
16  }
```

Baik, mari kita ulas satu-persatu baris-baris sketch 1.2.

```
5  const int pinLED = 8;
```

Nama-nama pin pada Arduino sama seperti yang tertera di *board*. Pada salah satu sisi *board*, nama-nama pinnya adalah 0 hingga 13, kemudian di sisi lain nama-nama pinnya A0 hingga A5, dst. Perintah pada baris 5 artinya : Variabel *pinLED* merupakan konstanta dalam bentuk *integer* yang merujuk pada pin 8 *board* Arduino.

Tolong diingat baik-baik, untuk kemudahan dalam memprogram, sebaiknya inisialisasi pin-pin dijadikan konstanta dan ditentukan di awal program. Sehingga, misal ketika kita ingin mengubah pin yang akan dirujuk, kita tidak akan kesusahan. Kita tinggal mengubah nilai dari variabel pin tersebut, maka kita tidak perlu langi mengubah variabel lainnya.

```
7  void setup() {
8    pinMode(pinLED, OUTPUT);
9  }
```

Fungsi *pinMode()* memberi tahu bahwa *pinLED* adalah *Output*. Dengan demikian mikrokontroller tidak akan "membaca" logika pin tersebut, akan tetapi dia hanya akan "menulis" logika pada pin tersebut. Dengan kata lain, jika kita ingin mendefinisikan bahwa pin ini adalah input, maka kita tinggal mengubah *OUTPUT* menjadi *INPUT*.

```
11  void loop() {
12    digitalWrite(pinLED, HIGH);
13    delay(500);
14    digitalWrite(pinLED, LOW);
15    delay(500);
16  }
```

Baris ini adalah inti dari program yang akan dieksekusi selama Arduino tersambung dengan listrik atau selama Arduino tidak direset. pinLED diset HIGH berarti LED akan diberi tegangan 5 volt, sedangkan LOW berarti LED akan diberi tegangan 0 volt. Oleh sebab itu, rangkaian LED di atas akan menyala ketika diberi HIGH dan akan mati ketika diberi LOW.

Fungsi *delay()* digunakan untuk berhenti selama sekian milidetik. Karena 1 detik = 1000 milidetik, maka pemberian nilai 500 berarti Arduino akan jeda selama ½ detik ketika LED nyala dan ½ detik ketika LED padam. Lalu bagaimana jika program yang Anda coba tidak berjalan dan *error*? Begini, ada beberapa yang perlu diperhatikan ketika Anda memprogram:

1. Penulisan *sketch* itu *case sensitive*, artinya, tulisan "pinLED" tidak sama dengan "PinLED". Jika terjadi *error*, coba perhatikan apakah ada penulisan yang keliru?
2. Jika Anda *copy-paste sketch* dari file PDF ke Arduino IDE, maka kemungkinan akan ada perubahan *whitespace* (spasi, tab, blank line). Jika spasi tergantikan denan karakter tab atau blank line, maka sketch akan *error*. Silakan Anda cek setiap spasi, jika ukuran *space*-nya berbeda dengan spasi yang lain, silakan dihapus dan ganti dengan spasi.
3. Setiap blok kode harus diapit dengan kurung kurawal '{' dan '}'. Jika kurungnya kurang satu, maka akan *error*.
4. Setiap ada kurung buka '(', harus ada kurung tutup ')'. Jadi jika ternyata kurungnya kurang, itu bisa menyebabkan *error*.
5. Penulisan angka tanpa embel-embel koma. Misal Anda ingin menulis 1000, maka tidak perlu menulis dengan 1,000 atau 1.000. Penulisan dengan 1,000 akan terjadi error, sedangkan jika Anda menulis dengan 1.000 akan dianggap 1, bukan 1000.
6. Setiap baris kode akan ditutup dengan titik koma (*semicolon*) ';', kecuali di akhir blokkode yang ditutup dengan kurung kurawal '}'.

1.5 Menambah Keterangan pada Sketch

Dalam membuat kode program, kita kadang butuh untuk menyisipkan keterangan tambahan pada fungsi kode tersebut. Misal, keterangan fungsi, keterangan pin, atau keterangan lainnya.

Pada Arduino, komentar bisa kita sisipkan dengan dua cara:

1. Menggunakan *double slash "//"*, cara ini hanya bisa digunakan perbaris. Jadi jika banyak baris yang ingin kita jadikan komentar, Anda bisa menggunakan cara yang kedua. Contoh:

```
1  // ini komentar
2  // ini komentar lagi
3  // ini juga komentar
```

2. Menggunakan pembuka "/*" dan penutup "*/"

```
1  /* ini komentar
2     ini masil komentar
3     ini juga komentar */
4
5  /* ini komentar */
```

Jadi, jika *sketch* kedipan led tadi kita tambahin keterangan untuk setiap baris kodenya, maka setidaknya akan seperti ini.

Sketch 1.3 Contoh Sketch dengan komentar

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  /*
6   Program untuk membuat LED berkedip
7   ½ detik nyala ½ detik mati
8   */
9
10 // inisialisasi pin untuk mengontrol LED, yaitu pin 8
11 const int pinLED = 8;
12
13 /*
14  Bagian ini akan dieksekusi sekali ketika Arduino pertama kali
15  Dinyalakan atau ketika Arduino direset
16  */
17 void setup() {
18    // pin 8 diset sebagai OUTPUT
19    pinMode(pinLED, OUTPUT);
20 }
21
22 /*
23  Bagian ini akan dieksekusi selama listrik tersambung
```

```
 Atau arduino tidak direset
 */
void loop() {
   // nyalakan LED
   digitalWrite(pinLED, HIGH);
   // delay 500 milisekon / ½ menit
   delay(500);
   // matikan LED
   digitalWrite(pinLED, LOW);
   // delay selama 500 milisekon / ½ menit
   delay(500);
}
```

Demikian untuk pengenalan singkat tentang Arduino, merakit komponen dengan peoject board, dan membuat program sederhana LED berkedip. Silakan lakukan improvisasi rangkaian tersebut sehingga lebih paham dan lebih nyaman menggunakan *project board*. Selanjutnya, kita akan lebih mendalami beberapa perintah dasar dan fungsi-fungsi logika untuk membuat program Arduino.

Salam semangat! ☺

Pemuda yang tidak bercita-cita bukan pemuda, tetapi pemuda yang sudah mati sebelum mati.

(Ir. Soekarno)

Bagian 2

Animasi LED

Sebelumnya kita sudah belajar tentang membuat LED berkedip, saat ini kita akan lebih memahami lebih dekat tentang bahasa pemrograman dan fungsi-fungsi logika yang umum dipakai. Implementasi dari pembelajaran fungsi logika tersebut akan diterapkan pada beberapa model Animasi LED seperti running LED atau yang lainnya.

2.1 Perintah IF dan IF - ELSE

Perintah IF memiliki beberapa kombinasi, bisa IF saja, IF-ELSE, IF-ELSE IF-ELSE, dan seterusnya. Semakin kompleks tentu logika yang dipakai akan tampak semakin "rumit". ☺

2.1.1 Perintah IF

Mari kita modifikasi program Sketch 1.1 menjadi seperti berikut:

Sketch 2.1 Modifikasi timeDelay

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // Pin 8 untuk LED
6  const int pinLED = 8;
7
8  void setup() {
9    // pin LED sebagai output
10   pinMode(pinLED, OUTPUT);
11 }
12
13 // awal time delay 1000 | 1 detik
14 int timeDelay = 1000;
15
16 void loop() {
17   // Setiap looping, nilai timeDelay dikurangi 100
18   timeDelay = timeDelay - 100;
```

```
19
20      /* Jika timeDelay bernilai 0 atau negatif
21         maka nilai timeDelay direset ke 1000
22       */
23      if(timeDelay <= 0){
24        timeDelay = 1000;
25      }
26
27      //Nyalakan dan matikan LED selama timeDelay
28      digitalWrite(pinLED, HIGH);
29      delay(timeDelay);
30      digitalWrite(pinLED, LOW);
31      delay(timeDelay);
32  }
```

Apakah Anda bisa membayangkan apa yang akan terjadi dengan LED tersebut? LED tersebut awalnya akan berkedip pelan, lama-lama akan berkedip cepat, dan akan akhirnya akan berkedip pelan lagi. Percayakah Anda? Jika tidak, silakan dicoba. ☺

Ketika awal dinyalakan, maka *timeDelay* adalah 1000. Nilai tersebut diinisialisasi pada baris 14. Baris ini tidak dijadikan konstanta (*const*) sebagaimana *pinLED* karena nilai *timeDelay* akan diubah-ubah.

```
14      int timeDelay = 1000;
```

Setelah masuk ke bagian utama aplikasi, pada baris 18 nilai timeDelay dikurangi 100.

```
18      timeDelay = timeDelay - 100;
```

Sehingga *timeDelay* pertama yang digunakan untuk menyalakan dan mematikan LED adalah 900. Pada perulangan selanjutnya, *timeDelay* kembali dikurangi 100, sehingga perulangan ke dua menggunakan *timeDelay* 800 (lebih singkat dari *timeDelay* yang awal), begitu seterusnya hingga pada *timeDelay* 100.

Pada waktu melewati *timeDelay* 100, ketika dikurangi dengan 100, maka *timeDelay* bernilai 0. Kondisi ini cocok dengan baris ke 23,

```
23      if(timeDelay <= 0){
24        timeDelay = 1000;
```

```
25    }
```

Jika (IF) *timeDelay* lebih kecil atau sama dengan 0, maka *timeDelay* akan diisi dengan 1000. Sehingga nilai *timeDelay* tidak akan pernah negatif dan hal tersebut akan berulang terus-menerus. Hal ini lah yang membuat durasi nyala hidup-mati LED bisa berubah lebih cepat.

Beberapa operator matematika yang dibutuhkan dalam bahasa pemrograman yaitu:

Operator	Arti
=	Operator assignment, untuk memberi nilai pada variabel
+	Operator penambahan
-	Operator pengurangan
*	Operator perkalian
/	Operator pembagian. Sebagai catatan: - Jika tipe data yang digunakan adalah integer (*int*), maka hasil bagi adalah nilai asli, bukan desimal. Misal *5/2 = 2*, bukan *2.5* atau *3*. Tapi jika tipe data yang digunakan adalah double/float, maka hasil bagi adalah angka desimal. Misal, *5/2 = 2.5*
%	Operator modulo (sisa pembagian). Misal: - 10%2 = 0, 10 dibagi 2 = 5 + **0** - 10%3 = 1, 10 dibagi 3 = 3 + **1** - 10%4 = 2, 10 dibagi 4 = 8 + **2** - 10%5 = 0, 10 dibagi 5 = 2 + **0**

Perintah IF pasti akan diikuti dengan kondisi yang bernilai *True* yang diapit dengan tanda kurung, *if (kondisi)*. Pada Sketch 2.1 di atas kondisi yang digunakan adalah *timeDelay <= 0*, *timeDelay* lebih kecil atau sama dengan nol. Artinya, jika timeDelay bernilai 0 atau lebih kecil dari 0, maka blok kode dalam *if* akan dieksekusi.

Selain operator <=, maka berikut ini adalah beberapa operator yang sering digunakan:

Operator	Arti
==	Sama dengan
!=	Tidak sama dengan
<	Lebih kecil

>	Lebih besar
<=	Lebih kecil atau sama dengan
>=	Lebih besar atau sama dengan

Yap, begitulah cara kerja IF dan beberapa operator yang bisa digunakan untuk memeriksa kondisi dalam IF. Simpel sekali bukan? ☺

2.1.2 Perintah IF - ELSE

Pada dasarnya IF-ELSE merupakan pengembangan dari IF. ELSE berarti kondisi yang tidak sesuai dengan kondisi dalam IF. Dengan kata lain, ELSE artinya "jika tidak". Coba perhatikan kedua pernyataan berikut:

```
1  Main;
2  jika (sekarang == jam 10){
3     Makan;
4  }
5
6
7
8  Jika (sekarang == jam 10){
9     Makan;
10 }jika tidak{
11    Main;
12 }
```

Pada baris 1 sampai 5, maka **Main** akan terus dieksekusi. Jika sekarang sama dengan jam 10, maka yang dilakukan adalah **Main** sambil **Makan**. Jadi kedua kegiatan atau statemen akan dieksekusi.

Berbeda dengan baris 8 sampai 12, jika jam 10 **Makan**, jika tidak jam 10, **Main**. Begitulah logika IF-ELSE.

Berikut ini adalah Sketch 2.2 yang merupakan hasil modifikasi dari Sketch 2.1 dengan tambahan ELSE dan pemindahan proses pengurangan:

Sketch 2.2 Perubahan IF-ELSE

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
```

```
// coder elangsakti

// Pin 8 untuk LED
const int pinLED = 8;

void setup() {
   // pin LED sebagai output
   pinMode(pinLED, OUTPUT);
}

// awal time delay 1000 | 1 detik
int timeDelay = 1000;

void loop() {

   /* Jika timeDelay bernilai lebih kecil sama dengan 0
      maka LED akan diam selama 3 detik
      lalu nilai timeDelay direset ke 1000
    */
   if(timeDelay <= 100){
     delay(3000);
     timeDelay = 1000;
   }else{
     // nilai timeDelay dikurangi 100 jika time delay > 100
     timeDelay = timeDelay - 100;
   }

   //Nyalakan dan matikan LED selama timeDelay
   digitalWrite(pinLED, HIGH);
   delay(timeDelay);
   digitalWrite(pinLED, LOW);
   delay(timeDelay);
}
```

Pada Sketch 2.2, jika *timeDelay* bernilai lebih kecil sama dengan 100, maka akan LED akan diam selama 3 detik lalu nilai *timeDelay* akan direset ke 1000, jika tidak maka akan dilakukan pengurangan terhadap *timeDelay* sebanyak 100. Perhatikan, peroses reset dan pengurangan tidak pernah dilakukan bersama-sama. Silakan dicoba dan cek hasilnya! ☺

Setelah membahas tentang IF dan IF-ELSE, kita akan mebahas tentang *While*. *While* merupakan salah satu model perulangan dengan karakteristik tertentu. Untuk lebih jelasnya, silakan lanjut ke pembahasan selanjutnya.

2.2 Perulangan dengan WHILE

Perintah WHILE merupakan perintah untuk melakukan perulangan berdasarkan suatu kondisi, jadi banyaknya perulangan tidak bisa ditentukan dengan pasti. Dalam WHILE seakan ada pengecekan kondisi seperti perintah IF untuk melakukan perulangan. Bentuk umum dari perintah WHILE yaitu :

```
1  while( kondisi ){
2     // eksekusi code
3
4  }
```

Jika kondisi sesuai, maka perintah atau *source code* yang ada dalam kurung kurawal "{}" tersebut akan dieksekusi. Untuk lebih memahami tentang perintah WHILE, mari kita modifikasi Sketch 2.2 dengan penambahan WHILE dan beberapa perubahan lainnya.

Sketch 2.3 Perulangan While

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // Pin 8 untuk LED
6  const int pinLED = 8;
7
8  void setup() {
9     // pin LED sebagai output
10    pinMode(pinLED, OUTPUT);
11 }
12
13 // awal time delay 1000 | 1 detik
14 int timeDelay = 1000;
15
16 void loop() {
17
18   // selama nilai timeDelay > 0
19   // eksekusi blok program ini
20   while(timeDelay > 0){
21     // LED hidup mati dengan durasi 500 milisekon
22     digitalWrite(pinLED, HIGH);
23     delay(500);
24     digitalWrite(pinLED, LOW);
25     delay(500);
26
27     // kurangi timeDelay dengan 100
```

```
    timeDelay = timeDelay - 100;
  }

  // setelah timeDelay dikurangi terus-menerus
  // maka pada akhirnya akan bernilai minus atau < 0
  // maka while di atas akan berhenti

  // selama nilai timeDelay < 1000
  // eksekusi blok program ini
  while(timeDelay < 1000){
    // LED hidup mati dengan durasi 100 milisekon
    digitalWrite(pinLED, HIGH);
    delay(100);
    digitalWrite(pinLED, LOW);
    delay(100);

    // tambahkan timeDelay dengan 100
    timeDelay = timeDelay + 100;
  }
}
```

Program pada Sketch 2.3 akan mengedipkan LED dengan durasi yang berbeda. Awalnya LED akan berkedip dengan durasi ½ detik sebanyak 10 kali, selanjutnya LED akan berkedip lebih cepat dengan durasi 1/10 detik sebanyak 10 kali.

2.3 Kondisi True dan False

Dalam bahasa pemrograman, kita nantinya akan mempelajari tentang perintah-perintah yang berkaitan dengan kondisi (logika) True dan False. Perintah-perintah yang berkaitan dengan logika yang umum digunakan misalnya IF dan IF-ELSE, dan WHILE yang sudah kita pelajari sebelumnya.

Secara teori, True berarti 1 dan False berarti 0. Dalam pemrograman yang ini, kondisi False memang selalu 0, tapi True tidak selalu 1. Kondisi True adalah selain 0, ingat selain nol (0) akan dianggap True. Coba perhatikan kode di bawah ini:

```
while(1){
   digitalWrite(pinLED, HIGH);
   delay(100);
   digitalWrite(pinLED, LOW);
```

```
   delay(100);
}
```

Program di atas akan dijalankan selamanya (*looping forever*) selama Arduino belum direset atau listrik tidak diputus. Kenapa? Sebab 1 berarti True. Secara harfiah, baris *while(1)* dapat diartikan "*selama bernilai benar, maka eksekusi kode ini*".

Ingat, penulisan operator logika "sama dengan" tidak hanya menggunakan satu tanda sama dengan "=", tapi menggunakan dua sama dengan "==". Jika hanya menggunakan "=", maka pertanyaan tersebut bukan untuk logika, tapi untuk operator penetapan isi variabel (*assignment*). Penulisan yang benar adalah:

```
while( timeDelay == 1000 ){
  //
}

if( timeDelay == 0){
   //
}
```

Bukannya,

```
while( timeDelay = 1000 ){    // penulisan yang SALAH!
   //
}

if( timeDelay = 0 ){   // penulisan yang SALAH!
   //
}
```

2.4 Kombinasi True dan False

Terkadang kondisi True atau False bisa terdiri dari beberapa kondisi. Misal, kita ingin menentukan bilangan *timeDelay* sebagai angka positif tidak lebih dari 10. Maka kita tidak bisa hanya menggunakan satu kondisi. Kita harus menggunakan 2 kondisi, yaitu timeDelay harus lebih besar dari 0 DAN *timeDelay* lebih kecil dari 10. Dalam logika, kita bisa menuliskannya dengan cara :

```
if( (timeDelay > 0) && (timeDelay < 10) ){
   //
}
```

Perhatikan disana ada tanda && yang berarti AND. Penambahan tanda kurung untuk memperjelas bahwa ada dua kondisi yang berbeda dan dihubungkan oleh operator &&. Silakan diingat kembali bahwa operator logika AND akan menghasilkan nilai True jika semua kondisi bernilai True. Selain operator AND, ada juga operator OR dan NOT. Perhatikan tabel berikut:

Operator	Contoh	Arti
&&	(A < 10) && (B >10)	Logika AND akan menghasilkan True apabila kondisi A dan B sesuai (True), jika tidak, maka False
||	(A < 10) && (B >10)	Logika OR akan menghasilkan True apabila salah satu dari A, B, atau semuanya sesuai (True), tapi jika keduanya False, maka akan bernilai False
!	! (A < 10)	Logika NOT akan menghasilkan True jika kondisi tidak sesuai (False), sebab NOT adalah kondisi sebaliknya.

Jika Anda masih bingung dengan NOT, maka perhatikan kode berikut ini:

```
int status = HIGH;

int led = !status;    // artinya, led TIDAK high (LOW)
```

Jika *status* = HIGH atau True, maka !*status* = LOW (tidak HIGH) atau False (tidak True). Penggunaan juga sering digunakan, jadi tolong diingat-ingat. ☺

2.5 Perulangan dengan FOR

Berbeda dengan WHILE, dengan FOR kita bisa menentukan jumlah perulangan dengan pasti. Pada Sketch 2.3, masing-masing WHILE akan melakukan perulangan sebanyak 10 kali dengan cara mengurangi angka 1000 dengan 100 secara bertahap. Jika menggunakan FOR, kita bisa melakukan perulangan tersebut lebih sederhana. Format dasar dari FOR adalah :

```
for(statemen; kondisi; statemen){
```

Statemen yang pertama berisi tentang kondisi awal, biasanya inisialisasi suatu variabel atau data (misal, a=0). Sedangkan statemen yang terakhir adalah perubahan yang akan terjadi pada variabel pada statemen awal (misal a=a+1). Sedangkan kondisi merupakan kondisi dimana perulangan akan terjadi, ketika kondisi sudah tidak sesuai, maka perulangan akan berhenti. untuk lebih jelasnya tentang FOR, berikut contoh sederhananya:

Sketch 2.4 Perulangan FOR

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // Pin 8 untuk LED
6  const int pinLED = 8;
7
8  void setup() {
9     // pin LED sebagai output
10    pinMode(pinLED, OUTPUT);
11 }
12
13 // awal time delay 1000 | 1 detik
14 int timeDelay = 3000;
15
16 void loop() {
17   // perulangan sebanyak 10 kali dari 1 hingga 10
18   for(int i=1; i<=10; i++){
19     // LED hidup mati dengan durasi 500 milisekon
20     digitalWrite(pinLED, HIGH);
21     delay(500);
22     digitalWrite(pinLED, LOW);
23     delay(500);
24   }
25   // diam selama 3 detik
```

```
26    delay(timeDelay);
27  }
```

Ketika program tersebut diupload, maka LED akan berkedip dengan durasi ½ detik sebanyak 10 kali, kemudian LED akan diam (mati) selama 3 detik (perhatikan, nilai timeDelay diganti 3000), dan melanjutkan berkedip lagi. Perhatikan pada baris 14:

```
18    for(int i=1; i<=10; i++){
```

Pada baris 18 menunjukkan FOR dengan kondisi nilai awal i=1, i akan terus-menerus ditambah dengan 1 (i++) selama perulangan selama i <= 10. Sederhananya, nilai i akan berubah dari 1, 2, 3, 4, … hingga 10.

Bagian penting yang perlu dipahami adalah i++, i++ sama dengan

$$i = i + 1;$$

Selain i++, ada juga ++i, i--, dan --i. Penulisan i-- sama dengan i++, hanya saja i-- berarti $i=i-1$, atau i akan berkurang 1 terus-menerus selama perulangan.

Penempatan ++ di depan atau di belakang i berarri bahwa : jika ++ nya ada di belang, maka proses penambahannya dilakukan setelah blok kode dalam for dijalankan. Tapi jika ++ nya ada di depan, maka proses penambahan akan dilakukan sebelum proses dijalankan. Begini contohnya:

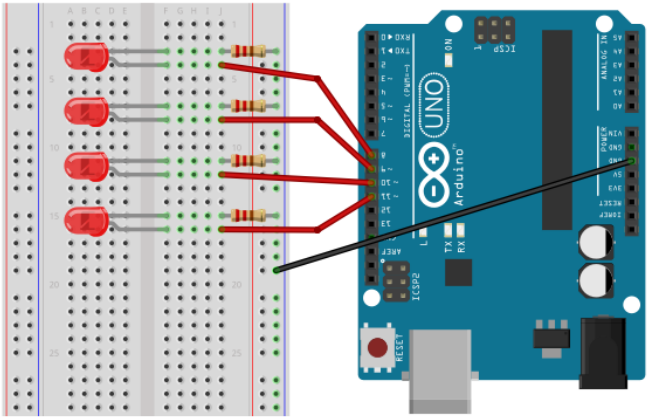
```
1   // Free Ebook Arduino
2   // www.elangsakti.com
3   // coder elangsakti
4
5   int i = 1;
6   int j = 3 * (i++);  // j = 3 karena 3 * 1
7
8   // i = 2
9
10  int j = 3 * (++i);  // j = 9 karena 3 * 3
11
12  // i sudah ditambah dengan 1 lagi (++i) sebelum baris 6 dieksekusi
```

Perhatikan, pada baris ke-6, j bernilai 3 karena 3 dikali dengan nilai i = 1. Setelah proses pada baris ke-4, maka nilai i akan berubah menjadi 2 karena ada statemen i++ (i = i+1).

Sedangkan pada baris ke-10, j bernilai 9 (bukan 6) karena i tidak bernilai 2 lagi, tapi i bernilai 3 sebab sebelum baris tersebut dieksekusi i terlebih dahulu ditambahkan dengan 1 (++1). Jadi penambahan i tidak dilakukan setelah eksekusi baris 10 baris tersebut, tapi sebelum mengeksekusi baris ke-10.

Semoga sekilas penjelasan ini bisa memberikan gambaran tentang perbedaan FOR dan WHILE. Jika ada penjelasan yang membingungkan, silakan ditanyakan di http://www.elangsakti.com. Setelah ini kita akan merangkai kombinasi LED lebih dari satu karena kita akan mencoba belajar tentang Array. ☺

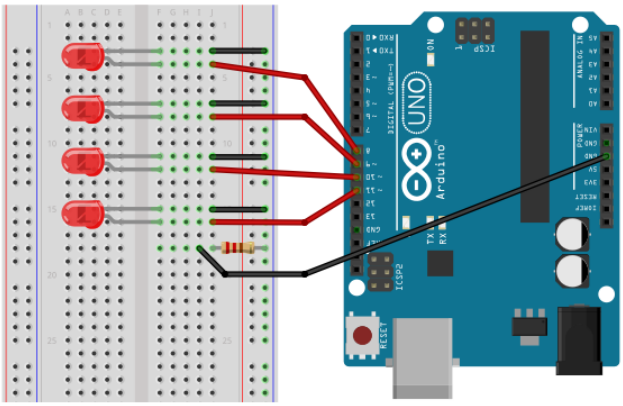
2.6 Update Rangkaian LED



Rangkaian 2.1 Array LED dengan 4 resistor

Silakan update rangkaian Arduino Anda seperti rangkain 2.1. Siapkan 4 buah resistor dan 4 buah LED. Siapkan kabel jumper untuk menyuplai GDN pada *project board*. Masing-masing kaki negatif LED dihubungkan ke GND dengan resistor. Sedangkan keempat LED tersebut dihubungkan berturut-turut dengan pin 8, 9, 10, dan 11 pada *board* Arduino.

Jika Anda tidak memiliki banyak resistor untuk dicoba, maka Anda bisa menggukaan 1 resistor saja dengan rangkaian seperti Rangkaian 2.2, tapi dengan konsekuensi : suplai arus akan dipakai bersama sehingga nyala LED akan semakin redup (atau mungkin LED tidak akan menyala, tergantung pada LED yang Anda gunakan).



Rangkaian 2.2 Array LED dengan 1 resistor

Pada rangkaian 2.2, keempat resistor digantukan dengan kabel jumper, kemudian pasang 1 resistor untuk menghubungkan kolom GND pada *project board* ke GND pada *board* Arduino.

Untuk sekedar percobaan, cara seperti Rangkaian 2.2 bisa Anda gunakan. Tapi Anda tidak disarankan mengaplikasikan cara yang kedua ketika menggunakan sensor sebab sedikit banyak akan mempengaruhi kinerja sensor tersebut.

Jika sudah dirangkai, silakan upload Sketch 2.5 berikut:

Sketch 2.5 Animasi LED

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // Inisialisasi Pin LED
6  const int pinLED1 = 8;
7  const int pinLED2 = 9;
8  const int pinLED3 = 10;
9  const int pinLED4 = 11;
10
```

```
11  void setup() {
12     // pin LED sebagai output
13     pinMode(pinLED1, OUTPUT);
14     pinMode(pinLED2, OUTPUT);
15     pinMode(pinLED3, OUTPUT);
16     pinMode(pinLED4, OUTPUT);
17  }
18
19  void loop() {
20    // perulangan sebanyak 5 kali
21    // dari i=0 hinga i=4 atau (i < 5)
22    for(int i=0; i<5; i++){
23      if(i==1){
24        // jika i=1, hidupkan led 1, led yang lain mati
25        digitalWrite(pinLED1, HIGH);
26        digitalWrite(pinLED2, LOW);
27        digitalWrite(pinLED3, LOW);
28        digitalWrite(pinLED4, LOW);
29      }else if(i==2){
30        // jika i=2, hidupkan led 1 & 2, led 3 & 4 mati
31        digitalWrite(pinLED1, HIGH);
32        digitalWrite(pinLED2, HIGH);
33        digitalWrite(pinLED3, LOW);
34        digitalWrite(pinLED4, LOW);
35      }else if(i==3){
36        // jika i=3, hidupkan led 1, 2, & 3, led 4 mati
37        digitalWrite(pinLED1, HIGH);
38        digitalWrite(pinLED2, HIGH);
39        digitalWrite(pinLED3, HIGH);
40        digitalWrite(pinLED4, LOW);
41      }else if(i==4){
42        // jika i=4, hidupkan semua led
43        digitalWrite(pinLED1, HIGH);
44        digitalWrite(pinLED2, HIGH);
45        digitalWrite(pinLED3, HIGH);
46        digitalWrite(pinLED4, HIGH);
47      }else{
48        // jika tidak, matikan semua led
49        digitalWrite(pinLED1, LOW);
50        digitalWrite(pinLED2, LOW);
51        digitalWrite(pinLED3, LOW);
52        digitalWrite(pinLED4, LOW);
53      }
54      // delai selama 5 detik
55      delay(5000);
56    }
57  }
```

Program di atas akan membuat LED menyala bergantian sebanyak 5 animasi (perulangan sebanyak 5 kali). Pertama, semua LED

akan mati selama 5 detik. Kedua, LED 1 akan menyala. Ketiga, LED 1 dan 2 akan menyala. Keempat, LED 1, 2, dan 3 akan menyala. Kelima, semua LED akan menyala.

Animasi tersebut ditentukan berdasarkan nilai i, nilai i diperiksa dengan perintah IF. Jika nilai i=0, maka semua LED mati, jika i=1 maka satu LED nyala, dan seterusnya.

Selain menggunakan IF, ada cara lain yang lebih simpel untuk membuat animasi LED seperti program pada Sketch 2.5. perhatikan Sketch 2.6:

Sketch 2.6 Animasi LED Alternatif

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // Inisialisasi Pin LED
6  const int pinLED1 = 8;
7  const int pinLED2 = 9;
8  const int pinLED3 = 10;
9  const int pinLED4 = 11;
10
11 void setup() {
12   // pin LED sebagai output
13   pinMode(pinLED1, OUTPUT);
14   pinMode(pinLED2, OUTPUT);
15   pinMode(pinLED3, OUTPUT);
16   pinMode(pinLED4, OUTPUT);
17 }
18
19 void loop() {
20   digitalWrite(pinLED1, LOW);
21   digitalWrite(pinLED2, LOW);
22   digitalWrite(pinLED3, LOW);
23   digitalWrite(pinLED4, LOW);
24   delay(1000);
25   digitalWrite(pinLED1, HIGH);
26   delay(1000);
27   digitalWrite(pinLED2, HIGH);
28   delay(1000);
29   digitalWrite(pinLED3, HIGH);
30   delay(1000);
31   digitalWrite(pinLED4, HIGH);
32   delay(1000);
33 }
```

Dengan memanfaatkan delay, program pada Sketch 2.6 lebih simple daripada Sketch 2.5. Kira-kira, apakah ada cara yang lebih simpel lagi? Ada! Kita bisa menggunakan Array. Apa itu Array?

2.7 Pengenalan Array

Array merupakan variabel yang bisa menampung banyak data, masing-masing data bisa diambil dengan alamat indeks (posisi) data dalam Array tersebut. Alamat indeks pada array standarnya adalah angka integer yang diawali dari angka 0. Jadi, jika kita punya data 5 dalam variabel Array, maka data pertama pada alamat indeks ke-0, data ke-2 pada alamat indeks ke-1, dan data ke-5 pada alamat indeks ke-4.

Untuk lebih memahami tentang Array, program pada Sketch 2.7 merupakan versi lebih simpel daripada Sketch 2.6 dengan memanfaatkan perulangan FOR dan Array.

Sketch 2.7 Animasi LED dengan Array

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // Inisialisasi Jumlah LED
6  const int numLED = 4;
7  // LED 1,2,3,&4 jadi 1 varibel
8  // dengaan alamat index 0,1,2,3
9  const int pinLED[numLED] = {8,9,10,11};
10
11 void setup() {
12   // Inisialisasi semua pin LED sebagai OUTPUT
13   for(int i=0; i<4; i++){
14     pinMode(pinLED[i], OUTPUT);
15   }
16 }
17
18 void loop() {
19   // Matikan semua LED
20   for(int i=0; i<4; i++){
21     digitalWrite(pinLED[i], LOW);
22   }
23   delay(1000);
24
25   // Hidupkan semua LED bertahap dg jeda 1 detik
26   for(int i=0; i<4; i++){
```

```
27      digitalWrite(pinLED[i], HIGH);
28      delay(1000);
29    }
30  }
```

Jika dibandingkan dengan Sketch 2.6 yang tanpa baris keterangan komentar, maka Sketch 2.7 jauh lebih pendek dan lebih simpel.

Pada Sketch 2.7, keempat LED dimuat dalam satu variabel yaitu variabel Array pinLED. Masing-masing LED bisa diakses berdasarkan alamat indeks. Perhatikan pada baris 9.

```
9   const int pinLED[numLED] = {8,9,10,11};
```

Secara otomatis, alamat indeks untuk masing-masing LED tergantung pada urutan pin tersebut. Pin 8 ada pada alamat indeks ke-0, pin 9 ada pada indeks ke 1, pin 10 ada di indeks ke 2, dan pin 11 ada pada indeks ke 3. Sehingga untuk mengakses pin 8, kita bisa menambahkan kurung siku dan alamat indeks ke-0, seperti berikut:

$$Pin8 = pinLED[0]$$

Karena indeks tertinggi adalah angka 3, maka di setiap perulangan LED tidak boleh lebih dari 3 atau < 4, perhatikan perintah FOR berikut:

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

Yap, semoga penjelasan Array di atas mudah dipahami. Jika ada yang membingungkan, silakan tanya-tanya di website, dengan senang hati penulis akan menjawab. Baik, selanjutnya kita akan lebih mengembangkan Sketch 2.7 agar sedikit lebih kompleks dan animasinya lebih menarik. Penulis memberikan Sketch 2.8 dan Sketch 2.9 untuk dianalisa dan dipelajari:

Sketch 2.8 Bonus Animasi LED 1

```
1   // Free Ebook Arduino
2   // www.elangsakti.com
3   // coder elangsakti
4
```

```
// Inisialisasi Jumlah LED
const int numLED = 4;
// LED 1,2,3,&4 jadi 1 varibel
// dengaan alamat index 0,1,2,3
const int pinLED[numLED] = {8,9,10,11};

void setup() {
  // Inisialisasi semua pin LED sebagai OUTPUT
  for(int i=0; i<4; i++){
    pinMode(pinLED[i], OUTPUT);
  }
}

void loop() {
  // Hidupkan semua LED bertahap dg jeda 1 detik
  for(int i=0; i<4; i++){
    digitalWrite(pinLED[i], HIGH);
    delay(500);
  }

  // Matika semua LED bertahap dg jeda 1 detik
  for(int i=0; i<4; i++){
    digitalWrite(pinLED[i], LOW);
    delay(500);
  }
}
```

Sketch 2.9 Bonus Animasi LED 2

```
// Free Ebook Arduino
// www.elangsakti.com
// coder elangsakti

// Inisialisasi Jumlah LED
const int numLED = 4;
// LED 1,2,3,&4 jadi 1 varibel
// dengaan alamat index 0,1,2,3
const int pinLED[numLED] = {8,9,10,11};

void setup() {
  // Inisialisasi semua pin LED sebagai OUTPUT
  for(int i=0; i<4; i++){
    pinMode(pinLED[i], OUTPUT);
  }
}

void loop() {
  // hidupkan led indeks 0 hingga 2 satu-persatu
```

```
  for(int i=0; i<3; i++){
    digitalWrite(pinLED[i], HIGH);
    delay(200);
    digitalWrite(pinLED[i], LOW);
  }
  // hidupkan led indeks 3 hingga 1 satu-persatu
  for(int i=3; i>0; i--){
    digitalWrite(pinLED[i], HIGH);
    delay(200);
    digitalWrite(pinLED[i], LOW);
  }
}
```

Apakah Kelemahan kita?

Kelemahan kita ialah, kita kurang percaya diri kita sebagai bangsa, sehingga kita menjadi bangsa penjiplak luar negeri, kurang mempercayai satu sama lain, padahal kita ini asalnya adalah Rakyat Gotong Royong!

(Ir. Soekarno)

Bagian 3

Input

Sebelumnya kita telah belajar tentang bagaimana mengendalikan LED. Untuk mengendalikan LED kita menjadi pin pada Arduino sebagai OUTPUT. Pada bagian ini kita akan membahas tentang bagaimana menjadikan pin Arduino sebagai INPUT dan sebagai aplikasinya, kita akan menggunakan komponen *pushbotton* dan *potensiometer* sebagai input untuk mengendalikan LED. Bagian ini akan menjadi dasar agar Anda memahami bagaimana membuat Arduino bisa membaca sensor untuk mendeteksi kondisi lingkungan sekitar.

3.1 Pushbutton

Pertama kita akan bermain dengan tombol *pushbutton* (*tactile*) atau tombol push on. Ketika tombol ini ditekan, maka jalur akan tertutup (ON), ketika dilepas jalur akan kembali terbuka (OFF). Tombol banyak digunakan untuk peralatan seperti remote, keypad, keyboard, atau tombol untuk pengaturan TV, ld atau sejenisnya.

Gambar 3.1 Pushbutton dan simbolnya

Gambar 3.1 merupakan bentuk fisik *pushbutton* dan salah satu simbol *pushbotton* jenis NO (*Normally Open*) dalam rangkaian elektronik. Berdasarkan simbol tersebut, *Normally Open* berarti kondisi normal (sebelum ditekan), maka terminal dalam kondisi tidak tersambung (*open*, terbuka). Tapi ketika ditekan, maka masing-masing terminal akan terhubung.

Selain jenis NO, ada juga *pushbutton* jenis NC (*Normally Close*), artinya ketika kondisi normal (sebelum ditekan), kaki terminal dalam keadaan terturup / tersambung (*Close*), tapi ketika ditekan, kaki terminalnya terbuka (tidak tersambung). Dalam ebook ini, kita akan menggunakan jenis *pushbutton* NO.

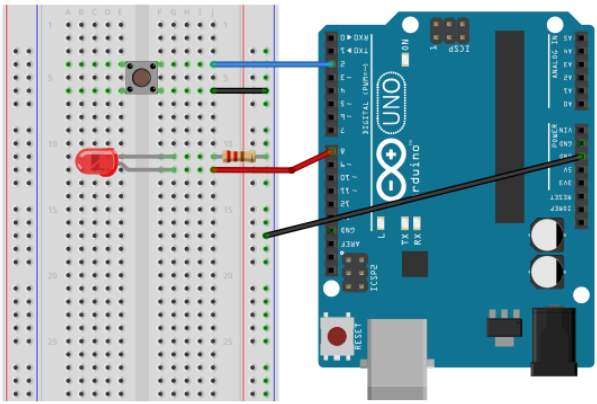
3.1.1 Satu Tombol dan Satu LED

Percobaan kali ini adalah untuk mengendalikan hidup/matinya-nya LED dengan tombol *pushbutton*. Jika tombol ditekan, LED akan menyala, jika dilepas, LED kembali padam.

Untuk melakukan percobaan ini, siapkan sebuah *pushbutton*, sebuah LED, dan sebuah resistor. Siapkan juga beberapa kabel jumper untuk merangkai komponen-komponen tersebut.

3.1.1.1 Rangkaian

Silakan buat rangkaian seperti Rangkaian 3.1 berikut:



Rangkaian 3.1 Pushbutton dan LED

1. Siapkan LED dan *pushbutton* pada project board. Karena karena pushbutton memiliki 4 buah kaki yang masing-masing terpisah, maka silakan tancapkan *pushbutton* di tengah-tengah lajur project board sehingga kaki-kainya tidak tersambung.

2. Salah satu kaki pushbutton dihubungkan ke GDN di project board, sedangkan kaki pasangannya disambungkan ke pin 2 pada board Arduino. Bagaimana cara mengetahui pasangan kaki-kaki pada pushbutton? Anda bisa mengeceknya dengan AVO meter.
    - ✓ Setting AVO meter untuk menghitung resistansi, kemudian cek masing-masing pin *pushbutton* dengan probe. Jika tombol ditekan jarum AVO meter bergerak menyimpang, berarti kaki-kaki tersebut sepasang.
3. Untuk LED, sambungkan kaki negatif (pin yang lebih pendek) ke GND dengan resistor
4. Kaki positif (kaki yang lebih panjang) disambungkan ke pin 8 pada board Arduino dengan jumper.

3.1.1.2 Program

Mari kita coba dengan program berikut:

Sketch 3.1 Mengendalikan LED dengan pushbutton

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // pin 2 sebagai input dan pin 8 sebagai output
6  const int pinButton = 2;
7  const int pinLED = 8;
8
9  void setup() {
10   pinMode(pinButton, INPUT);
11   pinMode(pinLED, OUTPUT);
12
13   // aktifkan pull-up resistor
14   digitalWrite(pinButton, HIGH);
15 }
16
17 void loop() {
18   if(digitalRead(pinButton) == LOW){
19     digitalWrite(pinLED, HIGH);
20   }else{
21     digitalWrite(pinLED, LOW);
22   }
23 }
```

Pertama kali dijalankan, maka awalnya LED akan padam. ketika kita menekan tombol *pushbutton*, maka LED akan menyala. LED akan kembali padam ketika tekanan tombol dilepas. Pada Sketch 3.1 di atas ada beberapa baris kode baru:

```
10   pinMode(pinButton, INPUT);
11   pinMode(pinLED, OUTPUT);
12
13   // aktifkan pull-up resistor
14   digitalWrite(pinButton, HIGH);
```

Baris 6 berfungsi untuk mengeset *pinButton* sebagai INPUT. Jika sebuah pin diset sebagai INPUT, maka mikrokontroller akan mengambil data dari pin tersebut. Jika sebuah pin diset sebagai OUTPUT, maka mikrokontroller akan menuliskan data pada pin tersebut. dalam hal ini, mikrokontroller akan mengambil data yang dari *pushbutton.*

Perhatikan baris ke-14. Nilai *pinButton* awalnya diset HIGH. Kenapa diset HIGH? Kenapa bisa diset nilai pinnya menjadi HIGH, padahal pin tersebut tidak terhubung dengan +5V?

Begini, pemilihan settingan awal dengan HIGH atau LOW untuk *pinButton* tergantung pada rangkaian yang akan digunakan. Rangkaian 3.1 menghubungkan *pinButton* (pin 8) ke GND, artinya, ketika *pushbutton* ditekan maka *pinButton* (pin 8) akan menjadi 0 (LOW). Padahal defaultnya, setiap pin bernilai LOW. Jika pin awalnya bernilai LOW, kemudian ditekan tetap bernilai LOW, lalu apa gunanya *pushbutton*? ☺

Padahal, fungsi utama dari saklar (dalam hal ini adalah *pushbutton*) adalah mengubah nilai yang awalnya LOW menjadi HIGH, atau sebaliknya. Nah, karena ketika *pushbutton* ditekan akan bernilai LOW (ke GND), maka awalnya harus kita set menjadi HIGH. Sehingga logika untuk *pushbutton* tersebut adalah: ketika tidak ditekan HIGH, ketika ditekan LOW.

Kondisi tersebut yang akan digunaka untuk mendeteksi apakah *pushbutton* tersebut ditekan atau tidak. Silakan perhatikan baris ke-18.

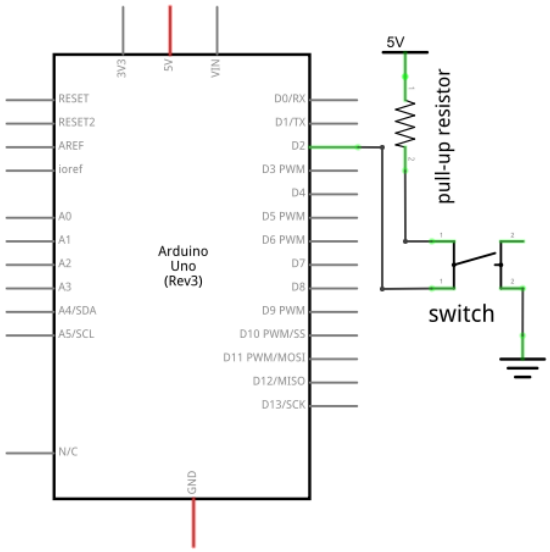
```
18   if(digitalRead(pinButton) == LOW){
```

```
19      digitalWrite(pinLED, HIGH);
20    }else{
21      digitalWrite(pinLED, LOW);
22    }
```

Pada baris 18, fungsi *digitalRead()* untuk membaca logika pada pinButton. Jika *pinButton* ditekan (LOW), maka hidupkan LED dengan perintah *digitalWrite(pinLED,HIGH)*. Ketika *pinButton* bernilai HIGH, matikan LED. Sederhana bukan? ☺

Untuk pertanyaan yang kedua, kenapa pin INPUT bisa diset HIGH?

Ketika kita menjadi pin INPUT sebagai HIGH, maka secara internal Arduino akan menghubungkan pin tersebut pada resistor *pull-up* bernilai 20k ohm. Apa itu *pull-up resistor*? Jika ada *pull-up*, apakah juga ada *pull-down resistor*?



Gambar 3.2 Skema pull-up resistor

Begini, dalam elektronika digital, jika sebuah pin diset sebagai INPUT, kemudian pin tersebut belum tersambung ke VCC atau GND, maka logika pada pin tersebut masih mengambang (*floating*). Oleh

sebab itu, pin tersebut harus ditentukan apakah akan diberi resistor *pull-up* (sehingga bernilai HIGH) atau diberi *pull-down* (sehingga bernilai LOW).

Jika pin tersebut diset HIGH, dalam internal mikrokontroller pin tersebut akan disambungkan ke VCC dengan pengaman sebuah resistor yang diistilahkan sebagai resistor *pull-up*. Begitu juga jika pin tersebut diset LOW, maka pin tersebut akan dihubungkan ke GND dengan pengaman resistor kemudian diistilahkan dengan resistor pull-down. Gambar 3.2 adalah rangkaian dasar *pull-up resistor* untuk rangkain di atas tanpa LED.

Semoga penjelasan tentang *pull-up resistor* bisa dipahami. Jika ada yang kurang jelas, mari kita diskusikan di website. ☺

3.1.2 Mengontrol Tingkat Kecerahan LED

Sebelumnya kita sudah membahas tentang cara menghidupkan dan mematikan LED dengan sebuah *pushbutton*. Selanjutnya, kita akan menggunakan dua buah *pushbutton* dengan ketentuan : *pushbutton* yang pertama untuk menaikkan kecerahan LED hingga paling terang, sedangkan *pushbutton* yang kedua untuk menurunkan kecerahan LED hingga LED padam.

Fungsi kedua *pushbutton* ini mirip dengan *volume-up* dan *volume-down*. Yang satu untuk meningkatkan volume (kecerahan), sedangkan satunya lagi untuk menurunkan volume (kecerahan). Yak, setidaknya Anda paham apa yang saya maksudkan. ☺

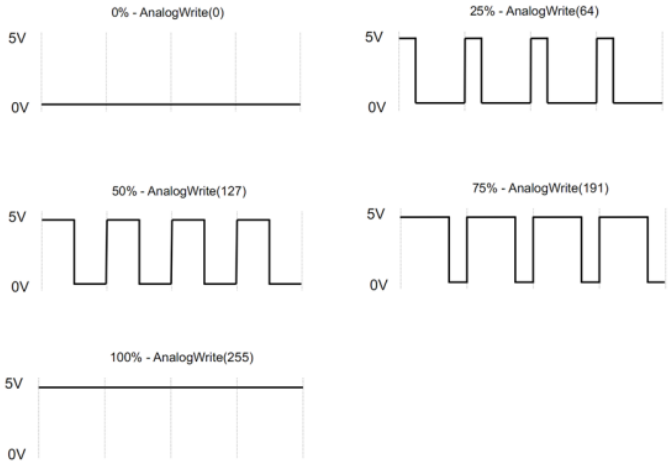
Setidaknya ada dua cara untuk menaikkan atau menurunkan tingkat kecerahan LED:

1. Mengubah arus yang masuk ke LED, cara ini bisa diaplikasikan dengan mengubah nilai resistor
2. Menghidup-matikan LED dengan cepat atau lambat. Begini, ketika kita menghidup-matikan LED dengan cepat, maka mata manusia tidak bisa mengetahuinya. Yang ditangkap oleh mata adalah terang atau redupnya saja. Jika kita menghidup-matikan led dengan cepat, maka LED tersebut akan terlihat terang, tapi kalau kita menghidup matikan LED dengan lebih lambat, maka LED akan terlihat lebih redup.

Dalam elektronika digital, konsep yang kedua dikenal dengan istilah PWM (*Pulse Width Modulation*). Apa itu PWM?

Sebagian kaki / pin Arduino support PWM, kaki yang support PWM ditandai dengan adanya tanda tilde (~) di depan angka pinnya, seperti 3, 5, 6, dan seterusnya. Frekuensi yang digunakan dalam Arduino untuk PWM adalah 500Hz (500 siklus dalam 1 detik). Jadi, Arduino bisa menghidup-matikan LED sebanyak 500 kali dalam 1 detik.

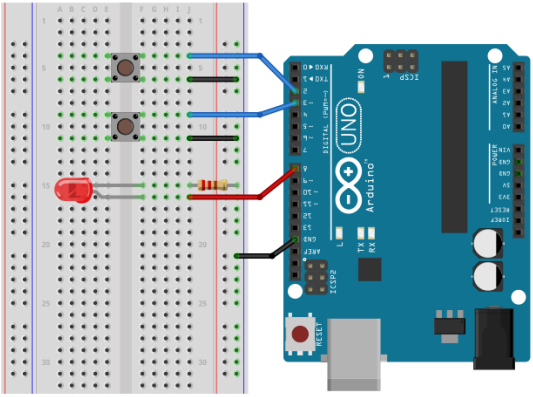
Untuk menggunakan PWM, kita bisa menggunakan fungsi *analogWrite()*. Nilai yang bisa dimasukkan pada fungsi tersebut yaitu antara 0 hingga 255. Nilai 0 berarti pulsa yang diberikan untuk setiap siklus selalu 0 volt, sedangkan nilai 255 berarti pulsa yang diberikan selalu bernilai 5 volt.



Gambar 3.3 Siklus pulsa PWM

Jika kita memberikan nilai 127 (kita anggap setengah dari 0 hingga 255, atau 50% dari 255), maka setengah siklus akan bernilai 5 volt, dan setengah siklus lagi akan bernilai 0 volt. Sedangkan jika jika memberikan 25% dari 255 (1/4 * 255 atau 64), maka 1/4 siklus akan bernilai 5 volt, dan 3/4 sisanya akan bernilai 0 volt, dan ini akan terjadi 500 kali dalam 1 detik. Untuk visualisasi siklus PWM, bisa Anda lihat Gambar 3.3.

3.1.2.1 Rangkaian



Rangkaian 3.2 Pengaturan Intensitas Cahaya LED

Buatlah rangkaian seperti gambar Rangkaian 3.2. Rencananya, *pushbutton* yang atas untuk menyalakan dan meningkatkan kecerahan LED, sedangkan *pushbutton* yang bawah untuk menurunkan tingkat kecerahan LED dan memadamkannya:

1. Seperti biasa, siapkan sebuah LED dan resistornya. Sambungkan kaki positif LED ke pin 8 Arduino.
2. Kemudian kaki negatif LED disambungkan ke resistor menuju GND.
3. Siapkan dua buah *pushbutton*. Pushbutton yang pertama (atas) disambunkan ke GND dan ke pin 2 pada board Arduino.
4. Lalu *pushbutton* yang kedua (bawah) disambungkan ke GND dan pin 3 pada board Arduino.

3.1.2.2 Program

Sketch 3.2 Mengatur intensitas cahaya LED

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // pin 2 & 3 sebagai input digital
6  const int pinBt1 = 2;
7  const int pinBt2 = 3;
8
```

```
 9  // Ingat, pin 9 support PWM
10  const int pinLED = 9;
11
12  void setup() {
13    pinMode(pinBt1, INPUT);
14    pinMode(pinBt2, INPUT);
15    pinMode(pinLED, OUTPUT);
16
17    digitalWrite(pinBt1, HIGH);
18    digitalWrite(pinBt2, HIGH);
19  }
20
21  int brightness = 0;
22  void loop() {
23    if(digitalRead(pinBt1) == LOW){
24      // jika pushbutton ditekan
25      // tambahkan nilai brightness
26      brightness++;
27    }else if(digitalRead(pinBt2) == LOW){
28      // jika pushbutton2 ditekan
29      // kurangi nilai brightness
30      brightness--;
31    }
32
33    // brightness dibatasi antara 0 - 255
34    // jika di bawah 0, maka ganti dengan 0
35    // jika di atas 255, maka ganti dengan 255
36    brightness = constrain(brightness, 0, 255);
37
38    // pinLED diberi nilai antara 0 - 255
39    analogWrite(pinLED, brightness);
40    // delay agar perubahannya bertahap
41    delay(20);
42  }
```

Ada 3 bagian pada Sketch 3.2 yang perlu dijelaskan lebih detail, yaitu pada baris 36, 39, dan 41.

```
33    // brightness dibatasi antara 0 - 255
34    // jika di bawah 0, maka ganti dengan 0
35    // jika di atas 255, maka ganti dengan 255
36    brightness = constrain(brightness, 0, 255);
```

Pada baris 36, kita menemukan satu fungsi baru, yaitu *constrain()*. Fungsi *constrain()* digunakan untuk menjaga agar nilai tetap pada range yang ditentukan. Pada kasus ini, *range* yang ditentukan adalah antara 0 – 255. Misal nilai *brightness* lebih kecil dari 0, maka akan dirubah

menjadi 0, tapi jika nilai brightness lebih besar dari 255, maka akan dirubah menjadi 255.

Untuk lebih memahami tentang fungsi *constrain()*, silakan perhatikan isi dari fungsi *constrain()* di bawah ini:

```
int constrain(int value, int min, int max){
   if(value > max){
      value = max;
   }
   if(value < min){
      value = min;
   }
   return value;
}
```

Tipe dari fungsi tersebut adalah *int* (integer), artinya fungsi tersebut akan mengembalikan nilai integer ketika dieksekusi disesuaikan dengan nilai value (***return*** *value*).

Jika diperhatikan fungsi *loop()* dan fungsi *setup()* bukanlah **int**, tapi **void**. Tipe fungsi **void** berbeda dengan tipe **int**, fungsi **void** tidak mengembalikan nilai apa pun, sehingga jika diperhatikan, tidak ada perintah *return* pada fungsi dengan tipe **void**. Yap, demikian sekilas tentang tipe fungsi integer dan void.

Selanjutnya, karena nilai brightness selalu antara 0-255, maka ketika dituliskan ke pinLED juga akan selalu selang antara 0-255 (perhatikan baris 39).

```
38   // pinLED diberi nilai antara 0 - 255
39   analogWrite(pinLED, brightness);
```

Fungsi *analogWrite()* digunakan untuk memberikan data PWM atau data analog. *analogWrite()* bisa menuliskan data dengan selang antara 0v hingga +5v pada pin INPUT. Berbeda dengan *digitalWrite()* yang hanya bisa menuliskan HIGH atau LOW, atau +5v atau 0v saja.

```
40   // delay agar perubahannya bertahap
41   delay(20);
```

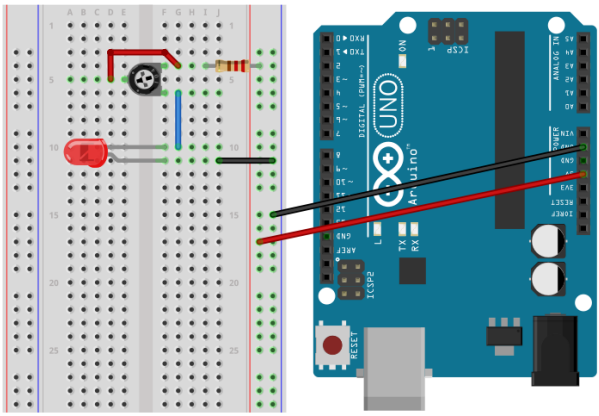
delay(20) berfungsi untuk mengatur durasi perubahan intensitas cahaya LED. Jika *delay(20)* kita hilangkan, maka LED akan langsung hidup atau langsung mati ketika tombol ditekan. Jika kecilkan nilainya, maka perubahan intensitas akan lebih cepat, dan sebaliknya, jika kita besarkan nilainya, maka perubahan intesitas akan lebih lama dengan catatan kita harus menahan ketika menekan tombol *pushbutton.*

Lalu bagaimana jika kita menekan kedua tombol *pushbutton* dengan bersamaan? Apa yang akan terjadi dengan LED? Silakan cari jawaban dan penjelasannya. ☺

3.2 Potensiomenter

Setelah kita belajar mengatur intensitas cahaya LED dengan *pushbutton*, kali ini kita akan mengunakan potensiometer. Kelebihan menggunakan potensiometer yaitu kita lebih mudah sebab kita hanya butuh satu alat untuk membuat LED lebih redup atau lebih terang.

Kenapa kita harus melalui mikrokontroller? Kan bisa saja kita menghubungkan langsung ke LED untuk mengatur arus yang masuk? Betul sekali.



Rangkaian 3.3 Mengatur intensitas cahaya LED dengan potensiometer

Jika kita langsung mengatur LED dengan potensiometer, kita harus memiliki potensiometer yang pas untuk LED tersebut. Jika hambatan potensiometer tidak sesuai, mungkin LED akan mati

sebelum potensiometer habis, atau LED sudah full nyalanya ketika potensiometer baru kita naikkan setengah. Jadi, kita tidak bisa menggunakan satu putaran full potensiometer untuk menaikkan atau menurunkan intensitas cahaya LED tersebut. Butuh bukti?

Pada rangkaian 3.3 menggunakan potensiometer 50k ohm jenis *trimmer*. Anda juga bisa mencobanya dengan menggunakan potensiometer putar. Yang digunakan di gambar adalah potensiometer trimmer yang ukurannya lebih kecil dan bisa ditancapkan ke project board. Gambar 3.4 adalah contoh salah satu *trimmer*.

Gambar 3.4 Potensiometer jenis trimmer

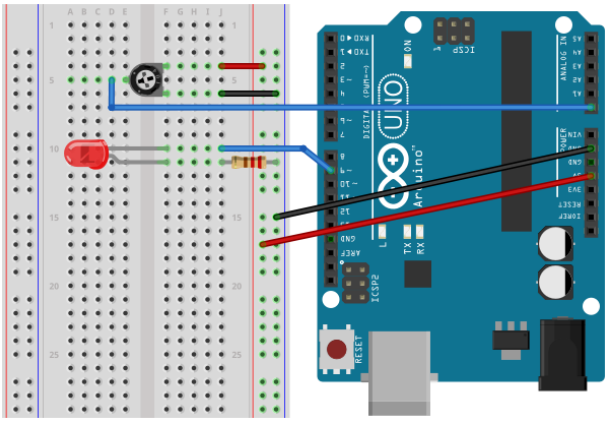
Berikut tahapan untuk membuat rangkaian tersebut:

1. Sambungkan kaki negatif LED ke GND dengan jumper
2. Kaki positif LED dihubungkan ke salah satu ujung kaki (kaki yang bawah) potensiometer dengan jumper
3. Kaki tengah (kaki di sisi yang sendirian) dihubungkan dengan ujung kaki yang lainnya (kaki yang atas) dengan jumper
4. Kaki yan atas dihubungkan ke +5v dengan resistor.

Jika Anda memutar kekiri atau ke kanan, maka LED akan menyala lebih terang, tapi LED tidak akan pernah padam. Hal tersebut disebabkan karena arus yang masuk ke LED tidak memungkinkan untuk membuat LED padam. Kenapa? Karena resistor yang dipilih tidak sesuai.

Solusinya, kita bisa menggukan teknik PWM seperti yang kita bahas sebelumnya. Hanya saja, untuk mengatur PWM kita menggunakan potensiometer, bukan *pushbutton*. Kebayang? ☺

3.2.1 Rangkaian



Rangkaian 3.4 Mengatur Brightness LED dengan potensiometer

Untuk membuat Rangkaian 3.4, siapkan LED, resistor, dan potensiometer. Anda juga bisa menggunakan *trimmer*.

1. Sambungkan kaki positif LED ke pin 9 pada *board* Arduino, pin tersebut support PWM
2. Kaki negatif LED disambungkan dengan resistor ke GND
3. Kedua ujung kaki trimmer yang satu sisi (sisi kanan) masing-masing disambungkan ke +5v dan GND. Jika Anda menggunakan potensiometer putar, yang disambungkan ke +5v dan GND adalah pin yang paling pinggi.
4. Pin yang satu (di sebelah kiri) disambungkan ke A0 pada *board* Arduino. Jika Anda menggunakan potensiometer putar,yang disambungkan ke A0 adalah pin yang tengah pada potensiometer.

Mungkin Anda akan bertanya, kenapa disambungkan ke A0? Begini, pada Arduino terdapat 3 kelompok pin dengan fungsi yang berbeda, yaitu:

- Pin digital (pin 0 - 13)
- Pin digital yang support PWM (ditandai dengan tilde "~", yaitu pin 3, 5, 6, 9, 10, 11)
- Pin Analog (A0 – A5)

Digital artinya hanya terdiri dari ON dan OFF, atau HIGH dan LOW. Digital dengan PWM artinya, frekuensi ON dan OFF bisa diatur berdasarkan siklus tertentu dalam frekuensi 500 Hz dengan selang antara 0 - 255. Hal ini sudah kita bahas sebelumnya, bukan?

Sedangkan pin Analog, berarti pin tersebut bisa ditulis mempunyai tegangan antara 0 – 5 volt dengan step kenaikan sebanyak 1024. Artinya angka 0 – 1023 akan dikonversi menjadi 0 – 5 volt pada pin tersebut. 0 berarti 0 volt, 1023 berarti 5 volt. Kenapa maksimal 1024?

Sebenarnya selang antara 0 – 5 volt bisa dicacah dengan jumlah tak terhingga. Akan tetapi, mikroposesor memiliki keterbatasan dalam mencacah angka, sehingga batas yang bisa dicacah hanya mencapai 1024 cacahan, yaitu dari 0 – 1023.

Itu sekilas tentang rangkaian yang akan kita buat dan karakteristik pin Analog pada Arduino.

3.2.2 Program

Sketch 3.3 Mengatur kecerahan LED dengan potensiometer

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // pin A0 adalah pin Analog
6  // pin 9 adalah pin digital support PWM
7  const int pinPot = A0;
8  const int pinLED = 9;
9
10 void setup() {
11   pinMode(pinPot, INPUT);
12   pinMode(pinLED, OUTPUT);
13 }
14
15 int sensor = 0;
16 int brightness = 0;
17
18 void loop() {
19   // baca nilai kaki A0 (sensor, potensiometer)
20   sensor = analogRead(pinPot);
21   // konversi nilai 0-1023 (Analog) menjadi 0-255 (PWM)
22   brightness = map(sensor, 0, 1023, 0, 255);
```

```
23
24    // tentukan brightness LED dengan PWM
25    analogWrite(pinLED, brightness);
26  }
```

Ketika program pada Sketch 3.3 diupload, maka kita bisa mengatur brightness dan hidup-mati LED dengan sempurna, berbeda dengan cara manual seperti Rangkaian 3.3. Pada Sketch 3.3, ada dua bagian yang perlu diperhatikan:

1. Pada baris 7, pinPot = A0. A0 adalah variabel untuk pin Analog ke 0. Sebenarnya A0 sama dengan pin 14. Kita juga bisa menggunakan pin tersebut sebagai pin digital. Tapi Anda tidak bisa menggunakan pin digital sebagai pin analog.

   Jadi, gunakan pin A0 – A5 jika akan dihubungkan dengan sensor analog.

2. Pada baris 22, ada fungsi *map()*. Sebagaimana namanya, fungsi *map()* digunakan untuk memetakan suatu nilai dari range tertentu ke range yang lain. Berikut adalah parameter dalam fungsi map()

   map(value, from_min, from_max, to_min, to_max);

   Maka pada baris 22 berfungsi untuk mengubah nilai *sensor* yang awalnya ada pada range 0-1024 menjadi nilai dengan range 0-255. Jika sensor benilai 512 (anggap saja ½ dari 1024), maka nilai tersebut akan dirubah menjadi 127 (anggap saja ½ dari 255).

Selanjutnya, mari kita coba untuk mengatur durasi kedipan LED berdasarkan nilai pada potensiometer. Jika '*volume*' potensiometer rendah, durasi kedipan LED akan cepat. Jika '*volume*' potensiometer tinggi, maka durasi kedipan LED akan lambat. Saya sebut '*volume*' karena potensiometer identik dengan alat untuk mengatur volume. ☺

Sketch 3.4 Kedipan LED dengan potensiometer

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
```

```
// coder elangsakti

// pin A0 adalah pin Analog
// pin 9 adalah pin digital support PWM
const int pinPot = A0;
const int pinLED = 9;

void setup() {
  pinMode(pinPot, INPUT);
  pinMode(pinLED, OUTPUT);
}

int sensor = 0;

void loop() {
  // baca nilai kaki A0 (sensor, potensiometer)
  sensor = analogRead(pinPot);

  // durasi kedipan sesuai nilai pada sensor 0-1023
  digitalWrite(pinLED, HIGH);
  delay(sensor);
  digitalWrite(pinLED, LOW);
  delay(sensor);
}
```

Sudah mencoba Sketch 3.4?

Kedipan LED pada Sketch 3.4 kurang responsive. Artinya, ketika potensiometer dimaksimalkan, maka delaynya akan lama. Tapi ketika potensiometer langsung diturunkan, paka kedipan LED tetap pada durasi yang lama tersebut. Kenapa? Karena mikrokontroller akan menyelesaikan nilai delay yang diberikan sebelumnya.

Setelah eksekusi semua delay dilakukan, baru potensiometer akan berubah ke durasi delay yang baru. Itulah yang saya sebut dengan kurang responsive. Harusnya, durasi kedipan LED akan segera berubah ketika potensiometer dirubah.

3.2.3 Menghilangkan Delay

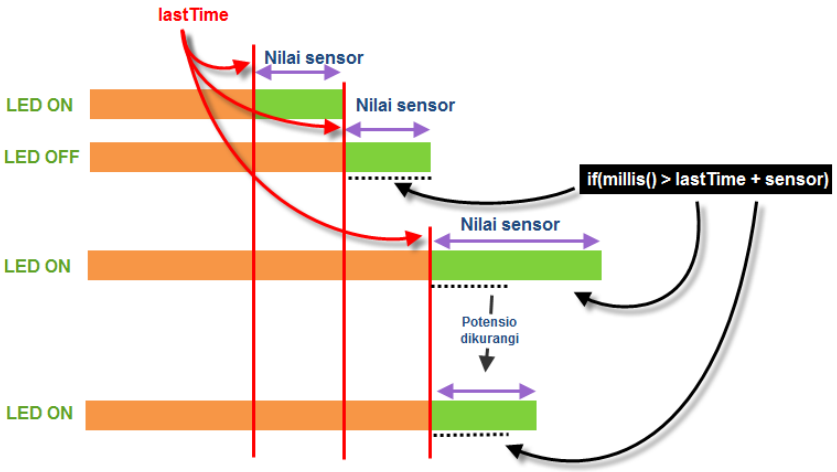
Sketch 3.5 Kedipan LED Responsive tanpa delay

```
// Free Ebook Arduino
// www.elangsakti.com
// coder elangsakti
```

```
4
5   // pin A0 adalah pin Analog
6   // pin 9 adalah pin digital support PWM
7   const int pinPot = A0;
8   const int pinLED = 9;
9
10  void setup() {
11    pinMode(pinPot, INPUT);
12    pinMode(pinLED, OUTPUT);
13  }
14
15  long lastTime = 0;
16  int sensor = 0;
17  int ledValue;
18
19  void loop() {
20    // baca nilai kaki A0 (sensor, potensiometer)
21    sensor = analogRead(pinPot);
22
23    // led akan hidup/mati dengan durasi = nilai sensor
24    // jika nilai sensor 100 maka durasinya adalah 100ms
25    if(millis() > lastTime + sensor){
26      if(ledValue == LOW){
27        ledValue = HIGH;
28      }else{
29        ledValue = LOW;
30      }
31
32      // set lastTime dg nilai millis() yang baru
33      lastTime = millis();
34      digitalWrite(pinLED, ledValue);
35    }
36  }
```

Program pada Sketch 3.5 mempunyai fungsi yang sama dengan Sketch 3.4, bedanya, yang Sketch 3.5 versi responsive. Jadi ketika potensiometer diputar, maka durasi kedipan LED segera berubah. Cara kerjanya seperti ini:

Ketika berbicara tentang delay, maka kita berbicara tentang waktu. Pada Sketch 3.5, waktu delay bersifat dinamis karena parameternya adalah waktu sekarang (diambil dari *millis()*). Sedangkan jeda waktu untuk perubahan adalah waktu perubahan terakhir + nilai dari potensiometer. Untuk lebih memahami yang saya maksudkan, perhatikan Gambar 3.5.



Gambar 3.5 Siklus perubahan status LED dan potensiometer

Pada Gambar 3.5, keempat tahap dari atas ke bawah tersebut adalah kondisi nyala atau tidaknya LED. Nilai *lastTime* dihitung sejak terjadinya perubahan pada potensiometer atau habisnya jeda berdasarkan nilai sensor yang ditentukan.

1. Step pertama kondisi LED adalah ON, lastTime didapat dari perubahan terakhir atau pemanggilan terakhir *millis()*. Ketika nilai sensor (warna hijau) sudah habis, maka LED akan menjadi OFF, millis() akan dipanggil dan akan menjadi nilai lastTime pada step kedua.
2. Step kedua, kondisi LED OFF selama nilai sensor milisekon, setiap saat nilai millis() akan diperiksa. Garis putus-putus menandakan waktu pemanggilan fungsi millis() dalam pengkondisian

   if*(millis() > lastTime + sensor)*

   jika nilai *millis()* sudah lebih besar daripada lastTime + sensor, maka waktu OFF LED sudah habis dan digantikan dengan waktu ON LED.
3. Step ketiga menunjukkan bahwa nilai potensimeter dinaikkan sehingga durasi yang dibutuhkan untuk menyalakan LED lebih lama. Garis-garis putus menunjukkan bahwa proses

pemanggilan *millis()* dalam IF masih separuh perjalanan. Tapi ditengah perjalanan, nilai potensiometer dikecilkan.

4. Setelah potensiometer dikecilkan, maka durasi yang awalnya masih setengah perjalanan, kini tinggal sedikit dan LED masih dalam keadaan ON.

Semoga ilustrasi diatas bisa menjelaskan bagaimana cara kerja program pada Sketch 3.5. ☺

Tapi sebentar, pada baris 15 *lastTime* bukan int (*integer*). Tipe data *lastTime* adalah *long* (*long integer*). Apa bedanya dengan integer tanpa *long*?

```
15  long lastTime = 0;
```

Untuk menjelaskan itu, berikit ini adalah tabel bilangan dan kapasitasnya:

Tipe	**Kapasitas**
boolean	Hanya bisa TRUE & FALSE
char	-128 – 127
unsigned char	0 – 255
byte	(sama dengan unsigned char)
int	-32.768 – 32.767
unsigned int	0 – 65.535
word	(sama dengan unsigned int)
long (long int)	-2.147.483.648 – 2.147.483.647
unsigned log	0 – 4.294.967.295
float	-3.4028235E+38 – 3.4028235E+38
double	(sama dengan float)

Perhatikan bahwa kapasitas *long* lebih besar daripada *int*. Tabel di atas hanya berlaku untuk Arduino saja, jadi jika untuk komputer lainnya bisa jadi akan berbeda tergantung arsitektur dan kemampuan komputer tersebut dalam mencacah.

Beri aku 1.000 orang tua, niscaya akan kucabut semeru dari akarnya. Beri aku 10 pemuda niscaya akan kuguncangkan dunia!

(Ir Soekarno)

Bagian 4

Sound

Pada bagian ini kita akan bermain-main dengan suara. Sehingga kita akan membutuhkan speaker untuk membangkitkan suara dan nada musik sederhana. Pada dasarnya, untuk membuat speaker berbunyi maka kita harus menghidup-matikan speaker sesuai dengan frekuensi suara yang ingin kita bunyikan. Hidup-matinya speaker akan membuat *spool* speaker bergetar (bergerak maju-mundur) dan menghasilkan bunyi dengan nada tertentu.

Suara musik kelas A menengah sekitar 440 Hz. Masih ingat apa itu Hz? Hz merupakan kependekan dari Hertz. Hertz adalah jumlah siklus perdetik. Dengan demikian, jika kita ingin memainkan musik kelas A menengah, maka kita harus menyalakan dan mematikan speaker sebanyak masing-masing 440 kali dalam 1 detik.

Untuk menghidup-matikan speaker sebanyak masing-masing 440 kali, kita bisa memanfaatkan fungsi *delay()*. Sebelumnya kita telah membuat LED berkedip dengan memanfaatkan delay. Perlakuan kita terhadap LED akan kita terapkan pada speaker, tapi dengan tempo yang lebih cepat.

Cara menghitung delay yang kita butuhkan untuk mendapatkan siklus 440 Hz (nada 440Hz) yaitu dengan cara:

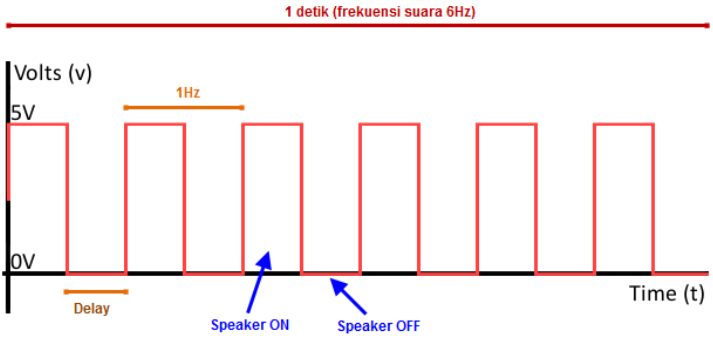
$$delay = \frac{1\ detik}{2 * frekuensi\_nada}$$

$$delay = \frac{1\ detik}{2 * 440}$$

$$delay = 0.001136\ detik = 1136\ \mu\ detik$$

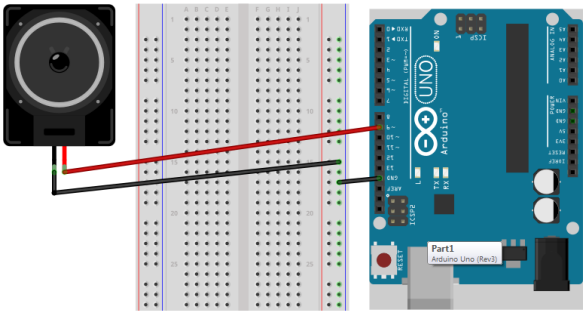
Kenapa frekuensi dikalikan 2? Gelombang suara merupakan gelombang analog (sinyal analog) yang merupakan gelombang sinus. Artinya, 1 siklus penuh adalah 1 tinggi/puncak dan 1 rendah/lembah. Kondisi tinggi adalah ketika speaker dinyalakan, sedangkan kondisi rendah adalah ketika speaker dimatikan. Oleh sebab itu, kita

membutuhkan 2 delay untuk 1 Hz. Karena 440 Hz adalah 440 siklus, maka setiap siklus pada 440 Hz dikalikan dengan 2. Semoga Gambar 4.1 memberikan pemahaman tentang bagaimana cara menentukan delay. Gambar 4.1 sekedar contoh sebab frekuensi 6 Hz tidak akan terdengar oleh telinga manusia.



Gambar 4.1 Siklus Frekuensi dan delay

4.1 Rangkaian



Rangkaian 4.1 Memasang speaker

Buatlah Rangkaian 4.1 dengan speaker 8 ohm atau 16 ohm. Speaker baru atau speaker bekas radio yang masih berfungsi bisa Anda gunakan. Anda bisa menggunakan *project board* atau tidak.

1. Sambungkan kaki positif speaker pada pin 9 board arduino
2. Sambungkan kaki negatif pada GND pada board arduino

3. Jika nanti suara yang dihasilkan terlalu nyaring, maka Anda bisa menambahkan resistor 100 – 1 k ohm pada kaki positif atau negatif speaker. Untuk itu, penggunaan *project board* akan memudahkan Anda untuk menyambung resistor tersebut. ☺

4.2 Membuat Nada

Jika kita kembali ke catatan pembuka pada bagian ini, maka untuk membuat bunyi / nada 440 Hz kita harus menghidup-matikan 440 kali dengan delay masing-masing 1136 mikrodetik. Perhatikan, satuannya adalah mikrodetik (1/1000 milidetik). Padahal fungsi *delay()* yang sering kita gunakan sebelumnya satuannya dalam milidetik. Oleh sebab itu, untuk membuat delay dengan satuan mikrodetik kita bisa menggunakan fungsi *delayMicroseconds()*.

Sketch 4.1 Membuat nada 440 Hz

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // speaker ada di pin 9
6  const int pinSpeaker = 9;
7  // delay untuk nada 440 Hz
8  const int timeDelay = 1163;
9
10 void setup() {
11   pinMode(pinSpeaker, OUTPUT);
12 }
13
14 void loop() {
15   digitalWrite(pinSpeaker, HIGH);
16   delayMicroseconds(timeDelay);
17   digitalWrite(pinSpeaker, LOW);
18   delayMicroseconds(timeDelay);
19 }
```

Ketika program Sketch 4.1 dijalankan, maka speaker akan berbunyi terus menerus hingga kabel speaker dilepas atau Arduino dimatikan. Begitulah cara kerja speaker berbunyi dan membuat program suara yang sederhana. Silakan Anda coba-coba dengan frekuensi lainnya supaya lebih paham.

4.3 Musik

Musik adalah kumpulan nada, sehingga jika kita ingin membuat musik, maka kita bisa merangkai nada-nada sehingga alunannya enak didengar. Pada Arduino kita bisa menggunakan fungsi *tone()* untuk membuat nada. Fungsi *tone()* memiliki 2 parameter inputan wajib dan 1 parameter tambahan. Cara menggunakan fungsi *tone()* yaitu:

tone(pin, frekuensi, durasi); atau *tone(pin, frekuensi);*

Parameter *pin* adalah pin yang disambungkan ke speaker, *frekuensi* adalah frekuensi yang digunakan, sedangkan *durasi* adalah lama nada berbunyi pada frekuensi tersebut.

Jika tanpa menginputkan durasi, maka nada akan dibunyikan hingga nada selanjutnya dijalankan atau ketika kita memberikan perintah *noTone()*. Sehingga kita bisa memanfaatkan delay untuk membuat nada yang panjang atau pendek.

Parameter *durasi* akan berguna ketika kita ingin membunyikan nada sambil menjalankan perintah lainnya. sebab jika kita menggunakan delay, maka kita harus menunggu delay selesai dahulu untuk menjalankan perintah selanjutnya.

Perintah *noTone()* berguna untuk menghentikan nada pada pin tertentu, sehingga kita bisa menggunakan perintah pin dengan format

noTone(pin);

Perintah *noTone()* akan berguna ketika kita menggunakan banyak speaker yang dikontrol oleh banyak pin. Sekedar catatan bahwa ketika kita menjalankan fungsi *tone()*, maka kita tidak bisa menggunakan fungsi PWM pada pin 3 dan pin 11. Oleh sebab itu, jika ingin menggunakan PWM dan fungsi *tone()*, sebaiknya Anda menggunakan pin lainnya untuk PWM.

Program pada Sketch 4.2 berfungsi untuk membuat tangga nada Do-Re-Mi kunci C. Silakan dicoba.

Sketch 4.2 Program Doremi

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
```

```
// coder elangsakti

// tangga nada C
#define NOTE_C4 262  // DO
#define NOTE_D4 294  // RE
#define NOTE_E4 330  // MI
#define NOTE_F4 349  // FA
#define NOTE_G4 392  // SOL
#define NOTE_A4 440  // LA
#define NOTE_B4 494  // SI
#define NOTE_C5 523  // DO

// speaker ada di pin 9
const int pinSpeaker = 9;

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

void loop() {
  tone(pinSpeaker, NOTE_C4, 500);
  delay(500);
  tone(pinSpeaker, NOTE_D4, 500);
  delay(500);
  tone(pinSpeaker, NOTE_E4, 500);
  delay(500);
  tone(pinSpeaker, NOTE_F4, 500);
  delay(500);
  tone(pinSpeaker, NOTE_G4, 500);
  delay(500);
  tone(pinSpeaker, NOTE_A4, 500);
  delay(500);
  tone(pinSpeaker, NOTE_B4, 500);
  delay(500);
  tone(pinSpeaker, NOTE_C5, 500);
  delay(500);

  noTone(pinSpeaker);
  delay(1000);
}
```

Pada Sketch 4.2, kita sudah mempraktekkan fungsi *tone()* dan *noTone()*. Di awal program ada bagian *#define* yang berfungsi untuk mengganti variabel tersebut dengan nilai yang dituju. Misal, variabel NOTE_C4 berarti isinya adalah angka 262.

Daftar nada tersebut merupakan daftar nada standar, Anda seharusnya sudah mendownloadnya bersama dengan ebook ini. Atau

Anda bisa mendapatkan informasinya di website Arduino[4]. Sketch 4.2 sebenarnya bisa kita sederhanakan lagi cara penulisannya dengan membuat satu fungsi untuk menjalankan *tone()*.

4.4 Membuat Fungsi

Dengan membuat fungsi, kita bisa mengurangi jumlah kode yang berulang-ulang. Dengan demikian, bisa mengurangi jumlah baris kode dan mempercepat pembuatan program. Sebelumnya kita sudah sering menggunakan fungsi bawaan Arduino, misal fungsi *delay()*, *tone()*, *noTone()*, dan *pinMode()*, *digitalWrite()*, *analogWrite()*, dan *analogRead()*.

Ketika membuat fungsi, kita harus menentukan tipe dari fungsi tersebut. Tipe fingsi bisa mengacu pada tipe-tipe data yang sudah ada, misal *int*, *float*, *long*, *char*, dst. Setiap fungsi harus memiliki nilai yang dikembalikan ke fungsi tersebut. maksudnya begini, jika kita membuat fungsi dengan tipe *int* (integer), artinya fungsi tersebut memiliki angka, fungsi tersebut harus memiliki value berupa integer. Jika tidak, maka akan error. Sama halnya ketika kita membuat satu variabel, maka variabel tersebut harus ada isinya (valuenya). Jika tidak ada valuenya, ya akan error juga. Fungsi juga sama dengan variabel, kecuali fungsi yang bertipe *void*.

Fungsi yang bertipe *void* tidak memiliki nilai kembalian, artinya fungsi tersebut hanya mejalankan perintah-perintah saja tanpa memiliki value sebagaimana fungsi dengan tipe *char*, *int*, *float*, dst. Untuk memberikan nilai kembalian untuk seuatu fungsi, maka bisa menambahkan perintah *return*. Perhatikan kedua fungsi berikut:

```
1  // fungsi bertipe integer
2  int customDelay(int x1, int x2){
3     int myDelay = x1 * x2;
4     return myDelay;
5  }
6
7
8  // fungsi bertipe void
9  void nyalakan_LED(int pin, int DELAY){
10    digitalWrite(pin, HIGH);
```

[4] *https://www.arduino.cc/en/tutorial/tone*

```
11      int timeDelay = customDelay(2, DELAY);
12      delay(timeDelay);
13  }
```

Fungsi *customDelay()* bertipe integer dan mempunyai 2 buah parameter integer sebagai masukan. *Value* dari *customDelay* adalah hasil perkalian dari kedua parameter tersebut yang diambil dari variabel *myDelay* (*return myDelay*). Variabel yang di-*return* menjadi value fungsi harus memiliki tipe data yang sama dengan tipe fungsi tersebut. Dalam contoh di atas, fungsi *customDelay* adalah integer dan *myDelay* juga integer.

Fungsi *nyalakan_LED()* merupakan fungsi bertipe void sehingga fungsi itu tidak memiliki value, fungsi ini hanya mengeksekusi perintah di dalamnya. Fungsi ini memiliki 2 parameter, yaitu pin dan DELAY yang sama-sama integer. Dalam fungsi tersebut ada perintah untuk menyalakan led pada pin *pin*, kemudian nilai *timeDelay* diambil dari hasil perkalian di dalam fungsi *customDelay()*. Perhatikan, value dari *timeDelay* diambil dari fungsi *customDelay()*. Yang paling penting, fungsi void tidak butuh *return*.

Mari kita mengubah Sketch 4.2 agar lebih singkat. ☺

Sketch 4.3 Program Doremi dengan fungsi

```
1   // Free Ebook Arduino
2   // www.elangsakti.com
3   // coder elangsakti
4
5   // tangga nada C
6   #define NOTE_C4 262  // DO
7   #define NOTE_D4 294  // RE
8   #define NOTE_E4 330  // MI
9   #define NOTE_F4 349  // FA
10  #define NOTE_G4 392  // SOL
11  #define NOTE_A4 440  // LA
12  #define NOTE_B4 494  // SI
13  #define NOTE_C5 523  // DO
14
15  // speaker ada di pin 9
16  const int pinSpeaker = 9;
17
18  void setup() {
19    pinMode(pinSpeaker, OUTPUT);
20  }
```

```
21
22  void loop() {
23    nada(NOTE_C4, 500);
24    nada(NOTE_D4, 500);
25    nada(NOTE_E4, 500);
26    nada(NOTE_F4, 500);
27    nada(NOTE_G4, 500);
28    nada(NOTE_A4, 500);
29    nada(NOTE_B4, 500);
30    nada(NOTE_C5, 500);
31
32    noTone(pinSpeaker);
33    delay(1000);
34  }
35
36  void nada(int frek, int durasi){
37    tone(pinSpeaker, frek, durasi);
38    delay(durasi);
39  }
```

Pada Sketch 4.3, kode program lebih kita sederhanakan dengan membuat fungsi *nada()*. Sehingga kita tidak perlu memanggil fungsi delay berulang kali. Mungkin awalnya akan agak lebih kesulitan untuk memahami sebab kodenya tidak sekuensial. Tapi model program di atas lebih mudah apabila kita membuat aplikasi dengan jumlah data NOTE yang lebih banyak lagi jika menggunakan Sketch 4.1.

Sketch 4.3 memiliki jumlah variasi hanya 8 nada, bisa Anda bayangkan jika musik yang akan Anda buat memiliki jumlah nada puluhan atau ratusa, kira-kira akan sepanjang apa program Anda? ☺

Oleh sebab itu, Sketch 4.3 harus dibuat lebih fleksibel lagi. Caranya? Kita bisa memanfaatkan array sebagaimana pernah kita pelajari ketika membuat LED, apakah Anda masih ingat? Saya punya 2 contoh potongan lagu yang bisa menjadi gambaran bahwa ketika nadanya bertambah banyak, maka kerumitan akan semakin menantang. ☺

Catatan untuk Sketch 4.4 dan Sketch 4.5, Anda harus membuat file ***pitches.h*** atau mendownloadnya dulu dari link yang sudah saya berikan di halaman-halaman sebelumnya. Perhatikan pada Sketch 4.4 pada baris kedua, di sana ada perintah *#include <pitches.h>*, jadi Anda harus punya file *pitches.h* untuk menjalankan aplikasi tersebut.

Sketch 4.4 Program Twinke-twinkle

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // referensi tangga nada
6  #include <pitches.h>
7
8  // speaker ada di pin 9
9  const int pinSpeaker = 9;
10
11 #define JUMLAH_NADA 15
12
13 const int daftar_nada[JUMLAH_NADA] = {
14   NOTE_C4, NOTE_C4, NOTE_G4, NOTE_G4,
15   NOTE_A4, NOTE_A4, NOTE_G4, NOTE_F4,
16   NOTE_F4, NOTE_E4, NOTE_E4, NOTE_D4,
17   NOTE_D4, NOTE_C4, 0
18 };
19 const int lama_beat = 300;
20 const int beats[JUMLAH_NADA] = {
21     1, 1, 1, 1,
22     1, 1, 2, 1,
23     1, 1, 1, 1,
24     1, 2, 4
25 };
26
27 void setup() {
28   pinMode(pinSpeaker, OUTPUT);
29 }
30
31 void loop() {
32   for(int i=0; i<JUMLAH_NADA; i++){
33     if(nada[i] == 0){
34       delay(beats[i] * lama_beat);
35     }else{
36       nada(daftar_nada[i], beats[i] * lama_beat);
37     }
38     // jeda antar nada
39     noTone(pinSpeaker);
40     delay(lama_beat / 2);
41   }
42 }
43
44 void nada(int frek, int durasi){
45   tone(pinSpeaker, frek, durasi);
46   delay(durasi);
47 }
```

Sketch 4.5 Program Garuda Pancasila

```
// Free Ebook Arduino
// www.elangsakti.com
// coder elangsakti

// referensi tangga nada
#include <pitches.h>

// speaker ada di pin 9
const int pinSpeaker = 9;

#define JUMLAH_NADA 85

const int daftar_nada[JUMLAH_NADA] = {
  NOTE_G4, NOTE_G4, NOTE_C5, NOTE_C5, NOTE_D5, NOTE_D5,
     NOTE_E5, 0,
  NOTE_E5, NOTE_F5, NOTE_G5, NOTE_C5, NOTE_D5, NOTE_E5,
     NOTE_F5, NOTE_D5, 0,
  NOTE_G4, NOTE_G4, NOTE_D5, NOTE_D5, NOTE_E5, NOTE_E5,
     NOTE_F5, 0,
  NOTE_E5, NOTE_D5, NOTE_C5, NOTE_G4, NOTE_G4, NOTE_G4,
     NOTE_A4, NOTE_B4, NOTE_C5, 0,
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_A4, NOTE_C5, NOTE_F5,
     NOTE_G5, NOTE_A5, NOTE_G5, 0,
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_A4, NOTE_C5, NOTE_F5,
     NOTE_G5, NOTE_A5, NOTE_G5, 0,
  NOTE_G5, NOTE_A5, NOTE_G5, NOTE_F5, NOTE_E5, NOTE_D5, 0,
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5, NOTE_A4, NOTE_G4, 0,
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5, NOTE_D5, NOTE_E5, 0,
  NOTE_C5, NOTE_C5, NOTE_A5, NOTE_G5, NOTE_B4, NOTE_C5, 0
};

const int lama_beat = 300;
const int beats[JUMLAH_NADA] = {
  1, 1, 2, 2, 2, 2, 3, 1,
  1, 1, 2, 1, 1, 2, 2, 3, 1,
  1, 1, 2, 2, 2, 2, 3, 1,
  1, 1, 2, 1, 1, 2, 1, 1, 3, 1,
  1, 1, 2, 1, 1, 2, 1, 1, 3, 1,
  1, 1, 2, 1, 1, 2, 1, 1, 3, 1,
  2, 3, 1, 2, 2, 3, 1,
  1, 1, 1, 3, 1, 1, 1,
  1, 1, 1, 3, 1, 1, 1,
  2, 2, 2, 3, 1, 3, 0
};

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

```

```
50  void loop() {
51    for(int i=0; i<JUMLAH_NADA; i++){
52      if(nada[i] == 0){
53        delay(beats[i] * lama_beat);
54      }else{
55        nada(daftar_nada[i], beats[i] * lama_beat);
56      }
57      // jeda antar nada
58      noTone(pinSpeaker);
59      delay(lama_beat / 2);
60    }
61  }
62
63  void nada(int frek, int durasi){
64    tone(pinSpeaker, frek, durasi);
65    delay(durasi);
66  }
```

Perhatikan, antara Sketch 4.4 dan 4.5 hanya berbeda jumlah nada, nada, dan beats-nya. Jadi program utama di dalam fungsi loop tidak ada yang berubah. Kenapa? Karena program tersebut sudah fleksibel dengan adanya fungsi dan bantuan array. Coba bayangkan jika kedua program tersebut dibuat secara manual. Lumayan ribet tentunya. ☺

Gantungkan cita-cita mu setinggi langit! Bermimpilah setinggi langit. Jika engkau jatuh, engkau akan jatuh di antara bintang-bintang.

(Ir Soekarno)

Bagian 5

Termometer Digital

Sebelum kita membuat termometer digital, kita akan belajar bagaimana menggunakan *Serial Monitor* sebagai alat untuk melihat apakah sensor menghasilkan data yang benar atau tidak. Maksudnya begini, ketika kita menggunakan sebuah sensor untuk mengambil data, maka sensor akan mengirimkan data untuk diproses oleh mikrokontroller. Untuk memastikan data dari sensor tersebut merupakan data yang benar, maka kita bisa melihatnya melalui *Serial Monitor*. Cara kerjanya begini:

1. Mikrokontroller akan membaca data dari sensor
2. Kemudian mikrokontroller akan membuat koneksi serial ke komputer
3. Selanjutnya mikrokontroller akan mengirimkan data ke komputer melalui komunikasi serial tersebut
4. Lalu kita bisa melihat data yang diterima oleh komputer menggunakan serial monitor, *hyperterminal*, atau aplikasi sejenis seperti *CoolTerm* dan *PuTTY*[5].

5.1 Serial Monitor

Serial monitor bisa kita gunakan untuk men-*debug* secara *software*. Jika tanpa *serial monitor*, kita tidak bisa melakukan *debug* untuk aplikasi yang kita buat sehingga untuk menemukan solusinya, kita harus men-debug dari sisi *hardware*. Misal ketika ada error, kita akan mencoba dengan LED atau menambah/mengurangi rangkaian.

Tapi jika menggunakan serial monitor, kita akan tahu error-nya melalui data yang dikirimkan oleh Arduino. Misal ketika nyala LED terlalu lama atau terlalu pelan, kita langsung bisa mengecek nilai (angka) yang digunakan untuk delay dan semua isi variabel dalam program yang kita buat. Sehingga kita bisa menelusuri logika dan algoritma program berdasarkan data-data yang dikirimkan tadi.

[5] *http://goo.gl/NFdgQo*

Perhatikan pada board arduino, pin 0 dan 1 ada tulisan RX dan TX. Pin tersebut berfungsi untuk menerima dan mengirim data melalui komunikasi serial dari Arduino ke komputer melalui kabel USB. Untuk menggunakan komunikasi serial, kita tidak perlu menambahkan komponen tambahan pada Arduino karena pada board tersebut sudah disediakan. Kita cukup menghubungkan Arduino ke komputer, dan kita bisa langsung membuat program. Mari kita mulai dengan Sketch 5.1.

Sketch 5.1 Komunikasi Serial

```
1   // Free Ebook Arduino
2   // www.elangsakti.com
3   // coder elangsakti
4
5   void setup() {
6     Serial.begin(9600);
7     Serial.println("With elangsakti.com :");
8   }
9
10  int number = 0;
11  void loop() {
12    Serial.print("Hello World! ");
13    Serial.println( number++ );
14    delay(1000);
15  }
```

Sebelum kita melihat hasilnya, mari kita sedikit membahas program pada Sketch 5.1.

```
6     Serial.begin(9600);
```

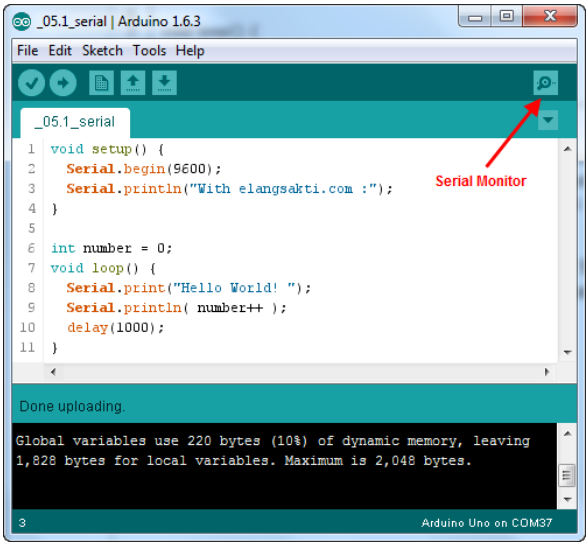
Pada baris ke-6, perintah *Seril.begin(9600);* berarti kita akan membuat koneksi serial dengan ***baud rate*** 9600. Sederhananya, **baud** berkaitan dengan jumlah bit yang akan ditransfer setiap detik. Nilai *baud rate* ini tergantung pada clock mikrokontroller. Arduino Uno sendiri menggunakan clock 16 MHz. Jika clock-nya makin rendah, maka baud rate harus kita kurangi. Sebagai contoh, jika kita menggunakan mikrokontroller dengan clock 1 MHz, maka baud rate yang cocok adalah 4800. Jika clock 1 MHz kita menggunakan baud rate 9600, maka data yang dikirim ke komputer tidak akan terbaca. Pada Arduino IDE, nilai baud rate default pada Serial Monitor adalah 9600.

Silakan Anda coba-coba dengan mengubah nilainya sesuai pilihan yang ada.

Pada baris ke-7, awalnya Arduino akan mengirim pesan "*With elangsakti.com*" ketika pertama kali Arduino *start*. Setelah itu, Arduino akan mengirim pesan "*Hello World!*" dan dilanjutkan dengan angka 0, 1, 2, 3, dst. Jika Serial Monitor kita tutup dan kita buka kembali, maka Arduino seakan-akan melakukan reset program. Arduino akan kembali mengirimkan pesan "*With elangsakti.com*" dan angka kembali ke 0 lagi.

```
12    Serial.print("Hello World! ");
13    Serial.println( number++ );
```

Pada baris ke 12 dan 13, kita menemukan perintha print yang berbeda, yaitu *print()* dan *println()*. Jika kita menggunakan print, maka kita sekedar mengirim data tanpa diikuti *end of line* (perintah untuk ganti baris , atau ENTER). Jika kita menggunakan *println()*, maka tulisan akan diikuti perintah untuk pindah baris.

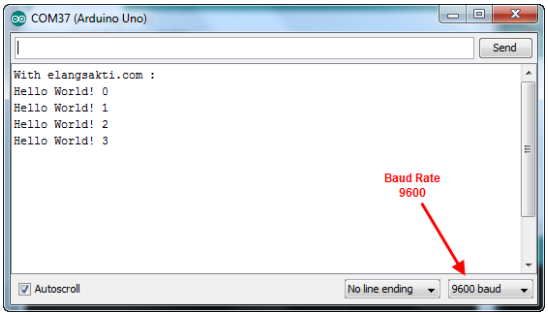


Gambar 5.1 Icon Serial Monitor

Kita kembali ke board Arduino. Setelah program Sketch 5.1 diupload ke Arduino. Perhatikan board Arduino, maka LED yang berlabel TX akan berkedip. Setiap LED TX berkedip, artinya Arduino

sedang mengirim data. Buka Serial Monitor dengan cara mengklik icon *Serial Monitor* (lihat Gambar 5.1). Jika Anda menggunakan versi Arduino IDE selain versi 1.63, mungkin letaknya berbeda.

Sebelum muncul pesan "*Done Uploading*", maka *Serial Monitor* belum bisa dibuka. Setelah *Serial Monitor* dibuka, maka setidaknya Anda akan melihat pesan seperti pada Gambar 5.2.



Gambar 5.2 Tampilan serial monitor

5.1.1 Tracking timeDelay

Kali ini kita akan men-*tracking timeDelay* dan mengirimkan pesan ke komputer bahwa LED sedang nyala atau mati. LED yang akan kita gunakan untuk percobaan adalah LED yang ada pada *board* Arduino yang terhubung ke pin 13, perhatikan LED dengan label "L" pada board Arduino.

Sketch 5.2 Program tracking timeDelay

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  const int LED = 13;
6  int timeDelay = 3000;
7
8  void setup() {
9    Serial.begin(9600);
10
11   Serial.println("pin 13 as OUTPUT");
12   pinMode(LED, OUTPUT);
13
```

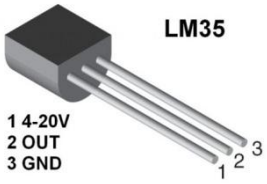
```
  Serial.println("LED = ON, selama timeDelay = 3 detik");
  digitalWrite(LED, HIGH);
  delay(timeDelay);
}

void loop() {
  if(timeDelay <= 100){
    Serial.println("Reset timeDelay to 1000");
    timeDelay = 1000;
  }

  Serial.println();
  Serial.print("timeDelay = ");
  Serial.println(timeDelay);

  Serial.println("LED = OFF");
  digitalWrite(LED, LOW);
  delay(timeDelay);
  Serial.println("LED = ON");
  digitalWrite(LED, HIGH);
  delay(timeDelay);

  timeDelay = timeDelay - 100;
}
```

Program pada Sketch 5.2 merupakan contoh simpel untuk melakukan *debugging*, artinya kita ingin mengetahui apa yang dilakukan program dengan mengetahui setiap variabel dan bagaimana perilaku program tersebut. Setidaknya program pada Sketch 5.2 memiliki output seperti berikut:

```
pin 13 as OUTPUT
LED = ON, selama timeDelay = 3 detik

timeDelay = 3000
LED = OFF
LED = ON

timeDelay = 2900
LED = OFF
LED = ON

timeDelay = 2800
LED = OFF
LED = ON

dst...
```

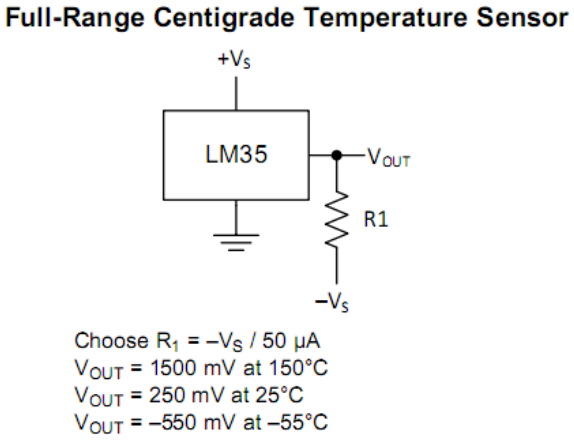
5.2 Mengukur Suhu dengan LM 35

LM35 merupakan IC sensor suhu dengan bentuk yang mirip dengan transistor. Kaki IC ini hanya ada tiga, yaitu untuk VCC, Output, dan GND (Gambar 5.3).



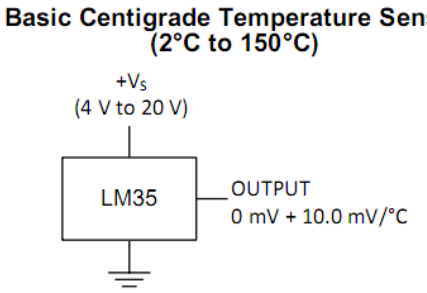
Gambar 5.3 Sensor suhu LM35

Sensor ini bisa digunakan untuk mengukur suhu dari -55° – 150° celcius. Berdasarkan datasheet LM35[6], maka kita bisa menggunakan pengukuran penuh (-55 – 150° celcius) atau pengukuran sebagian yaitu hanya bisa menghitung dari 2 – 150° celcius. Untuk pengukuran penuh, maka rangkaian dasarnya seperti tampak pada Gambar 5.4 sedangkan untuk pengukuran sebagian, rangkaian dasarnya adalah seperti pada Gambar 5.5.



Gambar 5.4 Rangkaian dasar pengukuran suhu penuh LM35

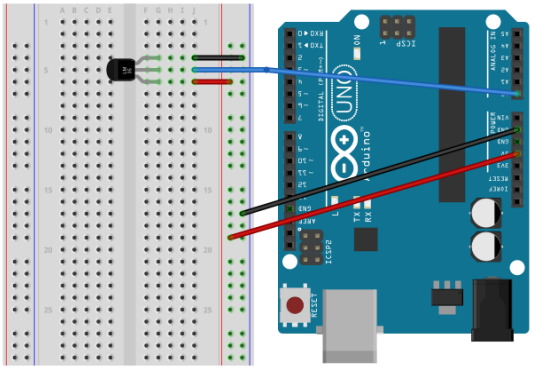
[6] *http://www.ti.com/lit/ds/symlink/lm35.pdf*



Gambar 5.5 Rangkaian dasar pengukuran suhu sebagian LM35

5.2.1 Rangkaian

Berdasarkan karakteristik kaki-kaki pada IC LM35, maka kita akan menggunakan rangkaian sebagian sehingga Rangkaian 5.1 hanya bisa mengukur suhu dari 2 hingga 150 derajat celcius. Cara merangkainya yaitu:



Rangkaian 5.1 Rangkaian sensor suhu LM35

1. Sambungkan kaki 1 ke VCC
2. Sambungkan kaki kedua (tengah) ke A0. A0 adalah pin analog, kaki pin analog berfungsi untuk berbagasi tranduser / sensor yang mengharuskan sinyal analog. Oleh sebab itu, untuk membaca kaki ini menggunakan *analogRead()*, sedangkan untuk menulisnya menggunakan *analogWrite()*.
3. Sambungkan kaki ke-3 ke GND.

Karakteristik dari sensor ini yaitu setiap kenaikan 10 mV pada kaki output, menandakan kenaikan suhu 1° celcius. Sehingga, karena Rangkaian 5.1 hanya mampu mengukur dari 2° celcius, maka output LM35 minimal adalah 20 mV dan maksimal 1500 mV. Konversi suhu pada output LM35 juga tergantung pada tegangan referensi yang digunakan.

Tegangan referensi pada arduino ada tiga (khusus Arduino Uno)[7], tegangan referensi default, internal, dan eksternal. Jika kita tidak mendefinisikan tegangan referensi yang akan kita gunakan, maka Arduino secara default akan menggunakan tegangan referensi 5 volt. Selain 5 volt, tegangan default yang disediakan oleh arduino adalah 3.3 volt. Akan tetapi kita harus membuat jumper dari 3.3 volt (di board Arduino) ke pin AREF, lalu mengeksekusi perintah *analogReference(DEFAULT)*.

Tegangan referensi internal Arduino yaitu 1.1 volt, untuk menggunakan tegangan referensi ini, kita harus memberikan perintah *analogReference(INTERNAL)*.

Tapi jika ingin menggunakan tegangan referensi selain 5, 3.3, dan 1.1 volt, kita bisa menggunakan tegangan referensi eksternal. Tegangan referensi ini harus antara 0 dan 5 volt, jika tidak, Arduino bisa jadi akan rusak. Jika kita menggunakan tegangan referensi custom ini, maka kita harus memasang sumber tegangan ke AREF dan memberi perintah *analogReference(EXTERNAL)*;

5.2.2 Program

Sebelum membuat program, kita akan menghitung bagaimana cara mengukur dan mengkonversi output dari LM35 menjadi suhu. Kita akan mengkonversi voltase pada kaki output LM35, kemudian menghitungnya berdasarkan tegangan referensi yang digunakan, mengubahnya menjadi celcius, lalu mengirimkannya ke komputer melalui komunikasi serial.

Jika kita menggunakan tegangan referensi 5 volt, maka Arduino bisa mengukur setidaknya hingga 5000 mV. padahal kemampuan LM35 hanya sebatas 150° celcius atau 150 x 10 mV = 1500 mV (1.5

[7] *https://www.arduino.cc/en/Reference/AnalogReference*

volt). Sehingga tegangan yang keluar dari kaki output LM35 tidak akan mungkin melebihi 1.5 volt.

Berdasarkan persamaan sederhana, maka kita bisa menghitung suhu berdasarkan perbandingan antara kapasitas voltase yang bisa dicacah oleh pin analog Arduino (1024) dan kemampuan LM35 mengukur suhu.

Suhu dalam Voltase (T) : 0 - 500

Cacahan Voltase input (Vin) : 0 - 1024

0/500 = 0/1024

T/500 = Vin/1024

T = (Vin * 500) / 1024

Sketch 5.3 Program sensor suhu LM35

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  const int pSuhu = A0;
6  float suhu, data;
7
8  void setup() {
9    Serial.begin(9600);
10   pinMode(pSuhu, INPUT);
11 }
12
13 void loop() {
14   data = analogRead(pSuhu);
15   suhu = data * 500 / 1024;
16
17   Serial.print("data: ");
18   Serial.print(data);
19   Serial.print(", suhu: ");
20   Serial.print(suhu);
21   Serial.println();
22   delay(1000);
23 }
```

Program pada Sketch 5.3 akan membaca data dari sensor suhu pada pin A0 di board Arduino kemudian mengkonversinya menjadi

suhu. Informasi suhu akan dikirim ke komputer melalui komunikasi serial dengan *baud rate* 9600 setiap 1000 milisekon.

```
6   float suhu, data;
```

Variabel *suhu* dan *data* menggunakan *float*, yaitu tipe data yang memungkinkan memuat angka desimal. Di sini menggunakan desimal karena adanya pembagian sehingga jika kita menggunakan integer, maka hasil perhitungan kita kurang presisi karena hasil pembagiannya akan selalu dibulatkan.

```
14    data = analogRead(pSuhu);
```

Fungsi *analogRead()* digunakan untuk membaca masukan dari sensor analog. Nilai dari analog read ini berkisar dari 0 hingga 1023 berdasarkan kemampuan dari mikrokontroller dalam mencacah dari 0 – 5 volt.

Untuk mendapatkan hasil pengukuran yang lebih presisi, maka kita bisa mengganti tegangan referensi yang digunakan. Jika kita menggunakan tegangan referensi 5000 mV, maka space dari 1500 – 5000 mV tidak akan pernah terpakai. Oleh sebab itu, kita bisa menggunakan tegangan referensi 1500 mV (sesuai dengan output maksimal pada LM35) atau menggunakan tegangan referensi yang lebih rendah, misal tegangan referensi INTERNAL yang nilainya adalah 1.1 volt. Sebagai catatan, jika Anda menggunakan tegangan referensi 1.1 volt (1100 mV), maka batas maksimal suhu yang bisa dihitungan adalah 110° celcius.

Sketch 5.4 Program sensor suhu tegangan referensi 1.1 volt

```
1   // Free Ebook Arduino
2   // www.elangsakti.com
3   // coder elangsakti
4
5   const int pSuhu = A0;
6   float suhu, data;
7
8   void setup() {
9     // mengubah tegangan referensi ke internal, 1.1 volt
10    analogReference(INTERNAL);
11
```

```
  Serial.begin(9600);
  pinMode(pSuhu, INPUT);

}

void loop() {
  data = analogRead(pSuhu);
  suhu = data * 110 / 1024;

  Serial.print("data: ");
  Serial.print(data);
  Serial.print(", suhu: ");
  Serial.print(suhu);
  Serial.print(" C (");
  Serial.print(convertToF(suhu));
  Serial.print(" F)");
  Serial.println();
  delay(1000);
}

float convertToF(float suhuC){
  return (suhuC * 9.0/5.0) + 32;
}
```

Program pada Sketch 5.4 menggunakan tegangan referensi internal 1.1 volt kemudian suhu dalam celcius dikonversi menjadi Fahrenheit. Berdasarkan konsepnya, konversi celcius ke Fahrenheit menggunakan rumus:

$$F = \left(Celcius * \frac{9}{5}\right) + 32$$

Kemudian informasi suhu dalam celcius dan Fahrenheit dikirim ke komputer dengan komunikasi serial.

5.3 Memasang LCD

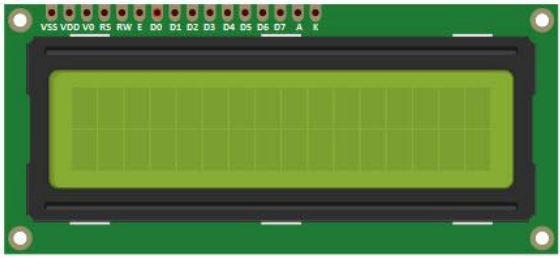
Dalam hal ini kita akan menyiapkan LCD untuk menampilkan informasi suhu yang telah kita buat. Sebab melihat informasi suhu dengan komputer tentu kurang praktis bukan? ☺

LCD merupakan singkatan dari Liquid Crystal Display, atau umumnya disebut dengan LCD atau display saja. Di pasaran beragam jenis LCD dan berbagai ukuran yang bisa Anda gunakan. LCD bisa

untuk menampilkan huruf dan angka, bahkan ada yang bisa untuk menampilkan gambar.

Dalam ebook ini, kita akan berkenalan dengan LCD yang umum digunakan dan harganya juga relatif terjangkau. LCD ini berukuran 16x2 (2 baris 16 kolom) yang cukup untuk menampilkan informasi suhu atau informasi yang tidak terlalu panjang. LCD ini dikenal juga dengan LCD 1602 dengan beberapa varian seperti 1602A, dll.

LCD ini bisa bekerja pada 5 volt, sehingga Anda bisa menyambungkannya secara langsung ke pin VCC pada board Arduino. Perlu diperhatikan, jika Anda menggunakan LCD jenis lainnya, ada juga LCD yang bekerja pada voltase yang berbeda. Sehingga kesalahan pemasangan sumber tegangan bisa membuat LCD rusak.



Gambar 6.6 LCD 1602

LCD 1602 memiliki 16 pin dengan fungsi-fungsi sebagai berikut:

Simbol	**Value**	**Fungsi**
VSS	0V	Ground
VDD	+5V	Power Supply / VCC
V0	-	Pengaturan kontras backlight
RS	H/L	H = data, L = command
R/W	H/L	H = read, L = write
E	H.H - L	Enable Signal
D1-D3	H/L	Jalur untuk transfer 8 bit data
D4-D7	H/L	Jalur untuk transfer 4 & 8 bit data
A	+5V	VCC untuk backlight
K	0V	GND untuk backlight

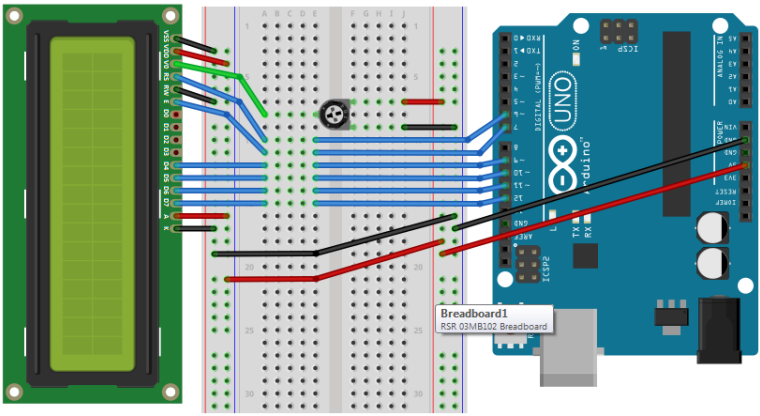
Berdasarkan karakteristik tersebut, maka semua pin akan digunakan kecuali pin D1 – D3 sebab kita akan menggunakan jalur

data untuk transfer 4 bit atau 8 bit. Penjelasan singkat tentang RS, R/W, dan E:

- RS merupakan kependekan dari *Register Selector*, pin ini berfungsi untuk memilih register control atau register data. Register control digunakan untuk mengkonfigurasi LCD, sedangkan register data digunakan untuk menuliskan data berupa karakter untuk ditampilkan di LCD.
- R/W atau *Read/Write*, digunakan untuk memilih aliran data mikrokontroller akan membaca data yang ada di LCD atau menuliskan data ke LCD. Jika LCD hanya digunakan untuk menulis / menampilkan data, maka pin ini bisa langsung disambungkan ke GND sehingga logika bisa diset menjadi L (Low).
- E atau *Enable*, digunakan untuk mengaktifkan LCD ketika proses penulisan data ke register control dan regiter data.

5.3.1 Rangkaian Dasar LCD 1602

Untuk merangkai LCD, yang Anda butuhkan adalah beberapa kabel jumper dan sebuah potensiometer. Potensiometer ini berfungsi untuk mengatur kontras *backlight* LCD. Perhatikan Rangkaian 5.2.



Rangkaian 5.2 Menghubungkan LCD 1602 ke Arduino

1. Pin V0 pada LCD disambungkan ke kaki tengah potensiometer, sementara masing-masing kaki potensiometer

yang ada di pinggir disambungkan ke VCC dan GND. Jika nanti tampilan tulisannya kurang jelas, silakan putar-putar potensiometernya.

2. Pin R/W pada LCD disambungkan ke GND
3. Pin RS pada LCD disambungkan ke pin 6 pada Arduino
4. Pin E pada LCD disambungkan ke pin 7 pada Arduino
5. Pin untuk data (D4 – D7) pada LCD disambungkan ke pin 9 – 12 pada Arduino
6. VDD dan A pada LCD disambungkan ke +5v
7. VSS dan K pada LCD disambungkan ke GND

5.3.2 Program Dasar LCD

Sketch 5.5 Program LCD dasar

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  #include <LiquidCrystal.h>
6
7  // Setting LCD   RS E D4  D5  D6  D7
8  LiquidCrystal lcd(7, 6, 9, 10, 11, 12);
9
10 void setup(){
11   // pilih LCD 16 x 2
12   lcd.begin(16,2);
13   lcd.print("ELANGSAKTI.COM");
14 }
15
16 int hitung = 0;
17 void loop(){
18   // pindah ke baris kolom 1 baris ke 2
19   // array selalu dimulai dari 0
20   lcd.setCursor(0,1);
21   lcd.print( hitung++ );
22   delay(1000);
23 }
```

Pada Arduino kita bisa menggunakan library untuk menulis LCD. Library tersebut adalah *LiquidCrystal.h*. Untuk menggunakan library tersebut Anda bisa meng-*include*-nya seperti pada baris ke-8.

```
7  // Setting LCD   RS E D4 D5 D6  D7
```

```
8  LiquidCrystal lcd(7, 6, 9, 10, 11, 12);
```

Perintah pada baris 8 adalah untuk melakukan setting LCD sesuai dengan library yang dipakai. Fungsi *lcd()* adalah instansiasi dari class LiquidCrystal yang dipanggila pada baris ke-8 memiliki parameter berturut-turut pin RS, E, D4, D5, D6, dan D7. Jika Anda ingin menggunakan formal lainnya, maka Anda bisa menggunakan beberapa pilihan parameter seperti di bawah ini:

```
LiquidCrystal(RS, E, D0, D1, D2, D3);
LiquidCrystal(RS, RW, E, D0, D1, D2, D3);
LiquidCrystal(RS, E, D0, D1, D2, D3, D4, D5, D6, D7);
LiquidCrystal(RS, RW, E, D0, D1, D2, D3, D4, D5, D6, D7);
```

Selanjutnya perhatikan baris ke-12

```
11   // pilih LCD 16 x 2
12   lcd.begin(16,2);
```

Fungsi *begin(kolom, baris, charsize)* memiliki 2 parameter wajib dan 1 parameter tambahan. Parameter wajibnya adalah informasi kolom dan informasi baris pada LCD. Sedangkan informasi tambahannya adalah *charsize* untuk masing-masing karakter yang akan digunakan. Standarnya, *charsize* berukuran 5x8 pixel untuk setiap karakter, jika Anda memiliki LCD dengan pixel yang berbeda, silakan disesuaikan.Fungsi *begin()* juga digunakan untuk inisialisasi jenis LCD yang akan digunakan. Mungkin ini agak mirip dengan *Serial.begin()* yang berfungsi untuk inisialisasi komunikasi serial dengan parameter berupa *baud rate* yang akan digunakan.

```
18   // pindah ke baris kolom 1 baris ke 2
19   // array selalu dimulai dari 0
20   lcd.setCursor(0,1);
21   lcd.print( hitung++ );
```

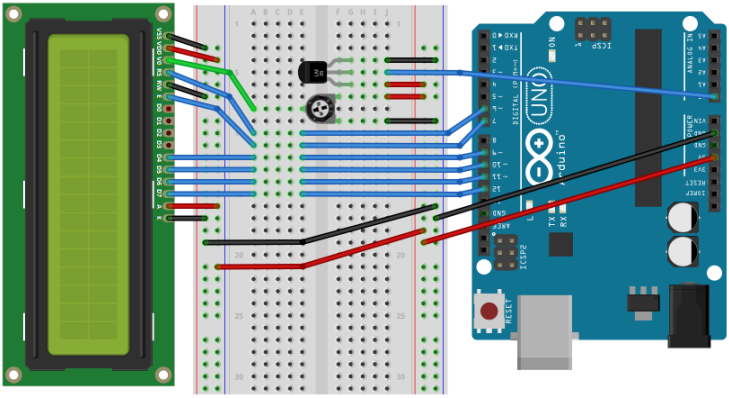
Fungsi *setCursor(kolom, baris)* berfungsi untuk memindah kursor aktif sesuai kolom dan baris yang kita tentukan. Kolom ke-0 berarti kolom ke 1 pada LCD (ingat, angka pertama dalam array adalah 0, bukan 1), sedangkan baris ke-1 berarti adalah baris ke 2 pada LCD.

Pada baris ke-21, kenapa menggunakan *print()* dan bukan *println()*? Penggunaan *println()* dalam LCD maka akan secara otomatis menambahkan 2 karakter tambahan untuk pindah baris (CR dan LF), maka jika Anda menggunakan *println()*, akan muncul 2 karakter aneh pada LCD. Jadi, gunakanlah *print()* untuk menulis ke LCD.

5.4 Sensor Suhu dengan LCD

Setelah kita tahu bagaimana cara membuat sensor suhu dengan IC LM35 dan cara memasang LCD, mari kita gabungkan keduanya sehingga menjadi termometer digital.

5.4.1 Rangkaian



Rangkaian 5.3 Termometer digital Arduino

Gabungkan sensor suhu pada Rangkaian 5.2 ke Rangkaian 5.3 sehingga terbentuk Rangkaian 5.4. Dalam hal ini, rangkaian sudah siap untuk diprogram sehingga menjadi termometer digital.

5.4.2 Program

Sketch 5.6 Program termometer digital

```
1  // Free Ebook Arduino
```

```
// www.elangsakti.com
// coder elangsakti

// Termometer digital

#include <LiquidCrystal.h>

const int pSuhu = A0;
float suhu, data;

// Setting LCD    RS E D4  D5  D6  D7
LiquidCrystal lcd(7, 6, 9, 10, 11, 12);

void setup(){
  // mengubah tegangan referensi ke internal, 1.1 volt
  analogReference(INTERNAL);
  // pinSuhu sebagai input
  pinMode(pSuhu, INPUT);

  // pilih LCD 16 x 2
  lcd.begin(16,2);
  lcd.print("ELANGSAKTI.COM");
}


void loop(){
  data = analogRead(pSuhu);
  suhu = data * 110 / 1024;

  // pindah ke baris kolom 1 baris ke 2
  // array selalu dimulai dari 0
  lcd.setCursor(0,1);
  lcd.print("Suhu: ");
  lcd.print(suhu);
  lcd.print("C");
  delay(1000);
}
```

Selamat, Anda sudah bisa membuat termometer digital. Selanjutnya, silakan Anda berkreasi dengan mengubah posisi atau mengubah tulisan yang ada di LCD. Silakan juga Anda mencoba untuk melakukan konversi dari celcius ke Fahrenheit, ke Kelvin, dan Reamur kemudian tampilkan di LCD. ☺

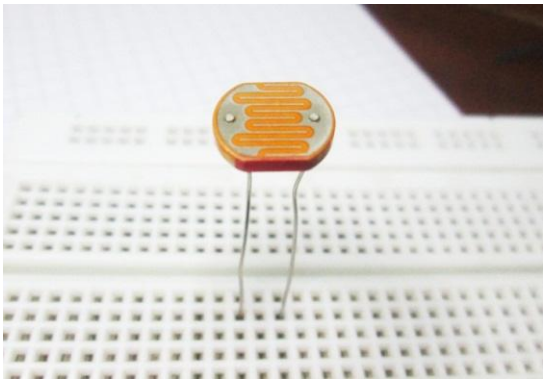
Kami menggoyakan langit, menggempakan darat, menggelorakan samudera, agar tidak menjadi bangsa yang hidup hanya dari 2 ½ sen sehari, Bangsa yg kerja keras, bukan bangsa tempe, bukan bangsa kuli, Bangsa yang rela menderita demi pembelian cita-cita.

(Ir Soekarno)

Bagian 6

Sensor Cahaya

Salah satu jenis sensor cahaya adalah LDR (*Light Dependent Resistor*). Dengan sensor ini, kita bisa membuat alat yang berkaitan dengan cahaya seperti jemuran otomatis, tracking arah sumber cahaya matahari, lampu otomatis (untuk rumah, aquarium, dll), atau sebagai pengatur intensitas cahaya lampu untuk tananaman di dalam ruangan, dan banyak lagi lainnya. Di pasaran ada LDR yang berukuran 4 mm dan 11 mm. Pada Gambar 6.1 adalah LDR dengan ukuran 11 mm.



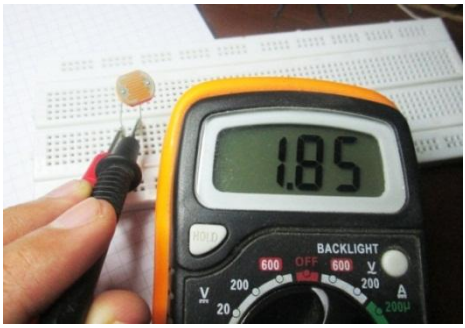
Gambar 6.1 LDR 11mm

6.1 Cara Kerja LDR

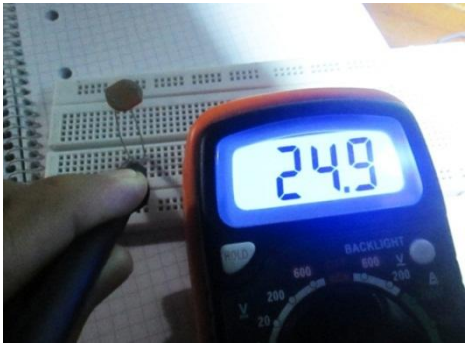
LDR disebut juga sebagai *photoresistor* sebab alat ini akan memiliki resistansi yang akan berubah seiring dengan perubahan intensitas cahaya yang mengenainya. Dalam kondisi gelap, resistansi LDR bisa mencapai 10 M ohm, tapi dalam kondisi terang, resistansi LDR turun hingga 1 K ohm bahkan bisa kecil lagi (Gambar 6.2 dan 6.3). Sifat inilah yang membuat LDR bisa dimanfaatkan sebagai sensor cahaya.

LDR terbuat dari sebuah cakram semikonduktor seperti kadmium sulfida dengan dua buah elektroda pada permukaannya. Pada saat intensitas cahaya yang mengenai LDR sedikit, bahan dari cakram LDR

tersebut menghasilkan elektron bebas dengan jumlah yang relatif kecil. Sehingga hanya ada sedikit elektron untuk mengangkut muatan elektrik. Artinya saat intensitas cahaya yang mengenai LDR sedikit maka LDR akan memiliki resistansi yang besar.



Gambar 6.2 Resistansi LDR diterangi lampu



Gambar 6.3 Resistansi LDR ketika lampu dihalangi kertas

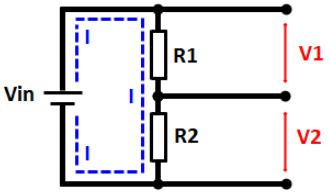
Sedangkan pada saat kondisi terang, maka intensitas yang mengenai LDR banyak. Maka energi cahaya yang diserap akan membuat elektron bergerak cepat sehingga lepas dari atom bahan semikonduktor tersebut. Dengan banyaknya elektron bebas, maka muatan listrik lebih mudah untuk dialirkan. Artinya saat intensitas cahaya yang mengenai LDR banyak maka LDR akan memiliki resistansi yang kecil dan menjadi konduktor yang baik.

Gambar di atas adalah resistansi pada LDR dalam kondisi terang dan kondisi gelap. Dalam kondisi terang, resistansi masih kisaran 1K

ohm, dan ketika cahaya sedikit terhalangi sehingga agak gelap, maka resistansi meningkat hingga puluhan kilo ohm. Karakteristik inilah yang bisa kita manfaatkan untuk mengaktifkan relay dan menghidupkan lampu.

6.2 Rangkaian Dasar LDR

Ketika ingin menjadikan LDR sebagai sensor, maka kita bisa mengacu pada rangkaian resistor sebagai pembagi tegangan (lihat Gambar 6.4). Dengan menggabungkan antara LDR dengan resistor (atau potensiometer), maka kita bisa mendapatkan variasi tegangan (pada V1 atau V2) yang nantinya menjadi inputan pada pin analog Arduino.



Gambar 6.4 Rangkaian pembagi tegangan

Tegangan pada V1 atau V2 dapat dihitung berdasarkan hukum ohm dan aturannya pada rangkaian seri. Pada rangkaian tersebut, arus pada semua titik dalam rangkaian tersebut nilainya sama sehingga kita bisa menghitung V1 atau V2 tanpa mengetahui arus yang mengalir. Lalu bagaimana cara menghitung V1 dan V2?

Pada rangkaian, ada 3 titik yang memiliki tegangan berbeda. Tegangan Vin, tegangan pada R1, dan tegangan pada R2. Berdasarkan hukum ohm, Vin, V1, dan V2 bisa dihitung dengan cara:

$$Vin = I\,.(R1 + R2)$$

$$I = \frac{Vin}{(R1+R2)} \qquad \text{(pers 1)}$$

$$V1 = I\,.\,R1 \qquad \text{(pers 2)}$$

$$V2 = I \,.\, R2 \qquad \text{(pers 3)}$$

Jika ingin menghitung V1, maka kita tinggal menyubstitusikan antara pers 1 dan pers 2.

$$V1 = I \,.R1$$

$$V1 = \frac{Vin}{(R1 + R2)} \,.\, R1$$

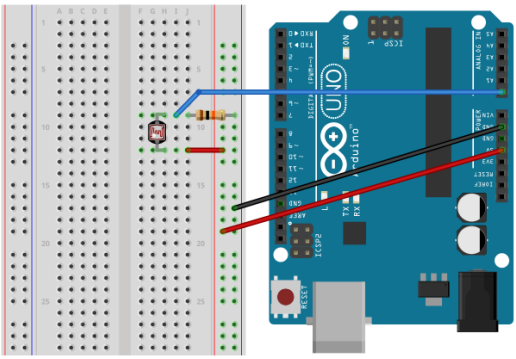
Atau lebih umum dikenal dengan rumus :

$$V1 = \frac{R1}{(R1 + R2)} \,.Vin$$

Lalu jika ingin menghitung V2, maka rumusnya adalah:

$$V2 = \frac{R2}{(R1 + R2)} \,.Vin$$

Berdasarkan cara kerja rangkaian tersebut, maka rangkaian untuk sensor cahaya adalah sebagai berikut:



Rangkaian 6.1 Sensor cahaya dan Arduino

Berdasarkan Rangkaian 6.1, yang perlu Anda siapkan adalah resistor 10 K ohm, LDR, dan beberapa kabel jumper. Agar bisa coba-coba, silakan resistor 10 K ohm bisa Anda ganti dengan potensiometer 50 K atau 100 K, sehingga Anda lebih mudah ketika mencoba dengan resistansi yang berbeda. Potensiometer juga bisa digunakan untuk kalibrasi input pada Arduino.

1. Salah satu kaki LDR disambungkan ke VCC pada Arduino
2. Salah satu kaki Resistor disambungkan ke GND pada arduino
3. Sambungkan sisa kaki LDR dan sisa kaki resistor, kemudian sambungan tersebut dihubungkan ke pin A0 pada board Arduino

6.3 Program Sensor Cahaya

Sketch 6.1 Program sensor cahaya

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // pin A0 ke LDR
6  const int pinLDR = A0;
7
8  void setup() {
9    Serial.begin(9600);
10   pinMode(pinLDR, INPUT);
11 }
12
13 int dataLDR = 0;
14 void loop() {
15   dataLDR = analogRead(pinLDR);
16   Serial.print("dataLDR : ");
17   Serial.print(dataLDR);
18   Serial.print(" Kondisi : ");
19   if(dataLDR < 150){
20     Serial.println("GELAP");
21   }else if(dataLDR < 300){
22     Serial.println("REDUP");
23   }else if(dataLDR < 450){
24     Serial.println("TERANG");
25   }else{
26     Serial.println("SILAU");
27   }
28
29   delay(1000);
30 }
```

Program pada Sketch 6.1 akan membaca nilai tegangan pada sensor dan mengirimkannya ke komputer melalui komunikasi serial. Dengan Arduino, kita bisa membuat berbagai logika untuk sensor cahaya sehingga aplikasi dari LDR ini bisa diperluas dan dibuat lebih kompleks diintegrasikan dengan berbagai sistem.

Apabila dalam di dalam diri seseorang masih ada rasa malu dan takut untuk berbuat suatu kebaikan, maka jaminan bagi orang tersebut adalah tidak akan bertemunya ia dengan kemajuan selangkah pun.

(Ir Soekarno)

Bagian 7

Sensor Ultrasonik

Gelombang ultrasonik merupakan gelombang yang umum digunakan untuk radar untuk mendeteksi keberadaan suatu benda dengan memperkirakan jarak antara sensor dan benda tersebut. Dalam ebook ini, kita akan mempelajarinya dengan salah satu sensor ultrasonik HC-SR04 sebab sensor ini juga relatif terjangkau untuk pembelajaran. Bentuk fisik dari sensor ini tampak seperti gambar 7.1.

Gambar 7.1 Sensor ultrasonik HC-SR04

7.1 Sekilas tentang Sensor Ultrasonik

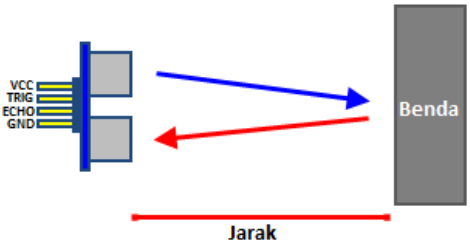
Sensor ultrasonik adalah sebuah sensor yang berfungsi untuk mengubah besaran fisis (bunyi) menjadi besaran listrik dan sebaliknya. Cara kerja sensor ini didasarkan pada prinsip dari pantulan suatu gelombang suara sehingga dapat dipakai untuk menafsirkan eksistensi (jarak) suatu benda dengan frekuensi tertentu. Disebut sebagai sensor ultrasonik karena sensor ini menggunakan gelombang ultrasonik (bunyi ultrasonik).

Gelombang ultrasonik adalah gelombang bunyi yang mempunyai frekuensi sangat tinggi yaitu 20.000 Hz. Bunyi ultrasonik tidak dapat di dengar oleh telinga manusia. Bunyi ultrasonik dapat didengar oleh anjing, kucing, kelelawar, dan lumba-lumba. Bunyi ultrasonik bisa merambat melalui zat padat, cair dan gas. Reflektivitas bunyi ultrasonik di permukaan zat padat hampir sama dengan reflektivitas bunyi ultrasonik di permukaan zat cair. Akan tetapi, gelombang bunyi ultrasonik akan diserap oleh tekstil dan busa.

Berikut ini adalah beberapa aplikasi dari gelombang ultrasonik:

- **Dalam bidang kesehatan**, gelombang ultrasonik bisa digunakan untuk melihat organ-organ dalam tubuh manusia seperti untuk mendeteksi tumor, liver, otak dan menghancurkan batu ginjal. Gelombang ultrasonik juga dimanfaatkan pada alat USG (ultrasonografi) yang biasa digunakan oleh dokter kandungan.
- **Dalam bidang industri**, gelombang ultrasonik digunakan untuk mendeteksi keretakan pada logam, meratakan campuran besi dan timah, meratakan campuran susu agar homogen, mensterilkan makanan yang diawetkan dalam kaleng, dan membersihkan benda benda yang sangat halus. Gelombang ultrasonik juga bisa digunakan untuk mendeteksi keberadaan mineral maupun minyak bumi yang tersimpan di dalam perut bumi.
- **Dalam bidang pertahanan**, gelombang ultrasonik digunakan sebagai radar atau navigasi, di darat maupun di dalam air. Gelombang ultrasonik digunakan oleh kapal pemburu untuk mengetahui keberadaan kapal selam, dipasang pada kapal selam untuk mengetahui keberadaan kapal yang berada di atas permukaan air, mengukur kedalaman palung laut, mendeteksi ranjau, dan menentukan puosisi sekelompok ikan.

7.2 Cara Kerja Sensor Ultrasonik



Gambar 7.2 Cara kerjas sensor ultrasonik

Pada sensor ultrasonik, gelombang ultrasonik dibangkitkan melalui sebuah alat yang disebut dengan piezoelektrik dengan frekuensi tertentu. Piezoelektrik ini akan menghasilkan gelombang ultrasonik (umumnya berfrekuensi 40kHz) ketika sebuah osilator

diterapkan pada benda tersebut. Secara umum, alat ini akan menembakkan gelombang ultrasonik menuju suatu area atau suatu target. Setelah gelombang menyentuh permukaan target, maka target akan memantulkan kembali gelombang tersebut. Gelombang pantulan dari target akan ditangkap oleh sensor, kemudian sensor menghitung selisih antara waktu pengiriman gelombang dan waktu gelombang pantul diterima (Gambar 7.2).

Karena kecepatan bunyi adalah 340 m/s, maka rumus untuk mencari jarak berdasarkan ultrasonik adalah :

$$S = \frac{340 \, . \, t}{2}$$

dimana S merupakan jarak antara sensor ultrasonik dengan benda (bidang pantul), dan t adalah selisih antara waktu pemancaran gelombang oleh transmitter dan waktu ketika gelombang pantul diterima receiver.

HC-SR04 merupakan sensor ultrasonik siap pakai, satu alat yang berfungsi sebagai pengirim, penerima, dan pengontrol gelombang ultrasonik. Alat ini bisa digunakan untuk mengukur jarak benda dari 2cm - 4m dengan akurasi 3mm. Dengan demikian, untuk menghitung jarak yang hanya maksimal 4 m maka rumus di atas harus dimodifikasi atau disesuaikan satuannya. Mikrokontroller bisa bekerja pada order mikrosekon (1s = 1.000.000 µs) dan satuan jarak bisa kita ubah ke satuan cm (1m = 100 cm). Oleh sebab itu, rumus di atas bisa diupdate menjadi:

$$S = \frac{340 \left(\frac{100}{1000000}\right) . \, t}{2}$$

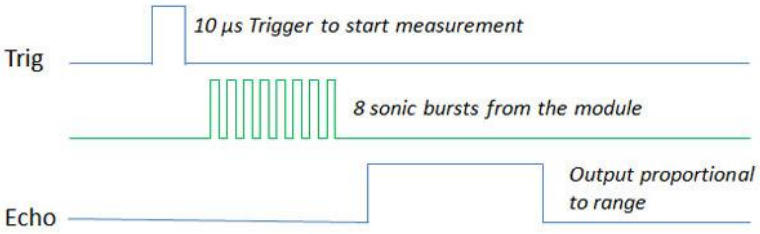
$$S = \frac{0.034 \, . \, t}{2}$$

Alat ini memiliki 4 pin, pin Vcc, Gnd, Trigger, dan Echo. Pin Vcc untuk listrik positif dan Gnd untuk ground-nya. Pin Trigger untuk trigger keluarnya sinyal dari sensor dan pin Echo untuk menangkap sinyal pantul dari benda.

Cara menggunakan alat ini yaitu:

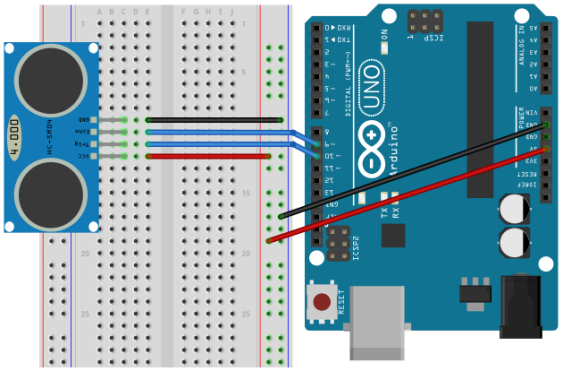
- Ketika kita memberikan tegangan positif pada pin Trigger selama 10µS, maka sensor akan mengirimkan 8 step sinyal ultrasonik dengan frekuensi 40kHz
- Selanjutnya, sinyal akan diterima pada pin Echo
- Untuk mengukur jarak benda yang memantulkan sinyal tersebut, selisih waktu ketika mengirim dan menerima sinyal digunakan untuk menentukan jarak benda tersebut
- Rumus untuk menghitung jaraknya adalah S = (0.034 *t) /2 cm.

Berikut adalah visualisasi dari sinyal yang dikirimkan oleh sensor HC-SR04



Gambar 7.3 Timing HC-SR04

7.3 Rangkaian Sensor Jarak dengan HC-SR04



Rangkaian 7.1 Sensor jarak dengan HC-SR04

Buatlah rangkaian seperti Rangkaian 7.1

1. VCC pada HC-SR04 dihubungkan ke +5V pada board Arduino
2. GND disambungkan ke GND
3. Pin Trig (*Trigger*) disambungkan ke pin 10 pada *board* Arduino
4. Pin *Echo* disambungkan ke pin 9 pada board Arduino

7.4 Program Sensor Jarak

Sketch 7.1 Program sensor jarak

```
1  // Free Ebook Arduino
2  // www.elangsakti.com
3  // coder elangsakti
4
5  // pin 9 trigger
6  // pin 10 echo
7  const int pTrig = 9;
8  const int pEcho = 10;
9
10 void setup() {
11   Serial.begin(9600);
12   pinMode(pTrig, OUTPUT);
13   pinMode(pEcho, INPUT);
14 }
15
16 long durasi = 0;
17 void loop() {
18   // trigger selama 10us
19   digitalWrite(pTrig, HIGH);
20   delayMicroseconds(10);
21   digitalWrite(pTrig, LOW);
22
23   durasi = pulseIn(pEcho, HIGH);
24   Serial.print("Durasi: ");
25   Serial.print(durasi);
26
27   Serial.print(", Jarak: ");
28   Serial.println((durasi *0.034)/2);
29   delay(1000);
30 }
```

Program pada Sketch 7.1 akan mengaktifkan pin *Trigger* selama 10µs, kemudian akan menunggu hingga pin *Echo* bernilai HIGH. Perhatikan pada baris ke-23.

```
23   durasi = pulseIn(pEcho, HIGH);
```

Fungsi *pulseIn()* akan memerintahkan sistem untuk menunggu hingga pin *Echo* bernilai HIGH. Lama proses menunggu akan dianggap sebagai durasi pengiriman + penerimaan sinyal *echo* yang dipantulkan oleh benda.

Sedikit penjelasan tentang fungsi *pulseIn()*, fungsi ini standarnya memiliki 3 parameter, 2 parameter Pin dan Value dan 1 parameter *Timeout* sebagai parameter tambahan:

```
pulseIn(Pin, Value);
pulseIn(Pin, Value, Timeout);
```

Pada parameter Value, kita bisa memasukkan HIGH atau LOW, jadi Arduino akan menunggu hingga kondisi tersebut dipenuhi. Sedangkan *Timeout* digunakan ketika dalam waktu tertentu kondisi belum juga terpenuhi. Begitulah dasar dari pembuatan sensor jarak dengan sensor ultrasonik HC-SR04.

Aku tinggalkan Kekayaan alam Indonesia, biar semua negara besar dunia iri dengan Indonesia, dan aku tinggalkan hingga bangsa Indonesia sendiri yang mengolahnya.

(Ir. Soekarno)

Penutup

Saat ini dunia mulai bergerak ke arah *Internet of Things* (IoT) dan *Web of Things* (WoT). Artinya, semua peralatan elektronik di sekitar kita akan dihubungkan ke internet termasuk website. Konsekuensinya, peralatan tersebut akan bisa diakses via web dan perangkat mobile. Misal, kita bisa memantau kondisi rumah dengan smartphone, memantau kondisi tanaman dan perkebunan dengan smartphone, adanya kamera dan *drone* sebagai pemantau lingkungan dan lain sebagainya.

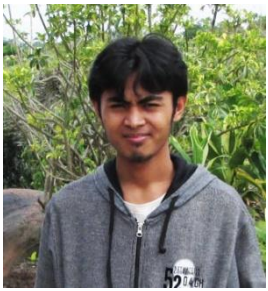
Sehingga rumah-rumah akan terintegrasi dengan peralatan dan otomatisasi yang mungkin dikenal dengan istilah *smart house*. Dunia perkebunan akan bergerak ke era *smart gardening*. Pertanian akan bergerak ke era *smart farming*. Begitu juga dalam dunia medis dan transportasi. Semua akan terintegrasi ke dalam *smart grid* dan *smart city*. Oleh sebab itu, mari kita asah kemampuan kita lebih dalam lagi.

Penulis tidak akan bosan-bosan untuk mengingatkan bahwa, kemajuan suatu negeri salah satunya tergantung teknologi yang dikuasainya. Dunia komputer dan mikrokontroller merupakan cikal-bakal teknologi untuk otomatisasi berbagai hal, baik untuk skala rumahan maupun skala industri.

Yang lebih penting lagi, untuk bisa bersaing dengan produsen luar negeri, maka kita membutuhkan berbagai teknologi yang bisa membantu UKM-UKM dan produsen lokal dalam pembuatan produk. Andai pembuatan tahu dan tempe bisa diotomatisasi, mungkin produsen tempe bisa lebih mudah dan lebih banyak memproduksi tempe. Begitu juga dengan produsen-produsen lainnya. Mereka membutuhkan kita, mereka membutuhkan sentuhan teknologi untuk lebih berkembang lagi. ☺

MARI BERKARYA!

Tentang Penulis



Hari Santoso, lahir di Situbondo 1988. Pendidikan formal dimulai dari MI Islamiyah Curah Kalak, dilanjutkan ke SLTP 1 Jangkar dan SMAN 1 Asembagus. Perguruan tinggi ditempuh di S1 Teknik Informatika UIN Maulana Malik Ibrahim Malang dan saat ini (2016) sedang menempuh Pascasarjana Teknik Elektro Universitas Brawijaya, Malang.

Penulis pernah menjadi dosen LB untuk mata kuliah Kecerdasan Buatan (*Artificial Intelligence*) dan Sistem Operasi di Universitas Muhammadiyah Malang (2011-2013) serta dosen LB untuk mata kuliah Rangkaian Logika, Sistem Pakar, dan Jaringan Wireless di Akademi Manajemen Informatika dan Komputer (AMIK) Ibrahimy di PP. Salafiyah Syafi'iyah Sukorejo, Situbondo (2011 - sekarang).

Minat utama penulis adalah di bidang Kecerdasan Buatan, Keamanan Komputer dan Jaringan, Programming, Elektronika, serta perpaduannya. Ebook ini adalah salah satu bentuk nyata dan upaya penulis untuk ikut andil dalam penyebaran ilmu sains dan teknologi bagi generasi muda Indonesia.