

Buku Panduan

Arduino Starterkit Dasar

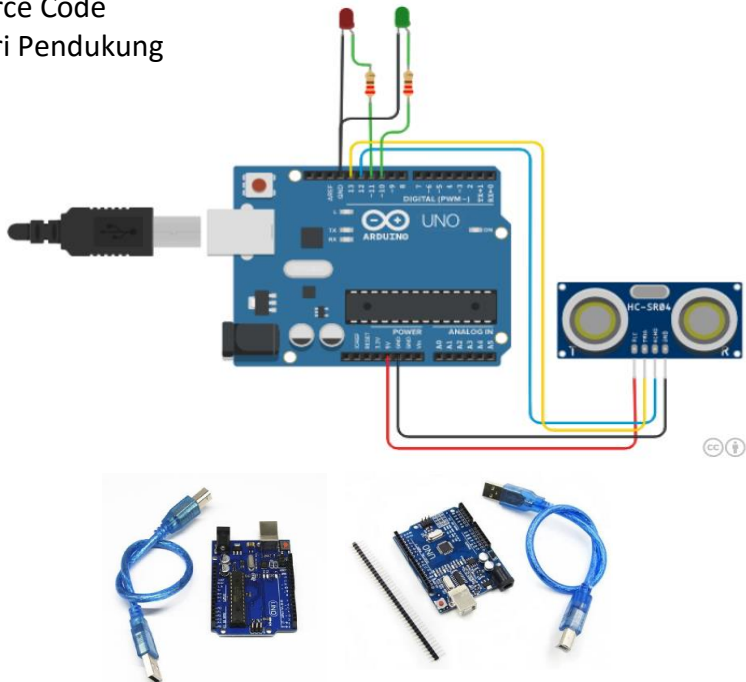
Rangkaian

Sensor

Simulasi

Source Code

Teori Pendukung



<https://digpartmalang.business.site/>

Tentang Kami

<https://digpartmalang.business.site/>

Anda ingin membangun instrument ilmiah dengan biaya rendah? Membuktikan prinsip kimia dan fisika? Atau hendak memulai pemrograman dan robotika? Membuat desain rumah lengkap dengan IOT? Atau bahkan akan bereksperimen dengan alat musik? Arduino jawabnya! He he...sesederhana itukah? Kami menjual produk dengan mengajak anda bereksperimen sesederhana mungkin sehingga bisa diaplikasikan dalam kehidupan sehari-hari.

Ebook ini terinspirasi dari buku Arduino Starters Kit Manual karangan Mike McRoberts. 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 Tinkercad. Arduino yang digunakan dalam buku ini adalah Arduino Uno.

###Mari Memajukan Bangsa dengan Teknologi###

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 Beliau, kepada para Nabi, keluarga, dan para penerusnya.

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

Salam,

Malang, 06 Desember 2020

Imam Sholahudin Mahmudi

Daftar Isi

| | |
|---|----|
| Tentang Kami..... | 1 |
| Persembahan..... | 2 |
| Daftar Isi | 3 |
| Pendahuluan..... | 5 |
| Cara menggunakannya | 6 |
| Apa yang Anda perlukan..... | 6 |
| Isi Starter Kit | 7 |
| Apa sebenarnya Arduino itu? | 8 |
| Instalasi Arduino..... | 9 |
| Arduino IDE | 15 |
| Proyek 1 - LED Berkedip | 17 |
| Ikhtisar Kode | 20 |
| Ikhtisar Perangkat Keras | 24 |
| Proyek 2 - Signal Kode SOS Morse..... | 30 |
| Ikhtisar Kode | 32 |
| Proyek 3 - Lampu Lalu Lintas | 36 |
| Proyek 4 - Lampu Lalu Lintas Interaktif | 39 |
| Ikhtisar Kode | 43 |
| Proyek 5 - Animasi LED efek kejar-kejaran | 50 |
| Ikhtisar Kode | 51 |
| Proyek 6 - Efek Pengejaran LED Interaktif | 55 |
| Ikhtisar Kode | 57 |
| Ikhtisar Perangkat Keras | 58 |
| Proyek 7 - Lampu Berdenyut | 60 |
| Ikhtisar Kode | 61 |
| Proyek 8 - LED RGB..... | 64 |
| Ikhtisar Kode | 66 |
| Proyek 9 - LED Efek Kebakaran..... | 72 |
| Ikhtisar Kode | 74 |
| Proyek 10 - Lampu RGB Terkendali Serial | 76 |
| Ikhtisar Kode | 79 |
| Proyek 11 - Piezo Sounder Melody Player | 93 |

| | |
|--|-----|
| Ikhtisar Kode | 95 |
| Ikhtisar Perangkat Keras | 100 |
| Proyek 12 - Sensor Jarak..... | 102 |
| Ikhtisar Kode | 104 |
| Ikhtisar Perangkat Keras | 104 |
| Proyek 13 - Sensor Cahaya | 108 |
| Ikhtisar Kode | 109 |
| Ikhtisar Perangkat Keras | 110 |
| Proyek 14 - Shift Register 8-Bit Binary Counter..... | 113 |
| Sistem Bilangan Biner..... | 116 |
| Ikhtisar Perangkat Keras | 118 |
| Proyek 15 – LCD Display Character 1602..... | 123 |
| Ikhtisar Perangkat Keras..... | 126 |
| Penutup | 128 |

Pendahuluan

Buku ini akan memandu Anda, selangkah demi selangkah untuk mempelajari tentang perangkat keras Arduino, perangkat lunak dan teori elektronik umum. Melalui penggunaan proyek elektronik kami akan membawa Anda dari tingkat pemula hingga memiliki seperangkat keterampilan menengah dalam menggunakan Arduino yang lengkap.

Buku ini ditulis dengan anggapan bahwa Anda tidak memiliki pengetahuan sebelumnya tentang elektronik, perangkat keras Arduino, lingkungan perangkat lunak atau pemrograman komputer.

Ada banyak sumber daya lain yang tersedia secara gratis yang memungkinkan Anda mempelajari lebih banyak tentang subjek ini jika Anda ingin melangkah lebih jauh. Cara terbaik untuk mempelajari Arduino, setelah menggunakan kit ini tentunya, adalah dengan bergabung dengan Forum Arduino di situs Arduino dan untuk melihat contoh kode dan perangkat keras di bagian 'Playground' di situs Arduino atau di <https://playground.arduino.cc/>

Kami harap Anda menikmati penggunaan kit ini dan mendapatkan kepuasan dari membuat proyek dan melihat kreasi Anda menjadi nyata.

Cara menggunakannya

Buku ini dimulai dengan pengantar Arduino, cara mengatur perangkat keras, menginstal perangkat lunak, dll. Kita kemudian menjelaskan IDE Arduino dan cara menggunakannya sebelum kita menyelami langsung ke beberapa proyek yang berkembang. Setiap proyek akan dimulai dengan deskripsi tentang cara mengatur perangkat keras dan kode apa yang diperlukan. Kami kemudian akan menjelaskan secara terpisah kode dan perangkat kerasnya dan menjelaskan secara rinci cara kerjanya.

Apa yang Anda perlukan

Pertama, Anda memerlukan akses ke internet untuk dapat mengunduh Arduino IDE (Integrated Development Environment) dan Library Arduino yang mungkin diperlukan agar proyek Anda berfungsi.







Anda akan membutuhkan meja yang cukup terang atau permukaan lainnya untuk meletakkan komponen Anda dan ini harus di sebelah PC atau laptop Anda untuk memungkinkan Anda mengunggah kode ke Arduino. Ingatlah bahwa Anda bekerja dengan listrik dan oleh karena itu meja atau permukaan logam harus terlebih dahulu ditutup dengan bahan non-konduktif (misalnya taplak meja, kertas, dll.)

Beberapa keuntungan juga, meskipun tidak penting, mungkin berupa tang potong, tang cucut, dan alat pengupas kabel. Buku catatan, pulpen dan penggaris juga akan berguna.

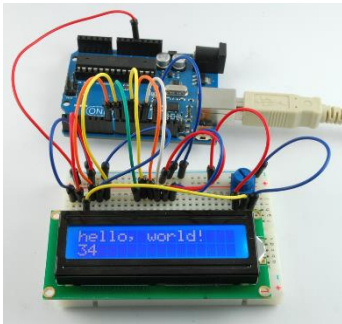
Terakhir, hal terpenting yang Anda perlukan adalah antusiasme dan kemauan untuk belajar. Arduino dirancang sebagai cara sederhana dan murah untuk terlibat dalam elektronik mikrokontroler dan tidak ada yang terlalu sulit untuk dipelajari jika setidaknya Anda bersedia untuk '**mencobanya**'.

Isi Starter Kit

Harap dicatat bahwa isi kit Anda mungkin terlihat sedikit berbeda dengan yang tercantum di sini

| | | | | |
|---|---|--|---|---|
|  <p>Arduino UNO R3 CH340</p> |  <p>Box 1000 ml Tempat Box</p> |  <p>Breadboard 400 Point</p> |  <p>Kabel Data Arduino Uno</p> |  <p>Passive Buzzer 2KHz</p> |
|  <p>Electrical Resistance</p> <p>Resistor 220Ω 10pcs</p> |  <p>Electrical Resistance</p> <p>Resistor 470Ω 10pcs</p> |  <p>Electrical Resistance</p> <p>Resistor 1KΩ 10pcs</p> |  <p>Electrical Resistance</p> <p>Resistor 10KΩ 10pcs</p> |  <p>Electrical Resistance</p> <p>Resistor 100KΩ 10pcs</p> |
|  <p>Led Merah 3mm 10pcs</p> |  <p>Led Kuning 3mm 10pcs</p> |  <p>Led Hijau 3mm 10pcs</p> |  <p>LED RGB 5mm Com Cathode</p> |  <p>Micro Switch 5 PCS</p> |
|  <p>Jumper Kabel Isi 10 Male- Female</p> |  <p>Jumper Kabel Isi 10 Male- Male</p> |  <p>Jumper Kabel Isi 10 Female- Female</p> |  <p>LDR 5528 5 Mm 2 pcs</p> |  <p>HCSR04 Sensor Ultrasonic</p> |
|  <p>IC Shift Register 74HC595 DIP</p> |  <p>Trimpot 4k7</p> |  <p>Trimpot 10K</p> |  <p>LCD 2x16 16x2 1602 Display</p> |  <p>Buku Panduan Arduino Starterkit Dasar</p> <p>e-book Arduino Starterkit Dasar</p> |

Apa sebenarnya Arduino itu?



Sekarang Anda bangga menjadi pemilik Arduino, atau Arduino Clone, mungkin membantu jika Anda tahu apa itu dan apa yang dapat Anda lakukan dengannya. Arduino adalah komputer kecil yang dapat Anda program untuk memproses masukan dan keluaran.

Arduino dapat digunakan untuk mengembangkan objek interaktif yang berdiri sendiri atau dapat dihubungkan ke komputer untuk mengambil atau mengirim data ke Arduino dan kemudian bertindak atas data tersebut (misalnya Mengirim data sensor ke internet). Arduino dapat dihubungkan ke LED, Tampilan Dot Matrix, monitor VGA, tombol, sakelar, motor, sensor suhu, sensor tekanan, sensor jarak, webcam, printer, penerima GPS, modul ethernet, Papan Arduino terbuat dari Mikroprosesor AVR Atmel, kristal atau osilator dan regulator linier 5 volt. Bergantung pada jenis Arduino yang Anda miliki, Anda mungkin juga memiliki konektor USB untuk mengaktifkannya agar dapat dihubungkan ke PC atau Mac untuk mengunggah atau mengambil data. Arduino mempunyai pin I / O (Input / Output) untuk memungkinkan Anda menghubungkan pin tersebut ke sirkuit lain atau ke sensor, dll.

Untuk memprogram Arduino (membuatnya melakukan apa yang Anda inginkan) Anda juga menggunakan Arduino IDE (Integrated Development Environment), yang merupakan bagian dari perangkat lunak gratis, yang memungkinkan Anda memprogram dalam bahasa yang dimengerti oleh Arduino. Dalam kasus Arduino, bahasanya adalah C. IDE memungkinkan Anda untuk menulis program komputer, yang merupakan sekumpulan instruksi langkah demi langkah yang kemudian Anda unggah ke Arduino. Kemudian Arduino Anda akan melaksanakan instruksi tersebut dan berinteraksi dengan dunia luar. Di dunia Arduino, program dikenal sebagai 'Sketch'.

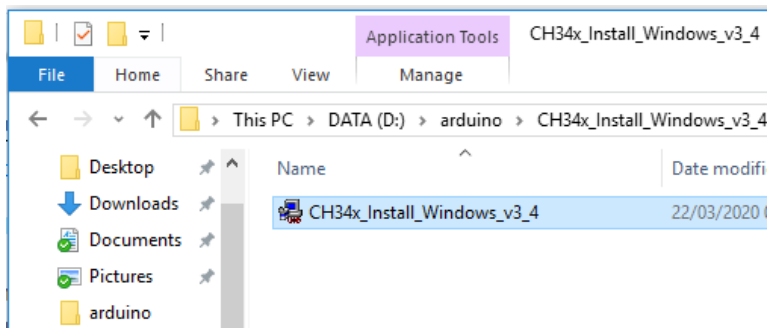
Instalasi Arduino

Karena Arduino Open Source baik hardware maupun software, maka banyak yang produksi Arduino. Dan beberapa ada yang menggunakan **Downloader tipe CH340** yang mana jika tidak install Driver nya por USB arduinonya tidak akan terbaca.

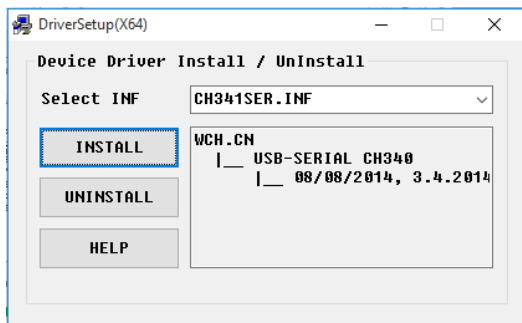
Kita harus install terlebih dahulu **Driver USB CH340** nya. Langkah pertama, anda bisa Download dulu file Driver tersebut melalui Link dibawah ini:

<https://sparks.gogo.co.nz/ch340.html>

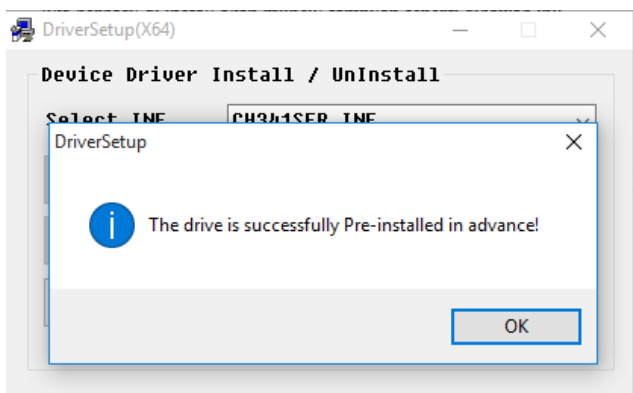
Jika sudah di Download filenya anda tinggal ekstrak file .zip nya dan buka, terus klik file **CH34x_Install_Windows_v3_4**.



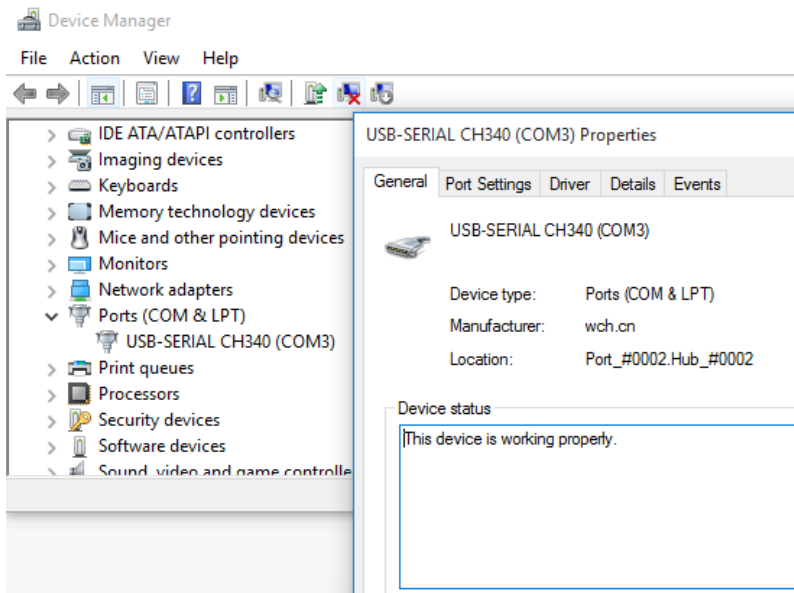
Jika sudah muncul tampilan seperti dibawah ini, anda tinggal klik **INSTALL**.



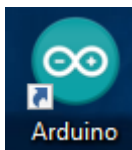
Jika berhasil di install akan muncul tampilan seperti dibawah ini:



Sekarang driver USB anda sudah terinstall. Terkadang tidak ada identitas khusus pada port Arduino IDE, untuk CH340G. Tapi anda bisa cek yang mana port yang CH340 dengan membuka DEVICE MANAGER. lalu pilih bagian port.

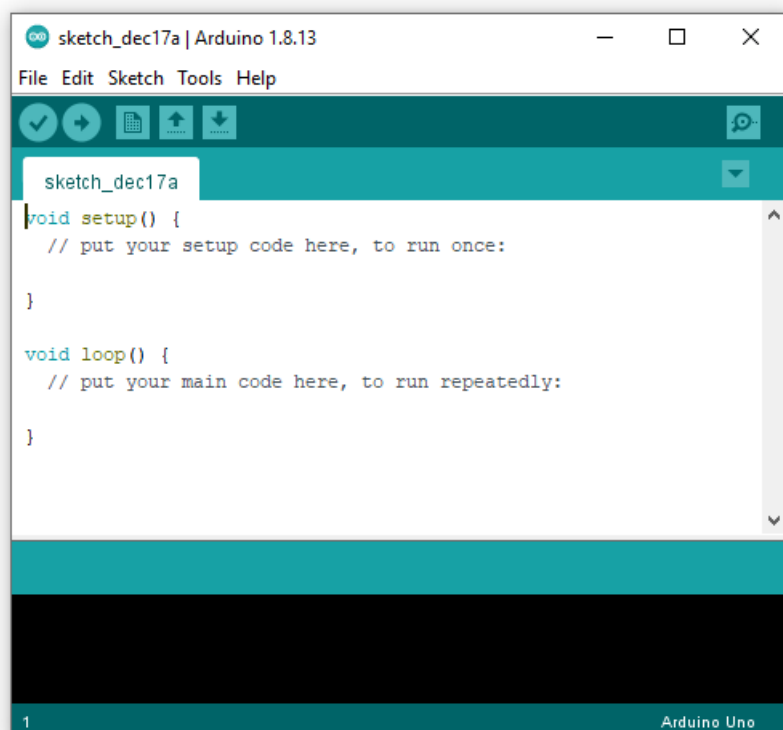


Dari gambar di atas menunjukkan bahwa Arduino dengan driver CH340 menempati COM3



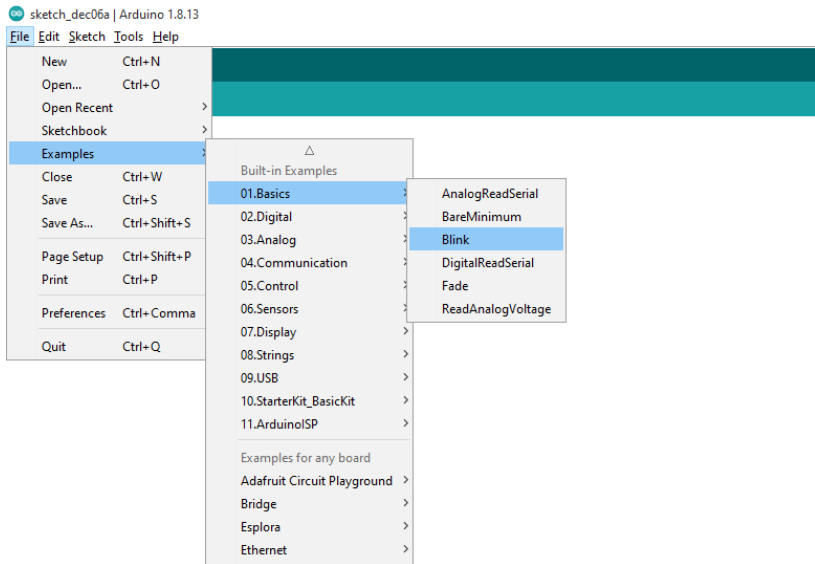
Cari ikon Arduino IDE di desktop, yang terlihat seperti ini

Klik dua kali ICON untuk membuka IDE. Anda kemudian akan disajikan dengan layar biru dan putih dengan sketsa default dimuat di dalamnya.

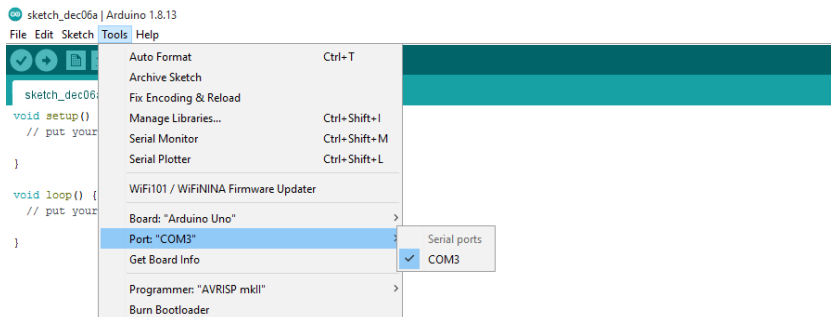


Ini adalah Arduino IDE (Integrated Development Environment) dan di sinilah Anda akan menulis Sketsa (program) untuk diunggah ke board Arduino Anda.

Kita akan melihat IDE sedikit lebih detail di bab berikutnya. Untuk saat ini, cukup klik **File** di file menu dan gulir ke bawah ke **Sketchbook**. Kemudian gulir ke bawah ke **Examples** dan klik. Anda akan disajikan daftar Contoh *sketch* yang dapat Anda gunakan untuk mencoba Arduino Anda. Sekarang klik **Basics** dan di dalamnya Anda akan menemukan contoh Sketch yang disebut **Blink**. Klik ini.

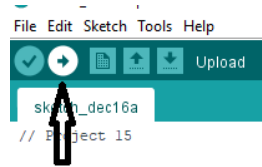


Sketch Blink akan dimuat ke dalam IDE dan port, jadi sekarang klik Tools lagi, gulir ke bawah ke Serial Port dan daftar port serial yang tersedia pada sistem Anda akan ditampilkan. Anda harus memilih salah satu yang mengacu pada kabel USB Anda, yang biasanya terdaftar sebagai sesuatu seperti /dev/tty.usbserial-xxxx di Mac atau COM3 di Windows jadi klik itu. Jika tidak yakin, coba masing-masing sampai Anda menemukan yang berfungsi

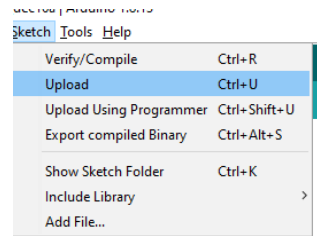


Sekarang Anda telah memilih board dan port USB yang benar, Anda siap untuk mengunggah Sketch Blink ke board.

Anda dapat mengklik tombol Unggah, yang merupakan tombol ke-2 dari kiri di atas dengan panah mengarah ke kanan (arahkan penunjuk mouse Anda ke atas tombol untuk melihat apa itu) atau dengan mengklik **upload** di **menu Sketch** (lihat gambar di samping)



Dengan asumsi semuanya telah diatur dengan benar, Anda sekarang akan melihat RX dan TX LED (dan juga LED 13) pada lampu led Arduino berkedip dengan sangat cepat saat data diunggah ke board.



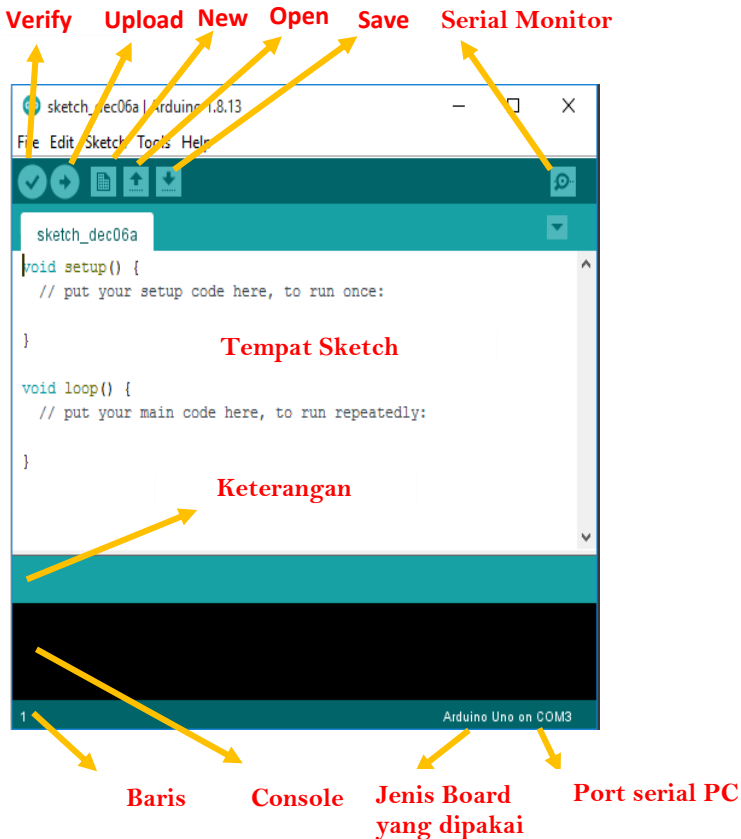
Setelah data berhasil diunggah ke board, Anda akan mendapatkan pesan Selesai Mengunggah di IDE dan LED RX / TX akan berhenti berkedip.

Arduino sekarang akan mengatur ulang dirinya sendiri dan segera mulai menjalankan Sketch yang baru saja Anda unggah.

Sketch Blink adalah sketch yang sangat sederhana untuk membuat LED 13 berkedip, yang merupakan LED kecil yang disolder ke board

Arduino IDE

Pada saat buku ini ditulis, Arduino IDE tersedia versi 1.8.13. 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. Sketch merupakan *source code* yang berisi logika dan algoritma yang akan diupload ke dalam IC mikrokontroler (Arduino).







Interface Arduino IDE tampak seperti gambar 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 mikrokontroler.
- ✓ **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 dipakai oleh *board* Arduino.

Proyek 1 - LED Berkedip

Dalam proyek ini kita akan mengulangi apa yang kita lakukan dalam menyiapkan dan menguji Arduino, yaitu mengedipkan LED. Namun, kali ini kita akan menggunakan salah satu LED dalam kit dan Anda juga akan belajar tentang beberapa rangkaian elektronik dan pemrograman bahasa C.

Apa yang Anda butuhkan

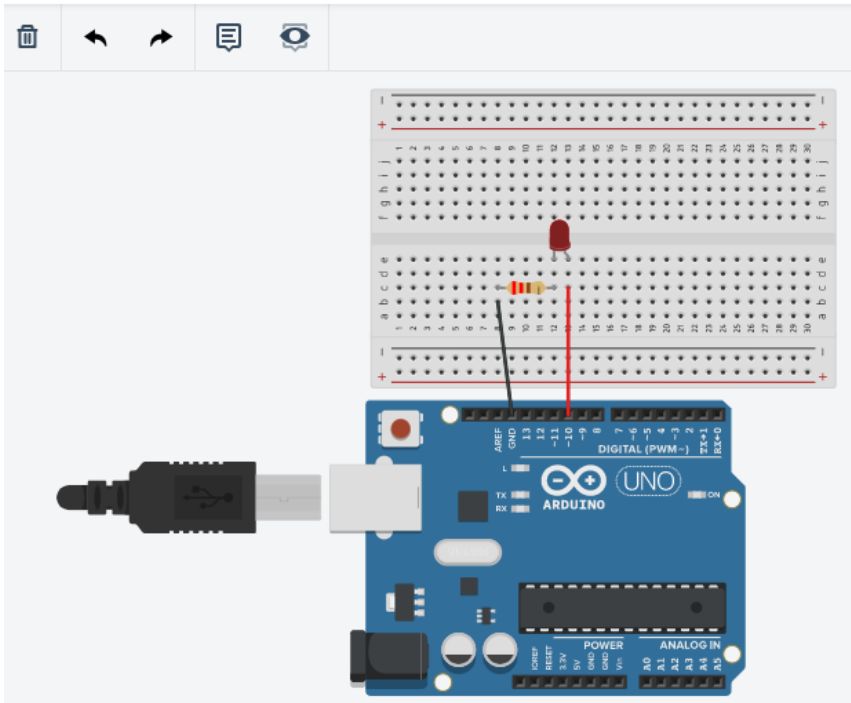
| | |
|---------------------------------|---|
| Solderless Breadboard 400 Point |  |
| Led Merah - Nyala Merah 3mm |  |
| Resistor 220Ω |  |
| Jumper Kabel Dupont 10cm |  |

Menghubungkan Rangkaian

Sekarang, pertama-tama pastikan Arduino Anda dimatikan. Ini adalah langkah keamanan kerja. Anda dapat melakukan ini dengan

mencabut kabel USB dan/ atau dengan mencabut jack catu daya pada board Arduino. Kemudian hubungkan semuanya seperti ini:

Proyek 1 - LED Berkedip



Tidak masalah jika Anda menggunakan kabel dengan warna berbeda atau menggunakan lubang yang berbeda pada papan tempat Breadboard selama komponen dan kabel tersambung dalam urutan yang sama seperti gambar.

Pastikan bahwa LED Anda terhubung dengan cara yang benar dengan kaki yang lebih panjang terhubung ke Digital Pin 10. Led panjang adalah Anoda LED dan harus selalu menuju ke suplai + 5v (dalam hal ini keluar dari Pin Digital 10) dan kaki pendek adalah Katoda dan harus ke Gnd (Ground).

Ketika Anda yakin semuanya telah terhubung dengan benar, nyalakan Arduino dan hubungkan kabel USB.

Masukkan kode

Sekarang, buka Arduino IDE dan ketik kode berikut:

```
// Proyek 1 - LED Berkedip

int ledPin = 10;
void setup() {
  pinMode(ledPin, OUTPUT) ;
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Sekarang tekan tombol Verify / Compile di bagian atas IDE untuk memastikan tidak ada kesalahan dalam kode Anda.

Jika ini berhasil, Anda sekarang dapat mengklik tombol Unggah untuk mengupload kode ke Arduino Anda. Jika Anda telah melakukan semuanya dengan benar, sekarang Anda akan melihat LED Merah menyala dan mati setiap detik.

Sekarang mari kita lihat kode dan perangkat kerasnya dan temukan bagaimana keduanya bekerja.

Proyek 1 – Ikhtisar Kode

```
// Proyek 1 - LED Berkedip

int ledPin = 10;
void setup() {
  pinMode(ledPin, OUTPUT) ;
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Jadi mari kita lihat kode untuk proyek ini. Baris pertama kita adalah

```
// Proyek 1 - LED Berkedip
```

Ini hanyalah sebuah komentar dalam kode Anda dan diabaikan oleh kompiler (bagian dari IDE yang mengubah kode Anda menjadi instruksi yang dapat dipahami Arduino sebelum mengunggahnya). Teks apa pun yang dimasukkan di belakang perintah `//` akan diabaikan oleh kompiler dan hanya ada untuk Anda, atau siapa pun yang membaca kode Anda. Komentar sangat penting dalam kode Anda untuk membantu Anda memahami apa yang sedang terjadi dan bagaimana kode Anda bekerja. Komentar juga dapat diletakkan setelah perintah seperti pada baris program berikutnya.

Nanti saat proyek Anda menjadi lebih kompleks dan kode Anda berkembang menjadi ratusan atau mungkin ribuan baris, komentar akan sangat penting dalam memudahkan Anda untuk melihat cara kerjanya. Anda mungkin mendapatkan potongan kode yang luar biasa, tetapi jika Anda kembali dan melihat kode itu di hari, minggu atau bulan selanjutnya, Anda mungkin lupa bagaimana semuanya bekerja.

Komentar akan membantu Anda memahaminya dengan mudah. Juga, jika kode Anda dimaksudkan untuk dilihat oleh orang lain (dan sebagai keseluruhan etos Arduino, dan memang seluruh komunitas Open Source adalah untuk berbagi kode dan skema. Kami berharap ketika Anda mulai membuat proyek keren Anda sendiri dengan Arduino Anda akan bersedia untuk membaginya dengan dunia) kemudian komentar akan memungkinkan orang tersebut untuk memahami apa yang sedang terjadi dalam kode Anda.

Anda juga dapat memasukkan komentar ke dalam pernyataan blok dengan menggunakan perintah `/*` dan `*/`. Misalnya.

```
/* Semua teks di dalam garis miring dan  
tanda bintang adalah komentar dan  
akan diabaikan oleh compiler */
```

IDE akan secara otomatis mengubah warna teks yang dikomentari menjadi abu-abu.

Baris berikutnya dalam program adalah `int ledPin = 10;`

Ini adalah yang dikenal sebagai variabel. Variabel adalah tempat menyimpan data. Dalam hal ini Anda menyiapkan variabel bertipe `int` atau `integer`. Bilangan bulat adalah angka dalam kisaran -32.768 hingga 32.767. Selanjutnya Anda telah menetapkan `integer` itu nama `ledPin` dan telah memberinya nilai 10. Kita tidak perlu menyebutnya `ledPin`, kita bisa menyebutnya apa pun yang kita inginkan. Tapi, karena kita ingin nama variabel kita menjadi deskriptif, kita menyebutnya `ledPin` untuk menunjukkan bahwa penggunaan variabel ini adalah untuk mengatur pin mana pada Arduino yang akan kita gunakan untuk menghubungkan LED kita. Dalam hal ini kita menggunakan Digital Pin 10. Di akhir pernyataan ini adalah titik koma. Ini adalah simbol untuk memberi tahu kompiler bahwa pernyataan ini sekarang sudah lengkap.

Meskipun kita dapat memanggil variabel kita apapun yang kita inginkan, setiap nama variabel di bahasa C harus dimulai dengan huruf, nama lainnya dapat terdiri dari huruf, angka dan karakter garis bawah. Bahasa C mengenali karakter huruf besar dan kecil sebagai karakter yang berbeda. Terakhir, Anda tidak dapat menggunakan kata kunci/keyword bahasa C seperti **main**, **while**, **switch** dll sebagai nama variabel. Kata kunci adalah konstanta, variabel dan nama fungsi yang didefinisikan sebagai bagian dari bahasa Arduino. Jangan gunakan nama variabel yang sama dengan kata kunci. Semua kata kunci di dalam sketch akan muncul dengan **warna merah**.

Jadi, Anda telah menyiapkan area dalam memori untuk menyimpan sejumlah tipe integer dan telah menyimpan di area itu angka 10. Bayangkan sebuah variabel sebagai kotak kecil tempat Anda dapat menyimpan sesuatu. Variabel disebut variabel karena Anda dapat mengubahnya. Nanti kami akan melakukan perhitungan matematis pada variabel untuk membuat program kami melakukan hal-hal yang lebih maju. Selanjutnya kita memiliki fungsi `setup()`

```
void setup() {  
  pinMode(ledPin, OUTPUT) ;  
}
```

Sketch Arduino harus memiliki fungsi `setup()` dan `loop()` jika tidak maka tidak akan berfungsi. Fungsi `setup()` dijalankan sekali dan sekali hanya pada awal program dan di sinilah Anda akan mengeluarkan instruksi umum untuk mempersiapkan program sebelum main loop berjalan, seperti mengatur mode pin, mengatur serial baud rate, dll.

Fungsi `setup` kita hanya memiliki satu pernyataan dan itu adalah `pinMode`. Di sini kita memberi tahu Arduino bahwa kita ingin mengatur mode salah satu pin digital kita menjadi mode Output, bukan Input. Di dalam tanda kurung kita memasukkan nomor pin dan mode (OUTPUT atau INPUT). Nomor pin kita adalah `ledPin`, yang sebelumnya telah diatur ke nilai 10 di program kami. Oleh karena itu,

pernyataan ini hanya memberi tahu Arduino bahwa Pin Digital 10 harus disetel ke mode OUTPUT.

Karena fungsi setup () hanya berjalan sekali, sekarang kita pindah ke loop fungsi utama.

Fungsi loop () adalah fungsi program utama dan berjalan terus menerus selama Arduino kita dihidupkan. Setiap pernyataan dalam fungsi loop () (dalam tanda kurung kurawal) dilakukan, satu per satu, selangkah demi selangkah, hingga bagian bawah fungsi tercapai, kemudian loop dimulai lagi di bagian atas fungsi, dan seterusnya selamanya atau sampai Anda mematikan Arduino atau menekan tombol Reset.

Dalam proyek ini kami ingin LED menyala, tetap menyala selama satu detik, mati dan tetap mati selama satu detik, lalu ulangi. Oleh karena itu, perintah untuk memberi tahu Arduino untuk melakukan yang terkandung dalam fungsi loop () seperti yang kita ingin mereka ulangi berulang kali. Pernyataan pertama adalah

```
digitalWrite(ledPin, HIGH);
```

dan ini menulis nilai HIGH / TINGGI atau LOW / RENDAH ke pin digital dalam pernyataan (dalam hal ini ledPin, yaitu Pin Digital 10).

Saat Anda menyetel pin digital ke TINGGI, Anda mengirimkan 5 volt ke pin itu. Saat Anda menyetelnya ke LOW pin menjadi 0 volt, atau Ground.

Oleh karena itu pernyataan ini mengirimkan 5v ke pin digital 10 dan menyalakan LED. Setelah itu

```
delay(1000);
```






dan pernyataan ini hanya memberitahu Arduino untuk menunggu selama 1000 milidetik (sampai 1 detik karena ada 1000 milidetik dalam satu detik) sebelum menjalankan pernyataan berikutnya yaitu

```
digitalWrite(ledPin, LOW);
```

yang akan mematikan daya ke pin 10 digital dan karenanya mematikan LED. Kemudian ada pernyataan penundaan lain selama 1000 milidetik lagi dan kemudian fungsi berakhir. Namun, karena ini adalah fungsi loop () utama kita, fungsi tersebut sekarang akan dimulai lagi dari awal. Dengan mengikuti struktur program selangkah demi selangkah lagi kita dapat melihat bahwa itu sangat sederhana.

Proyek 1 – Ikhtisar Perangkat Keras

Perangkat keras yang digunakan untuk proyek ini adalah:

| | |
|---------------------------------|---|
| Solderless Breadboard 400 Point |  |
| Led Merah - Nyala Merah 3mm |  |
| Resistor 220Ω |  |
| Jumper Kabel Dupont 10cm |  |

Breadboard adalah perangkat tanpa solder yang dapat digunakan kembali yang umumnya digunakan untuk membuat prototipe sirkuit elektronik atau untuk bereksperimen dengan desain sirkuit. Papan terdiri dari serangkaian lubang dalam kisi dan di bawah papan lubang-lubang ini dihubungkan dengan strip logam konduktif. Cara penataan strip tersebut biasanya seperti ini:



Strip di sepanjang bagian atas dan bawah sejajar dengan papan dan dirancang untuk jalur daya (+) dan jalur ground (-). Komponen di tengah papan dapat dengan mudah terhubung ke 5v (atau tegangan apa pun yang Anda gunakan) dan Ground. Beberapa Breadboard memiliki garis merah dan hitam yang sejajar dengan lubang-lubang ini untuk menunjukkan mana yang merupakan daya (Merah) dan mana yang merupakan ground (Hitam). Pada Breadboard yang lebih besar, jalur power terkadang terbelah, ditandai dengan putusnya garis merah. Ini jika Anda ingin voltase yang berbeda mengalir ke berbagai bagian papan Anda. Jika Anda hanya menggunakan satu voltase, sepotong kabel jumper pendek dapat ditempatkan melintasi celah ini untuk memastikan voltase yang sama diterapkan di sepanjang jalur.



Strip di tengah yang membentuk 90 derajat ke jalur listrik dan ground dalam jarak pendek dan ada celah di tengah untuk memungkinkan Anda memasang IC melintasi celah dan setiap pin chip ke satu set lubang yang berbeda dan karena itu jalur yang berbeda.

Komponen selanjutnya yang kita miliki adalah Resistor. Resistor adalah komponen yang dirancang untuk menyebabkan 'hambatan' ke arus listrik dan karena itu menyebabkan penurunan tegangan di terminalnya. Jika Anda membayangkan resistor seperti pipa air yang jauh lebih kecil dari pipa yang terhubung dengannya. Saat air (arus listrik) masuk ke resistor, pipa menjadi lebih kecil dan arus yang keluar dari ujung lainnya berkurang. Kita menggunakan resistor untuk menurunkan tegangan atau arus ke perangkat lain. Nilai resistansi dikenal sebagai Ohm dan simbolnya adalah simbol Omega yunani Ω .

Dalam hal ini Digital Pin 10 mengeluarkan 5 volt DC pada (menurut datasheet Atmega) 40mA (milliamps) dan LED kami membutuhkan (menurut datasheetnya) tegangan 2v dan arus 20mA. Oleh karena itu kita perlu memasang resistor yang akan mengurangi 5v menjadi 2v dan arus dari 40mA menjadi 20mA jika kita ingin menampilkan LED pada kecerahan maksimumnya. Jika kita ingin LED menjadi redup kita bisa menggunakan nilai resistansi yang lebih tinggi.

Untuk mengetahui resistor apa yang perlu kita lakukan ini, kita menggunakan apa yang disebut hukum Ohm yaitu $I = V / R$ di mana I adalah arus, V adalah tegangan dan R adalah resistansi. Jadi untuk menghitung hambatannya kita susun rumusnya menjadi $R = V / I$ yaitu $R = 3 / 0,02$ yaitu 150 Ohm. V adalah 3 karena kita membutuhkan Penurunan Tegangan, yang

PERINGATAN: Selalu pasang resistor (umumnya dikenal sebagai resistor pembatas arus) secara seri dengan LED. Jika Anda gagal melakukan ini, Anda akan memasok terlalu banyak arus ke LED dan itu dapat meledakkan atau merusak rangkaian Anda.

merupakan tegangan suplai (5v) dikurangi Tegangan Maju (2v) dari LED (terdapat di datasheet LED) yaitu 3v. Oleh karena itu kita perlu mencari resistor 150Ω. Jadi bagaimana kita melakukannya?

Sebuah resistor terlalu kecil untuk ditulisi sehingga dapat dibaca oleh kebanyakan orang, jadi resistor menggunakan kode warna. Di sekitar resistor Anda biasanya akan menemukan 4 pita berwarna dan dengan menggunakan kode warna pada grafik di halaman berikutnya Anda dapat mengetahui nilai resistor atau kode warna apa yang akan menunjukkan nilai dari resistansi.

| Warna | Pita pertama | Pita kedua | Pita ketiga (pengali) | Pita keempat (toleransi) | Pita kelima (koefisien suhu) |
|-----------------|--------------|------------|--------------------------|-----------------------------|---------------------------------|
| Hitam | 0 | 0 | $\times 10^0$ | | |
| Cokelat | 1 | 1 | $\times 10^1$ | $\pm 1\%$ (F) | 100 ppm |
| Merah | 2 | 2 | $\times 10^2$ | $\pm 2\%$ (G) | 50 ppm |
| Jingga (oranye) | 3 | 3 | $\times 10^3$ | | 15 ppm |
| Kuning | 4 | 4 | $\times 10^4$ | | 25 ppm |
| Hijau | 5 | 5 | $\times 10^5$ | $\pm 0.5\%$ (D) | |
| Biru | 6 | 6 | $\times 10^6$ | $\pm 0.25\%$ (C) | |
| Ungu | 7 | 7 | $\times 10^7$ | $\pm 0.1\%$ (B) | |
| Abu-abu | 8 | 8 | $\times 10^8$ | $\pm 0.05\%$ (A) | |
| Putih | 9 | 9 | $\times 10^9$ | | |
| Emas | | | $\times 10^{-1}$ | $\pm 5\%$ (J) | |
| Perak | | | $\times 10^{-2}$ | $\pm 10\%$ (K) | |
| Kosong | | | | $\pm 20\%$ (M) | |

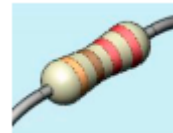
Kita membutuhkan resistor 150Ω, jadi jika kita melihat tabel warna kita melihat bahwa kita membutuhkan 1 di pita pertama, yaitu Coklat, diikuti dengan 5 di pita berikutnya yaitu Hijau dan kemudian kita perlu mengalikannya dengan 101 (dengan kata lain tambahkan 1 nol) yang berwarna Coklat di pita ke-3. Pita terakhir tidak relevan untuk tujuan kami karena ini adalah toleransi. Resistor kami memiliki pita emas dan oleh karena itu memiliki toleransi $\pm 5\%$ yang berarti nilai sebenarnya

dari resistor dapat bervariasi antara $142,5\Omega$ dan $157,5\Omega$. Oleh karena itu kita membutuhkan resistor dengan kombinasi pita warna Coklat, Hijau, Coklat, Emas yang terlihat seperti ini:

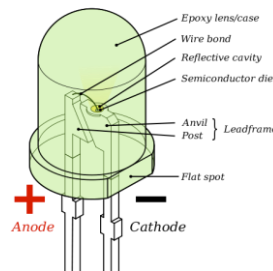


Jika kita membutuhkan resistor 1K (atau 1 kilo-ohm) kita akan membutuhkan kombinasi Coklat, Hitam, Merah (1, 0, +2 nol). Jika kita membutuhkan resistor 570K, warnanya adalah Hijau, Violet dan Kuning.

Dengan cara yang sama, jika Anda menemukan resistor dan ingin mengetahui nilainya, Anda akan melakukan hal yang sama secara terbalik. Jadi jika Anda menemukan resistor ini dan ingin mengetahui nilainya sehingga Anda dapat menyimpannya di kotak penyimpanan resistor yang diberi label dengan baik, kita dapat melihat tabel untuk melihatnya memiliki nilai 220Ω .



Komponen terakhir kami adalah LED, yang merupakan singkatan dari Light Emitting Diode. Dioda adalah komponen yang memungkinkan arus mengalir hanya dalam satu arah. Jadi, ini seperti katup dalam sistem air, tetapi dalam hal ini membiarkan arus listrik mengalir ke satu arah, tetapi jika arus mencoba untuk berbalik dan kembali ke arah yang berlawanan, dioda akan menghentikannya.



LED adalah hal yang sama, tetapi juga memancarkan cahaya. LED datang dalam berbagai warna dan kecerahan berbeda dan juga dapat memancarkan cahaya di bagian spektrum ultraviolet dan inframerah (seperti pada LED di remote control TV Anda).

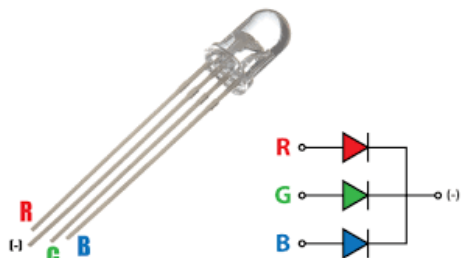
Jika Anda memperhatikan LED dengan cermat, Anda akan melihat dua hal. Salah satunya adalah bahwa kaki memiliki panjang yang berbeda dan juga di satu sisi LED, bukannya silinder, itu diratakan. Ini adalah

indikator untuk menunjukkan kepada Anda kaki mana yang merupakan Anoda (Positif) dan mana yang merupakan Katoda (Negatif). Kaki yang lebih panjang terhubung ke tegangan Positif (3.3v) dan kaki dengan sisi yang pendek terhubung ke Ground.

Jika Anda menghubungkan LED dengan cara yang salah, itu tidak akan merusaknya (kecuali Anda mengalirkan arus yang sangat tinggi) dan memang Anda dapat menggunakan 'fitur' itu seperti yang akan kita lihat nanti.

Selain LED satu warna, Anda juga bisa mendapatkan LED dua warna dan tiga warna. Ini akan memiliki beberapa kaki keluar dari mereka dengan salah satunya adalah common (yaitu common anoda atau common katoda).

Kit Anda dilengkapi dengan LED RGB, yang merupakan 3 LED dalam satu paket. Sebuah LED RGB memiliki LED Merah, Hijau dan Biru (karenanya RGB/ RED, Green dan Blue) dalam satu paket. LED memiliki 4 kaki, satu akan menjadi common anoda atau katoda, common untuk semua 3 LED dan 3 lainnya akan ke anoda atau katoda dari masing-masing LED Merah, Hijau dan Biru. Dengan menyesuaikan nilai kecerahan saluran R, G, dan B dari LED RGB, Anda bisa mendapatkan warna apa pun yang Anda inginkan. Efek yang sama dapat diperoleh jika Anda menggunakan 3 LED merah, hijau, dan biru yang terpisah.



Sekarang setelah Anda mengetahui cara kerja komponen dan cara kerja kode dalam proyek ini, mari kita coba sesuatu yang sedikit lebih menarik.

Proyek 2 - Signal Kode SOS Morse

Untuk proyek ini kita akan menggunakan rangkaian yang sama persis seperti pada Proyek 1, tetapi akan menggunakan beberapa kode yang berbeda untuk membuat LED menampilkan pesan dalam Kode Morse. Dalam hal ini, kita akan mendapatkan LED untuk memberi sinyal huruf S.O.S., yang merupakan sinyal marabahaya kode morse internasional.

Kode Morse adalah jenis pengkodean karakter yang mengirimkan huruf dan angka menggunakan pola On dan Off. Oleh karena itu, ini sangat cocok untuk sistem digital kita karena kita dapat menyalakan dan mematikan LED dalam pola yang diperlukan untuk mengeja kata atau serangkaian karakter. Dalam hal ini kami akan menandakan S.O.S. yang dalam alfabet Kode Morse adalah tiga dit (flash pendek), diikuti oleh tiga dah (flash panjang), diikuti oleh tiga dits lagi..

Karena itu, kami sekarang dapat mengkodekan sketsa kami untuk menyalakan dan mematikan LED dalam pola ini, menandakan SOS.



Masukkan kodenya

Buat sketsa baru lalu ketik kode yang tercantum di bawah. Verifikasi kode Anda bebas dari kesalahan dan kemudian unggah ke Arduino Anda.

Jika semua berjalan dengan baik Anda sekarang akan melihat LED berkedip sinyal SOS Kode Morse, tunggu 5 detik, lalu ulangi.

Jika Anda akan memasang Arduino yang dioperasikan dengan baterai ke cahaya yang sangat terang dan kemudian menempatkan seluruh

rakitan ke dalam kotak tahan air dan genggam, kode ini dapat digunakan untuk mengontrol lampu sorot darurat SOS untuk digunakan di kapal, saat mendaki gunung, dll.

```
// Proyek 2 - Signal Kode SOS Morse

// LED connected to digital pin 10
int ledPin = 10;

// run once, when the sketch starts
void setup()
{
    // sets the digital pin as output
    pinMode(ledPin, OUTPUT);
}

// run over and over again
void loop()
{
    // 3 dits
    for (int x = 0; x < 3; x++) {
        digitalWrite(ledPin, HIGH);    // sets the LED on
        delay(150);                    // waits for 150ms
        digitalWrite(ledPin, LOW);     // sets the LED off
        delay(100);                    // waits for 100ms
    }

    // 100ms delay to cause slight gap between letters
    delay(100);

    // 3 dahs
    for (int x = 0; x < 3; x++) {
        digitalWrite(ledPin, HIGH);    // sets the LED on
        delay(400);                    // waits for 400ms
        digitalWrite(ledPin, LOW);     // sets the LED off
        delay(100);                    // waits for 100ms
    }

    // 100ms delay to cause slight gap between letters
    delay(100);

    // 3 dits again
    for (int x = 0; x < 3; x++) {
        digitalWrite(ledPin, HIGH);    // sets the LED on
        delay(150);                    // waits for 150ms
    }
}
```



```

    digitalWrite(ledPin, LOW);    // sets the LED off
    delay(100);                  // waits for 100ms
}

// wait 5 seconds before repeating the SOS signal
delay(5000);
}

```

Proyek 2 - Ikhtisar Kode

Jadi bagian pertama dari kode ini identik dengan proyek terakhir di mana kita menginisialisasi variabel dan kemudian menetapkan pin 10 sebagai keluaran. Dalam loop kode utama kita dapat melihat jenis pernyataan yang sama untuk menyalakan dan mematikan LED untuk jangka waktu tertentu, tetapi kali ini pernyataan tersebut berada dalam 3 blok kode terpisah.

Blok pertama adalah yang menghasilkan 3 dit

```

// 3 dits
for (int x = 0; x < 3; x++) {
    digitalWrite(ledPin, HIGH);    // sets the LED on
    delay(150);                    // waits for 150ms
    digitalWrite(ledPin, LOW);     // sets the LED off
    delay(100);                    // waits for 100ms
}

```

Kita dapat melihat bahwa LED dihidupkan selama 150ms dan kemudian mati selama 100ms dan kita dapat melihat bahwa pernyataan tersebut berada dalam satu set kurung kurawal dan karenanya berada dalam blok kode yang terpisah. Tapi, saat kita menjalankan sketsa kita bisa melihat lampu berkedip 3 kali tidak hanya sekali.

Ini dilakukan dengan menggunakan for loop.

```

for (int x = 0; x < 3; x++) {

```

Ini dilakukan dengan menggunakan for loop.

Pernyataan inilah yang membuat kode di dalamnya dieksekusi sebanyak 3 kali. Ada 3 parameter yang perlu kita berikan pada loop for. Ini adalah inisialisasi, kondisi, kenaikan. Inisialisasi terjadi pertama kali dan tepat sekali. Setiap kali melalui loop, kondisi diuji; jika benar, blok pernyataan, dan kenaikan dijalankan, maka kondisinya diuji lagi. Ketika kondisi menjadi salah, loop berakhir.

Jadi, pertama-tama kita perlu menginisialisasi variabel menjadi nomor awal loop. Dalam hal ini kami menyiapkan variabel X dan menyetelnya ke nol.

```
int x = 0;
```

Kami kemudian menetapkan kondisi untuk memutuskan berapa kali kode dalam loop akan dieksekusi.

```
x < 3;
```

Dalam hal ini kode akan mengulang jika X lebih kecil dari (<) 3. Kode di dalam perulangan for akan selalu dijalankan sekali tidak peduli apa kondisinya.

Simbol < inilah yang dikenal sebagai ‘operator perbandingan’. Mereka digunakan untuk membuat keputusan dalam kode Anda dan untuk membandingkan dua nilai. Simbol yang digunakan adalah: -

```
== (sama dengan)
!= (tidak sama dengan)
< (kurang dari)
> (lebih dari)
<= (kurang dari atau sama dengan)
>= (lebih besar dari atau sama dengan)
```

Dalam kode kita, kita membandingkan x dengan nilai 3 untuk melihat apakah lebih kecil dari 3. Jika x lebih kecil dari 3, maka kode di blok akan berulang lagi.

Pernyataan terakhirnya adalah

```
x++
```

ini adalah pernyataan untuk meningkatkan nilai x sebesar 1. Kita juga bisa menentikkan $x = x + 1$; yang akan memberikan nilai x ke $x + 1$. Perhatikan bahwa tidak perlu meletakkan titik koma setelah pernyataan akhir ini di loop for.

Anda dapat mengerjakan matematika sederhana menggunakan simbol +, -, * dan / (penjumlahan, pengurangan, perkalian dan pembagian). Misalnya.

```
1 + 1 = 2
3 - 2 = 1
2 * 4 = 8
8 / 2 = 4
```

Jadi, loop for kami menginisialisasi nilai x ke 0, lalu menjalankan kode di dalam blok (tanda kurung kurawal). Ini kemudian meningkatkan kenaikan, dalam hal ini menambahkan 1 ke x. Akhirnya ia kemudian memeriksa bahwa kondisinya terpenuhi, yaitu x lebih kecil dari 3 dan jika demikian diulangi.

Jadi, sekarang kita tahu cara kerja for loop, kita dapat melihat di kode kita bahwa ada 3 for loop, yang loop 3 kali dan menampilkan 'dits', yang berikutnya mengulang 3 kali dan menampilkan 'dahs', lalu ada pengulangan dari dit's lagi.

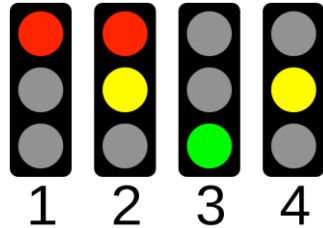
Harus diperhatikan bahwa variabel x memiliki 'scope' lokal, yang berarti hanya dapat dilihat oleh kode di dalam blok kodenya sendiri. Kecuali Anda menginisiasinya sebelum fungsi setup () yang

memiliki 'lingkup global' dan dapat dilihat oleh seluruh program. Jika Anda mencoba mengakses `x` di luar perulangan `for`, Anda akan mendapatkan pesan kesalahan.







Di antara setiap loop `for` ada penundaan kecil untuk membuat jeda kecil yang terlihat di antara huruf SOS. Terakhir, kode menunggu selama 5 detik sebelum loop program utama dimulai lagi dari awal. OK sekarang mari kita lanjutkan menggunakan beberapa LED.

Proyek 3 - Lampu Lalu Lintas

Kami sekarang akan membuat satu set lampu lalu lintas Inggris yang akan berubah dari hijau menjadi merah, melalui kuning, dan kembali lagi, setelah jangka waktu tertentu menggunakan sistem 4-status. Proyek ini dapat digunakan pada model rel kereta api untuk membuat satu set lampu lalu lintas yang berfungsi atau untuk kota mainan anak-anak.



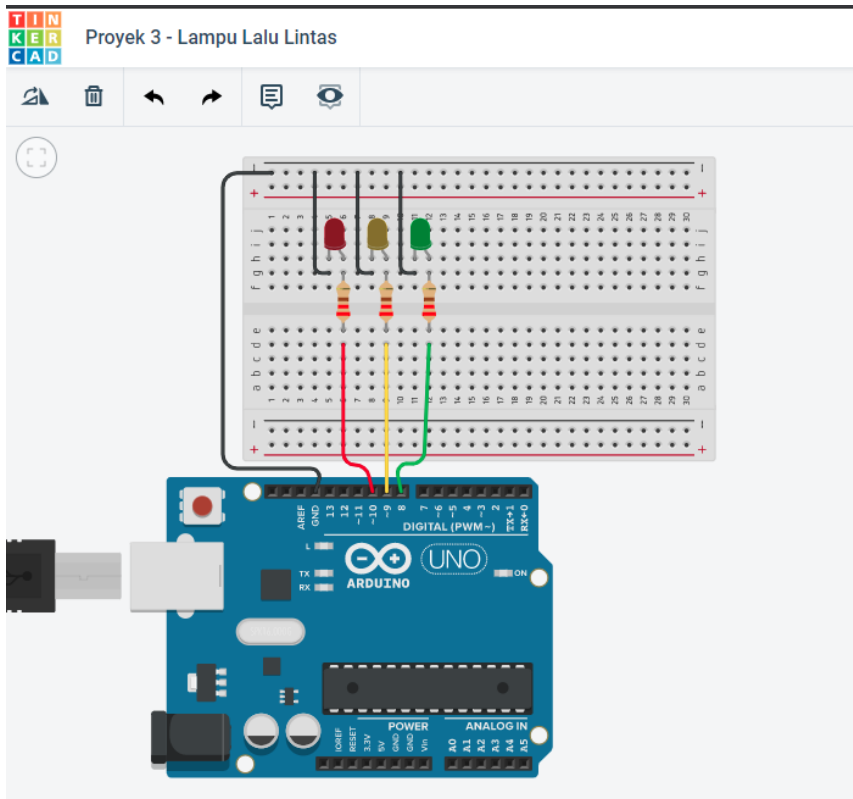
Apa yang Anda butuhkan

| | |
|---------------------------------|---|
| Solderless Breadboard 400 Point |  |
| Led Merah - Nyala Merah 3mm |  |
| Led Kuning - Nyala Kuning 3mm |  |
| Led Hijau - Nyala Hijau 3mm |  |
| Resistor 220Ω (3 pcs) |  |
| Jumper Kabel Dupont 10cm |  |

Membuat Rangkaian

Kali ini kita menghubungkan 3 LED dengan Anoda masing-masing menuju ke Pin Digital 8, 9 dan 10, masing-masing melalui resistor 220 Ω .

Kita hubungkan kabel jumper dari Ground Arduino ke ground rail di bagian atas breadboard dan kabel ground dari kaki Katoda setiap LED ke rel ground bersama.



Masukkan kode

```
// Proyek 3 - Lampu Lalu Lintas

int ledDelay = 10000; // delay in between changes
int redPin = 10;
int yellowPin = 9;
int greenPin = 8;

void setup() {
    pinMode(redPin, OUTPUT);
    pinMode(yellowPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
}

void loop() {

    // turn the red light on
    digitalWrite(redPin, HIGH);
    delay(ledDelay); // wait 5 seconds

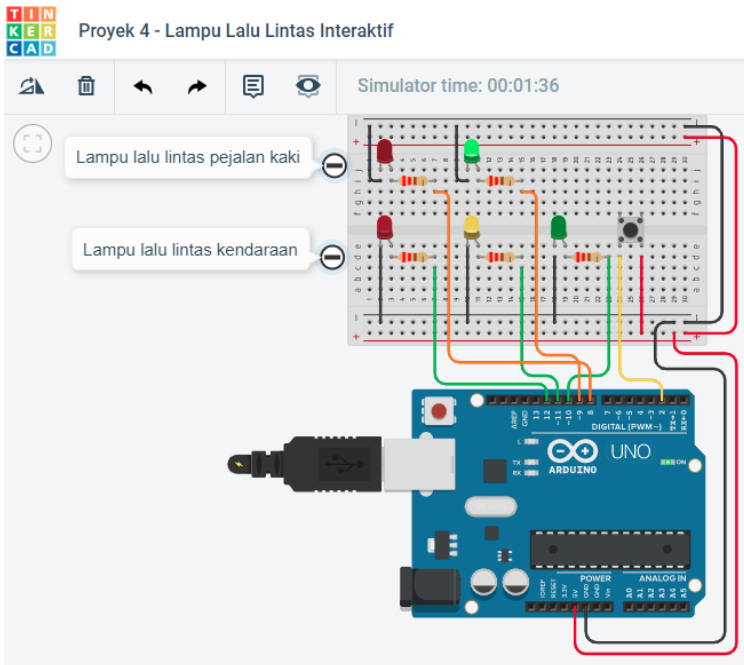
    digitalWrite(yellowPin, HIGH); // turn on yellow
    delay(2000); // wait 2 seconds

    digitalWrite(greenPin, HIGH); // turn green on
    digitalWrite(redPin, LOW); // turn red off
    digitalWrite(yellowPin, LOW); // turn yellow off
    delay(ledDelay); // wait ledDelay milliseconds

    digitalWrite(yellowPin, HIGH); // turn yellow on
    digitalWrite(greenPin, LOW); // turn green off
    delay(2000); // wait 2 seconds

    digitalWrite(yellowPin, LOW); // turn yellow off
    // now our loop repeats }
}
```

Proyek 4 - Lampu Lalu Lintas Interaktif







Kali ini kami akan memperluas proyek sebelumnya dengan menyertakan satu set lampu pejalan kaki dan tombol tekan pejalan kaki untuk meminta menyeberang jalan. Arduino akan bereaksi ketika tombol ditekan dengan mengubah keadaan lampu untuk membuat mobil berhenti dan memungkinkan pejalan kaki untuk menyeberang dengan aman.

Untuk pertama kalinya kami dapat berinteraksi dengan Arduino dan menyebabkannya melakukan sesuatu ketika kami mengubah status tombol yang sedang ditonton Arduino (yaitu Tekan untuk mengubah status dari terbuka menjadi tertutup). Dalam proyek ini kita juga akan belajar bagaimana membuat fungsi kita sendiri.

Mulai sekarang saat menghubungkan komponen, kami tidak akan lagi mencantumkan breadboard dan kabel jumper. Anggap saja sebagai sudah dibaca bahwa Anda akan selalu membutuhkan keduanya..

Apa yang Anda butuhkan

| | |
|-------------------------------------|---|
| Led Merah - Nyala Merah 3mm (2 pcs) |  |
| Led Kuning - Nyala Kuning 3mm |  |
| Led Hijau - Nyala Hijau 3mm (2 pcs) |  |
| Resistor 220Ω (6 pcs) |  |
| Micro Switch Tactile Push Button |  |

Membuat Rangkaian

Hubungkan LED dan sakelar ke atas seperti pada diagram di halaman sebelumnya. Anda perlu memindahkan kabel dari pin 8, 9 dan 10 di proyek sebelumnya ke pin 10, 11 dan 12 untuk memungkinkan Anda menghubungkan lampu pejalan kaki ke pin 8 dan 9.

Masukkan kode

Masukkan kode, verifikasi dan unggah.

```

// Proyek 4 - Lampu Lalu Lintas Interaktif
int carRed = 12; // assign the car lights
int carYellow = 11;
int carGreen = 10;
int pedRed = 9; // assign the pedestrian lights
int pedGreen = 8;
int button = 2; // button pin
int crossTime = 5000; // time allowed to cross
unsigned long changeTime; // time since button pressed

void setup() {
    pinMode(carRed, OUTPUT);
    pinMode(carYellow, OUTPUT);
    pinMode(carGreen, OUTPUT);
    pinMode(pedRed, OUTPUT);
    pinMode(pedGreen, OUTPUT);
    pinMode(button, INPUT); // button on pin 2
    // turn on the green light
    digitalWrite(carGreen, HIGH);
    digitalWrite(pedRed, HIGH);
}

void loop() {
    int state = digitalRead(button);
    /* check if button is pressed and it is
       over 5 seconds since last button press */
    if (state == HIGH && (millis() - changeTime) > 5000) {
        // Call the function to change the lights
        changeLights();
    }
}

void changeLights() {
    digitalWrite(carGreen, LOW); // green off
    digitalWrite(carYellow, HIGH); // yellow on
    delay(2000); // wait 2 seconds

    digitalWrite(carYellow, LOW); // yellow off
    digitalWrite(carRed, HIGH); // red on
    delay(1000); // wait 1 second till its safe

    digitalWrite(pedRed, LOW); // ped red off
    digitalWrite(pedGreen, HIGH); // ped green on
    delay(crossTime); // wait for preset time period

    // flash the ped green

```

```

for (int x = 0; x < 10; x++) {
    digitalWrite(pedGreen, HIGH);
    delay(250);
    digitalWrite(pedGreen, LOW);
    delay(250);
}
// turn ped red on
digitalWrite(pedRed, HIGH);
delay(500);

digitalWrite(carYellow, HIGH); // yellow on
digitalWrite(carRed, LOW); // red off
delay(1000);
digitalWrite(carGreen, HIGH);
digitalWrite(carYellow, LOW); // yellow off

// record the time since last change of lights
changeTime = millis();
// then return to the main program loop }
}

```

Ketika Anda menjalankan program, Anda akan melihat bahwa lampu lalu lintas mobil mulai menyala hijau untuk memungkinkan mobil lewat dan lampu pejalan kaki menyala merah.

Saat Anda menekan tombol, program akan memeriksa bahwa setidaknya 5 detik telah berlalu sejak terakhir kali lampu diubah (untuk memungkinkan lalu lintas bergerak), dan jika demikian meneruskan eksekusi kode ke fungsi yang telah kita buat yang disebut `changeLights()`. Dalam fungsi ini lampu mobil berubah dari hijau menjadi kuning lalu merah, lalu lampu pejalan kaki menjadi hijau. Setelah jangka waktu yang ditentukan dalam variabel `crossTime` (waktu yang cukup untuk memungkinkan pejalan kaki menyeberang), lampu hijau pejalan kaki akan menyala dan mati sebagai peringatan kepada pejalan kaki untuk segera menyala karena lampu akan berubah kembali menjadi merah. Kemudian lampu pejalan kaki berubah kembali menjadi merah dan lampu kendaraan berubah dari merah menjadi kuning ke hijau dan lalu lintas dapat dilanjutkan.

Kode dalam proyek ini mirip dengan proyek sebelumnya. Namun, ada beberapa pernyataan dan konsep baru yang telah diperkenalkan, jadi mari kita lihat.

Proyek 4 - Ikhtisar Kode

Sebagian besar kode dalam proyek ini akan Anda pahami dan kenali dari proyek sebelumnya. Namun, mari kita lihat beberapa kata kunci dan konsep baru yang telah diperkenalkan dalam sketsa ini.

```
unsigned long changeTime; // time since button pressed
```

Di sini kami memiliki tipe data baru untuk variabel. Sebelumnya kami telah membuat tipe data integer, yang dapat menyimpan angka antara -32,768 dan 32,767. Kali ini kita telah membuat tipe data long, yang bisa menyimpan angka dari -2,147,483,648 hingga 2,147,483,647. Namun, kami telah menentukan panjang unsigned, yang berarti variabel tidak dapat menyimpan angka negatif, yang memberi kami rentang dari 0 hingga 4.294.967.295. Jika kita menggunakan bilangan bulat untuk menyimpan lama waktu sejak pergantian lampu terakhir, kita hanya akan mendapatkan waktu maksimum 32 detik sebelum variabel bilangan bulat mencapai angka yang lebih tinggi daripada yang bisa disimpan.

Karena penyeberangan pejalan kaki tidak mungkin digunakan setiap 32 detik, kami tidak ingin program kami mogok karena variabel kami 'meluap/ overflowing' saat mencoba menyimpan angka yang terlalu tinggi untuk tipe data variabel. Itulah mengapa kami menggunakan tipe data panjang unsigned karena kami sekarang mendapatkan waktu yang sangat lama di antara penekanan tombol.

4294967295 * 1ms = 4294967 seconds
4294967 seconds = 71582 minutes
71582 minutes - 1193 hours
1193 hours - 49 days

Karena tidak dapat dipungkiri bahwa penyeberangan pejalan kaki akan menekan tombolnya setidaknya sekali dalam 49 hari, kami seharusnya tidak mengalami masalah dengan tipe data ini.

Anda mungkin bertanya mengapa kami tidak hanya memiliki satu tipe data yang dapat menyimpan jumlah besar setiap saat dan diselesaikan dengannya. Nah, alasan kami tidak melakukan itu adalah karena variabel memakan ruang dalam memori dan semakin besar angkanya, semakin banyak memori yang digunakan untuk menyimpan variabel. Di PC atau laptop rumah Anda, Anda tidak perlu terlalu khawatirkan hal itu, tetapi pada mikrokontroler kecil seperti Atmega328 yang digunakan Arduino, penting bagi kami untuk hanya menggunakan jenis data variabel terkecil yang diperlukan untuk tujuan kami.

Ada berbagai tipe data yang dapat kita gunakan sebagai sketsa kita dan ini adalah: -

| Data type | RAM | Number Range |
|-------------------------------|------------|-------------------------|
| void keyword | N/A | N/A |
| boolean | 1 byte | 0 to 1 (True or False) |
| byte | 1 byte | 0 to 255 |
| char | 1 byte | -128 to 127 |
| unsigned char | 1 byte | 0 to 255 |
| int | 2 byte | -32,768 to 32,767 |
| unsigned int | 2 byte | 0 to 65,535 |
| word | 2 byte | 0 to 65,535 |
| long | 4 byte | -2,147,483,648 to |
| unsigned long | 4 byte | 0 to 4,294,967,295 |
| float | 4 byte | -3.4028235E+38 to |
| double | 4 byte | -3.4028235E+38 to |
| string | 1 byte + x | Arrays of chars |
| array | 1 byte + x | Collection of variables |

Setiap tipe data menggunakan sejumlah memori pada Arduino seperti yang Anda lihat pada tabel di atas. Beberapa variabel hanya menggunakan 1 byte memori dan yang lain menggunakan 4 atau lebih (jangan khawatir tentang apa itu byte untuk saat ini karena kita akan membahasnya nanti). Anda tidak dapat menyalin data dari satu tipe data ke tipe lainnya, mis. Jika x adalah int dan y adalah string maka $x = y$ tidak akan berfungsi karena kedua tipe data berbeda.

Atmega168 memiliki 1Kb (1000 bytes) dan Atmega328 memiliki 2Kb (2000 bytes) SRAM. Ini tidak banyak dan dalam program besar dengan banyak variabel, Anda dapat dengan mudah kehabisan memori jika Anda tidak mengoptimalkan penggunaan tipe data yang benar. Dari daftar di atas kita dapat melihat dengan jelas bahwa penggunaan tipe data int kita boros karena menggunakan hingga 2 byte dan dapat menyimpan angka hingga 32.767. Karena kami telah menggunakan int untuk menyimpan nomor pin digital kami, yang hanya akan setinggi 13 pada Arduino kami (dan hingga 54 pada Arduino Mega), kami telah menggunakan lebih banyak memori daripada yang diperlukan. Kita bisa menghemat memori dengan menggunakan tipe data byte, yang dapat menyimpan angka antara 0 dan 255, yang lebih dari cukup untuk menyimpan jumlah pin I / O.

Selanjutnya kita punya

```
pinMode(button, INPUT); // button on pin 2
```

Ini memberitahu Arduino bahwa kita ingin menggunakan Pin Digital 2 (tombol = 2) seperti pada INPUT. Kami akan menggunakan pin 2 untuk mendengarkan penekanan tombol sehingga modusnya perlu disetel ke input.

Di loop program utama kami memeriksa status digital pin 2 dengan pernyataan ini: -

```
int state = digitalRead(button);
```

Ini menginisialisasi integer (ya itu boros dan kita harus menggunakan boolean) yang disebut 'state' dan kemudian menetapkan nilai status menjadi nilai pin digital 2. Pernyataan `digitalRead` membaca status pin digital dalam tanda kurung dan mengembalikannya ke bilangan bulat yang telah kita tetapkan. Kami kemudian dapat memeriksa nilai dalam status untuk melihat apakah tombol telah ditekan atau belum.

```
int state = digitalRead(button);  
/* check if button is pressed and it is  
   over 5 seconds since last button press */  
if (state == HIGH && (millis() - changeTime) > 5000) {  
    // Call the function to change the lights  
    changeLights();  
}
```

Pernyataan `if` adalah contoh dari struktur kontrol dan tujuannya adalah untuk memeriksa apakah kondisi tertentu telah dipenuhi atau tidak dan jika demikian untuk mengeksekusi kode di dalam blok kodenya. Misalnya, jika kita ingin menyalakan LED jika variabel `x` naik di atas nilai 500 kita bisa menulis

```
if (x>500) {digitalWrite(ledPin, HIGH);
```

Saat kita membaca pin digital menggunakan perintah `digitalRead`, status pin akan menjadi HIGH atau LOW. Jadi perintah `if` dalam sketch kita terlihat seperti ini

```
if (state == HIGH && (millis() - changeTime) > 5000)
```

Apa yang kami lakukan di sini adalah memeriksa bahwa dua syarat telah terpenuhi. Yang pertama adalah variabel yang disebut status tinggi. Jika tombol telah ditekan statusnya akan tinggi karena kami telah mengaturnya menjadi nilai yang dibaca dari pin digital 2. Kami juga memeriksa bahwa nilai `millis() - changeTime` lebih besar dari 5000 (menggunakan perintah logika AND `&&`). Fungsi `millis()` adalah salah satu yang dibangun ke dalam bahasa Arduino dan

mengembalikan jumlah milidetik sejak Arduino mulai menjalankan program saat ini. Variabel `changeTime` kami awalnya tidak akan memiliki nilai, tetapi setelah fungsi `changeLights` dijalankan, kami menentukannya di akhir fungsi tersebut ke nilai `millis()` saat ini.

Dengan mengurangi nilai dalam variabel `changeTime` dari nilai `millis()` saat ini, kita dapat memeriksa apakah 5 detik telah berlalu sejak `changeTime` ditetapkan terakhir. Perhitungan `millis() - changeTime` diletakkan di dalam tanda kurung sendiri untuk memastikan bahwa kita membandingkan nilai `state` dan hasil dari perhitungan ini dan bukan nilai `millis()` dengan sendirinya. Simbol `' && '` di antaranya

```
state == HIGH
```

dan perhitungannya adalah contoh Operator Boolean. Dalam hal ini berarti AND. Untuk melihat apa yang kami maksud dengan itu, mari kita lihat semua Operator Boolean.

| | |
|-------------------------|------------|
| <code>&&</code> | Logika AND |
| <code> </code> | Logika OR |
| <code>!</code> | NOT |

Ini adalah pernyataan logika dan dapat digunakan untuk menguji berbagai kondisi dalam pernyataan `if`.

`&&` berarti benar jika kedua operan benar, mis. :

```
if (x == 5 && y == 10) {....
```

Pernyataan `if` ini akan menjalankan kode itu hanya jika `x` adalah 5 dan juga `y` adalah 10.

`||` Berarti benar jika salah satu operan benar, mis. :

```
if (x == 5 || y == 10) {.....
```


Ini akan berjalan jika x adalah 5 atau jika y adalah 10.

Tanda ! atau NOT berarti benar jika operan salah, mis. :

```
if (!x) {.....
```

Akan berjalan jika x salah, yaitu sama dengan nol.

Anda juga bisa 'nest' kondisi dengan tanda kurung,

```
misalnya if (x == 5 && (y == 10 || z == 25)) {.....
```

Dalam hal ini, ketentuan dalam tanda kurung diproses secara terpisah dan diperlakukan sebagai ketentuan tunggal, lalu dibandingkan dengan ketentuan kedua. Jadi, jika kita menggambar tabel kebenaran sederhana untuk pernyataan ini, kita dapat melihat cara kerjanya.

| x | y | z | True/False? |
|---|----|----|-------------|
| 4 | 9 | 25 | FALSE |
| 5 | 10 | 24 | TRUE |
| 7 | 10 | 25 | FALSE |
| 5 | 10 | 25 | TRUE |

Perintah dalam pernyataan if adalah

```
changeLights();
```

dan ini adalah contoh panggilan fungsi. Fungsi hanyalah blok kode terpisah yang telah diberi nama. Namun, fungsi dapat mengirimkan parameter dan / atau mengembalikan data juga. Dalam hal ini kami belum mengirimkan data apa pun ke fungsi atau kami memiliki fungsi yang mengembalikan tanggal apa pun. Kita akan membahas lebih detail nanti tentang melewati parameter dan mengembalikan data dari fungsi.

Saat `changeLights ()`; dipanggil, eksekusi kode melompat dari baris saat ini ke fungsi, mengeksekusi kode di dalam fungsi itu dan kemudian kembali ke titik dalam kode setelah fungsi dipanggil.

Jadi, dalam kasus ini, jika kondisi di pernyataan `if` terpenuhi, maka program akan mengeksekusi kode di dalam fungsi dan kemudian kembali ke baris berikutnya setelah `changeLights ()`; dalam pernyataan `if`.

Kode dalam fungsi tersebut hanya mengubah lampu kendaraan menjadi merah, melalui kuning, lalu menyalakan lampu hijau pejalan kaki. Setelah jangka waktu yang ditentukan oleh variabel `crossTime`, lampu berkedip beberapa kali untuk memperingatkan pejalan kaki bahwa waktunya akan segera habis, lalu lampu pejalan kaki menjadi merah dan lampu kendaraan berubah dari merah menjadi hijau, melalui kuning dan kembali ke itu keadaan normal.



Loop program utama hanya memeriksa terus menerus apakah tombol pejalan kaki telah ditekan atau tidak dan jika sudah, dan (`&&`) waktu sejak lampu terakhir diubah lebih besar dari 5 detik, itu memanggil fungsi `changeLights ()` lagi.

Dalam program ini tidak ada keuntungan dari menempatkan kode ke dalam fungsinya sendiri selain membuat kode terlihat lebih bersih. Hanya ketika suatu fungsi melewati parameter dan / atau mengembalikan data maka manfaat sebenarnya terungkap dan kita akan melihatnya nanti.

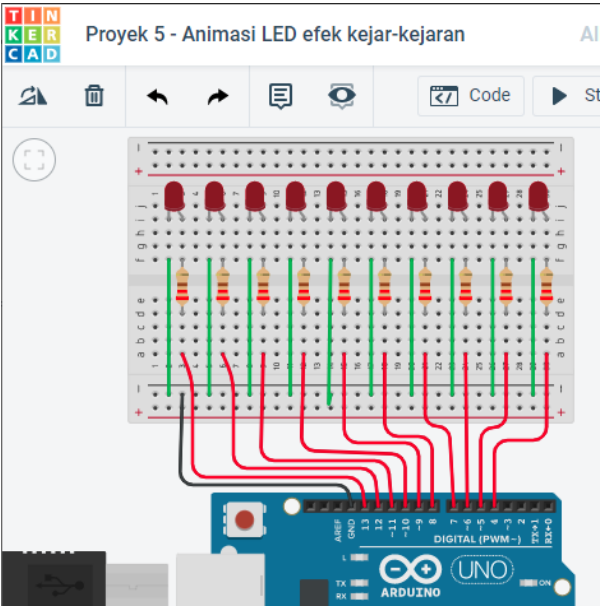
Proyek 5 - Animasi LED efek kejar-kejaran

Sekarang kita akan menggunakan rangkaian LED (total 10) untuk membuat LED efek kejar-kejaran

Apa yang Anda butuhkan

| | |
|--------------------------------------|---|
| Led Merah - Nyala Merah 3mm (10 pcs) |  |
| Resistor 220Ω (10 pcs) |  |

Membuat Rangkaian



Masukkan kode

```
// Proyek 5 - Animasi LED efek kejar-kejaran
// Create array for LED pins
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
int ledDelay(65); // delay between changes
int direction = 1;
int currentLED = 0;
unsigned long changeTime;

void setup() {
  // set all pins to output
  for (int x = 0; x < 10; x++) {
    pinMode(ledPin[x], OUTPUT); }
  changeTime = millis();
}

void loop() {
  // if it has been ledDelay ms since last change
  if ((millis() - changeTime) > ledDelay) {
    changeLED();
    changeTime = millis();
  }
}

void changeLED() {
  // turn off all LED's
  for (int x = 0; x < 10; x++) {
    digitalWrite(ledPin[x], LOW);
  }
  // turn on the current LED
  digitalWrite(ledPin[currentLED], HIGH);
  // increment by the direction value
  currentLED += direction;
  // change direction if we reach the end
  if (currentLED == 9) {direction = -1;}
  if (currentLED == 0) {direction = 1;}
}
```

Proyek 5 - Ikhtisar Kode

Baris pertama kami dalam sketsa ini adalah

```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
```

dan ini adalah deklarasi variabel array tipe data. Array adalah kumpulan variabel yang diakses menggunakan nomor indeks. Dalam sketsa kami, kami telah mendeklarasikan array tipe data byte dan menyebutnya ledPin. Kami kemudian menginisialisasi array dengan 10 nilai, yang merupakan pin digital 4 hingga 13. Untuk mengakses elemen array, kami cukup merujuk ke nomor indeks elemen itu. Array diindeks nol, yang berarti indeks pertama dimulai dari nol dan bukan 1. Jadi dalam 10 elemen array nomor indeks adalah 0 sampai 9.

Dalam hal ini, elemen 3 (ledPin [2]) memiliki nilai 6 dan elemen 7 (ledPin [6]) memiliki nilai 10.

Anda harus mengetahui ukuran array jika Anda tidak menginisialisasi dengan data terlebih dahulu. Dalam sketsa kami, kami tidak secara eksplisit memilih ukuran karena kompiler dapat menghitung nilai yang telah kami tetapkan ke array untuk mengetahui bahwa ukurannya adalah 10 elemen. Jika kita telah mendeklarasikan array tetapi tidak menjalankannya dengan nilai pada saat yang sama, kita perlu mendeklarasikan ukuran, misalnya kita bisa melakukan ini:

```
byte ledPin[10];
```

dan kemudian memuat data ke dalam elemen di kemudian hari. Untuk mengambil nilai dari array kita akan melakukan sesuatu seperti ini:

```
x = ledpin[5];
```

Dalam contoh ini x sekarang akan memiliki nilai 8. Untuk kembali ke program Anda, kami telah memulai dengan mendeklarasikan dan menginisialisasi sebuah array dan telah menyimpan 10 nilai yang merupakan pin digital yang digunakan untuk output ke 10 LED kami.

Dalam loop email kami, kami memeriksa bahwa setidaknya milidetik ledDelay telah berlalu sejak perubahan terakhir LED dan jika demikian ia melewati kontrol ke fungsi kami. Alasan kita hanya akan meneruskan kontrol ke fungsi changeLED () dengan cara ini, daripada menggunakan perintah delay (), adalah untuk mengizinkan kode lain jika diperlukan untuk dijalankan di loop program utama (selama kode itu membutuhkan waktu kurang dari ledDelay untuk menjalankan).

Fungsi yang kami buat adalah

```
void changeLED() {  
    // turn off all LED's  
    for (int x = 0; x < 10; x++) {  
        digitalWrite(ledPin[x], LOW);  
    }  
    // turn on the current LED  
    digitalWrite(ledPin[currentLED], HIGH);  
    // increment by the direction value  
    currentLED += direction;  
    // change direction if we reach the end  
    if (currentLED == 9) {direction = -1;}  
    if (currentLED == 0) {direction = 1;}  
}
```

dan tugas fungsi ini adalah mematikan semua LED dan kemudian menyalakan LED saat ini (ini dilakukan dengan sangat cepat sehingga Anda tidak akan melihatnya terjadi), yang disimpan dalam variabel currentLED.

Variabel ini kemudian ditambahkan arah ke dalamnya. Karena arah hanya dapat berupa 1 atau -1 maka angkanya akan meningkat (+1) atau berkurang satu (currentLED + (- 1)).

Kami kemudian memiliki pernyataan if untuk melihat apakah kami telah mencapai akhir baris LED dan jika demikian kami kemudian membalikkan variabel arah.


Dengan mengubah nilai `ledDelay` Anda dapat membuat ping LED bolak-balik pada kecepatan yang berbeda. Cobalah nilai yang berbeda untuk melihat apa yang terjadi.

Namun, Anda harus menghentikan program dan secara manual mengubah nilai `ledDelay` kemudian mengunggah kode yang telah diubah untuk melihat perubahan apa pun. Bukankah menyenangkan bisa menyesuaikan kecepatan saat program berjalan? Ya, jadi mari kita lakukan persis seperti itu di proyek berikutnya dengan memperkenalkan cara untuk berinteraksi dengan program dan menyesuaikan kecepatan menggunakan `trimpot`.

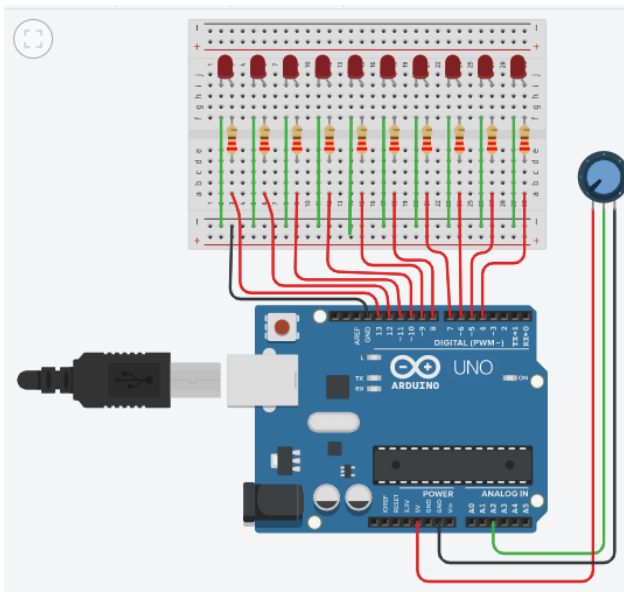
Proyek 6 - Animasi LED efek kejar-kejaran Interaktif

Sekarang kita akan menggunakan rangkaian LED (total 10) untuk membuat efek kejar-kejaran LED

Apa yang Anda butuhkan

| | |
|---|---|
| Komponen dari proyek sebelumnya plus | |
| Trimpot 4K7 |  |

Membuat Rangkaiannya.



Ini adalah sirkuit yang sama seperti di Proyek 5, tetapi kami hanya menambahkan trimpot dan menghubungkannya ke 5 v, Ground dan Analog Pin 5

Masukkan kode

```
// Create array for LED pins
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
int ledDelay; // delay between changes
int direction = 1; int
currentLED = 0;
unsigned long changeTime;
int potPin = 2; // select the input pin for the
potentiometer
void setup() {
    // set all pins to output
    for (int x = 0; x < 10; x++) {
        pinMode(ledPin[x], OUTPUT); }
    changeTime = millis();
}

void loop() {
    // read the value from the pot
    ledDelay = analogRead(potPin);
    // if it has been ledDelay ms since last change
    if ((millis() - changeTime) > ledDelay) {
        changeLED();
        changeTime = millis();
    }
}

void changeLED() {
    // turn off all LED's
    for (int x = 0; x < 10; x++) {
        digitalWrite(ledPin[x], LOW);
    }
    // turn on the current LED
    digitalWrite(ledPin[currentLED], HIGH);
    // increment by the direction value
    currentLED += direction;
    // change direction if we reach the end
    if (currentLED == 9) {
        direction = -1;
    }
}
```

```

    if (currentLED == 0) {
        direction = 1;
    }
}

```

This time when verify and upload your code, you should now see the lit LED appear to bounce back and forth between each end of the string of lights as before. But, by turning the knob of the potentiometer, you will change the value of `ledDelay` and speed up or slow down the effect. Let's take a look at how this works and find out what a potentiometer is.

Proyek 6 – Ikhtisar Kode

Kode untuk Proyek ini hampir sama dengan proyek sebelumnya. Kami baru saja menambahkan trimpot ke perangkat keras kami dan kode memiliki tambahan untuk memungkinkan kami membaca nilai dari trimpot dan menggunakannya untuk menyesuaikan kecepatan efek pengejaran LED.

Kami pertama-tama mendeklarasikan variabel untuk pin trimpot

```

int potPin = 2;    // select the input pin for the
trimpot

```

karena trimpot kita terhubung ke pin analog 2. Untuk membaca nilai dari pin analog kita menggunakan perintah `analogRead`. Arduino memiliki 6 input / output analog dengan konverter analog ke digital 10-bit (kita akan membahas bit nanti). Ini berarti pin analog dapat membaca dalam tegangan antara 0 hingga 5 volt dalam nilai integer antara 0 (0 volt) dan 1023 (5 volt). Ini memberikan resolusi 5 volt / 1024 unit atau 0,0049 volt (4,9mV) per unit.

Kami perlu mengatur penundaan kami menggunakan trimpot sehingga kami hanya akan menggunakan nilai langsung yang dibaca dari pin untuk menyesuaikan penundaan antara 0 dan 1023 milidetik. Kami melakukan ini dengan langsung membaca nilai pin trimpot ke dalam `ledDelay`. Perhatikan bahwa kita tidak perlu mengatur pin analog untuk menjadi input atau output seperti yang kita perlukan dengan pin digital.

```
ledDelay = analogRead(potPin);
```

Ini dilakukan selama loop utama kami dan oleh karena itu terus-menerus dibaca dan disesuaikan. Dengan memutar kenop, Anda dapat menyesuaikan nilai penundaan antara 0 dan 1023 milidetik (atau lebih dari satu detik) dan karena itu memiliki kendali penuh atas kecepatan efek.

Oke mari kita cari tahu apa itu trimpot dan cara kerjanya.

Proyek 6 - Ikhtisar Perangkat Keras

Ini dilakukan selama loop utama kami dan oleh karena itu terus-menerus dibaca dan disesuaikan. Dengan memutar kenop, Anda dapat menyesuaikan nilai penundaan antara 0 dan 1023 milidetik (atau lebih dari satu detik) dan karena itu memiliki kendali penuh atas kecepatan efek.

Oke mari kita cari tahu apa itu trimpot dan cara kerjanya.

Satu-satunya perangkat keras tambahan yang digunakan dalam proyek ini adalah trimpot 4K7 (4700Ω).

 digipart



Anda telah menemukan sebuah resistor dan tahu bagaimana caranya. Trimpot hanyalah resistor yang dapat disesuaikan dengan rentang dari 0 hingga nilai yang

ditetapkan (tertulis di sisi pot). Dalam kit Anda telah diberi trimpot 4K7 atau 4.700Ω yang berarti kisarannya dari 0 hingga 4700 Ohm.

Trimpot memiliki 3 kaki. Dengan menghubungkan hanya dua kaki trimpot menjadi resistor variabel. Dengan menghubungkan ketiga kaki dan menerapkan tegangan di atasnya, pot menjadi pembagi tegangan. Ini adalah cara kami menggunakannya di sirkuit kami. Satu sisi terhubung ke ground, yang lain ke 5v dan pin tengah ke pin analog kami.

Dengan mengatur kenop, tegangan antara 0 dan 5v akan keluar dari pin tengah dan kita dapat membaca nilai tegangan tersebut pada Pin Analog 2 dan menggunakannya untuk mengubah laju penundaan efek cahaya.



Trimpot dapat sangat berguna dalam menyediakan sarana untuk menyesuaikan nilai dari 0 ke jumlah yang ditetapkan, mis. volume radio atau kecerahan lampu. Faktanya, sakelar peredup untuk lampu rumah Anda adalah sejenis trimpot.

Proyek 7 - Lampu Berdenyut

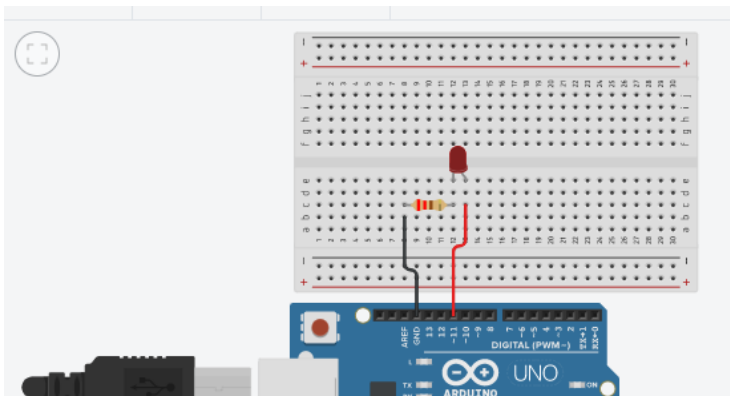
Sekarang kita akan mempelajari lebih lanjut metode yang lebih canggih untuk mengendalikan LED. Sejauh ini kami hanya menyalakan atau mematikan LED. Bagaimana kalau bisa menyesuaikan kecerahannya juga? Bisakah kita melakukannya dengan Arduino?

Ya kita bisa. Saatnya kembali ke dasar..

Apa yang Anda butuhkan

| | |
|-----------------------------|---|
| Led Merah - Nyala Merah 3mm |  |
| Resistor 220Ω |  |

Membuat Rangkaian



Masukkan kodenya.

Ikuti program sederhana ini.

```
// Proyek 7 - Lampu Berdenyut
int ledPin = 11;
float sinVal;
int ledVal;

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

void loop() {
  for (int x = 0; x < 180; x++) {
    // convert degrees to radians
    // then obtain sin value
    sinVal = (sin(x * (3.1412 / 180)));
    ledVal = int(sinVal * 255);
    analogWrite(ledPin, ledVal);
    delay(25);
  }
}
```

Verifikasi dan unggah. Anda sekarang akan melihat LED Anda berdenyut dan mati dengan mantap. Alih-alih keadaan hidup / mati yang sederhana, kami sekarang menyesuaikan kecerahannya. Mari kita cari tahu cara kerjanya.

Proyek 7 – Ikhtisar Kode

Kode untuk proyek ini sangat sederhana, tetapi membutuhkan penjelasan.

```
// Project 7 - Pulsating lamp
int ledPin = 11;
float sinVal;
int ledVal;

void setup() {
```

```

    pinMode(ledPin, OUTPUT);
}

void loop() {
    for (int x = 0; x < 180; x++) {
        // convert degrees to radians
        // then obtain sin value
        sinVal = (sin(x * (3.1412 / 180)));
        ledVal = int(sinVal * 255);
        analogWrite(ledPin, ledVal);
        delay(25);
    }
}

```

Pertama-tama kami menyiapkan variabel untuk Pin LED, sebuah float (tipe data pecahan) untuk nilai gelombang sinus dan ledVal yang akan menahan nilai integer untuk dikirim ke Pin 11.

Konsepnya di sini adalah bahwa kami membuat gelombang sinus dan memiliki kecerahan LED mengikuti jalur gelombang itu. Inilah yang membuat cahaya berdenyut dengan cara itu berubah memudar menjadi kecerahan penuh dan kembali turun.

Kami menggunakan fungsi sin (), yang merupakan fungsi matematika untuk menghitung sinus suatu sudut. Kita perlu memberikan fungsi derajat dalam radian. Kami memiliki loop for yang berjalan dari 0 hingga 179, kami tidak ingin melewati setengahnya karena ini akan membawa kami ke nilai negatif dan nilai kecerahan yang perlu kami masukkan ke Pin 11 harus dari 0 hingga 255 saja.

Fungsi sin () mengharuskan sudut dalam radian dan bukan derajat sehingga persamaan $x * (3.1412 / 180)$ akan mengubah sudut derajat menjadi radian. Kami kemudian mentransfer hasilnya ke ledVal, mengalikannya dengan 255 untuk memberi kami nilai. Hasil dari fungsi sin () akan menjadi angka antara -1 dan 1 jadi kita perlu mengalikannya dengan 255 untuk memberi kita kecerahan

maksimum. Kita 'cast' nilai titik mengambang `sinVal` ke dalam bilangan bulat dengan menggunakan `int ()` dalam pernyataan

```
ledVal = int(sinVal * 255);
```

Kemudian kami mengirimkan nilai itu ke Pin Digital 11 menggunakan pernyataan

```
analogWrite(ledPin, ledVal);
```

Tapi, bagaimana kita bisa mengirim nilai analog ke pin digital? Nah, jika kita melihat Arduino kita dan melihat Pin Digital Anda dapat melihat bahwa 6 dari pin tersebut (3, 5, 6, 9, 10 & 11) memiliki tulisan PWM di sebelahnya. Pin-pin tersebut berbeda dari pin digital yang tersisa karena mereka dapat mengirimkan sinyal PWM.



PWM adalah singkatan dari Pulse Width Modulation. PWM adalah teknik untuk mendapatkan hasil analog dari sarana digital. Pada pin ini, Arduino mengirimkan gelombang persegi dengan mengaktifkan dan menonaktifkan pin dengan sangat cepat. Pola ON / OFF dapat mensimulasikan tegangan yang bervariasi antara 0 dan 5v. Ini dilakukan dengan mengubah jumlah waktu di mana output tetap tinggi (hidup) versus mati (rendah). Durasi tepat waktu dikenal sebagai 'Pulse Width'.

Nanti kita akan meninjau kembali PWM karena kita dapat menggunakannya untuk membuat nada yang dapat didengar menggunakan suara piezo.

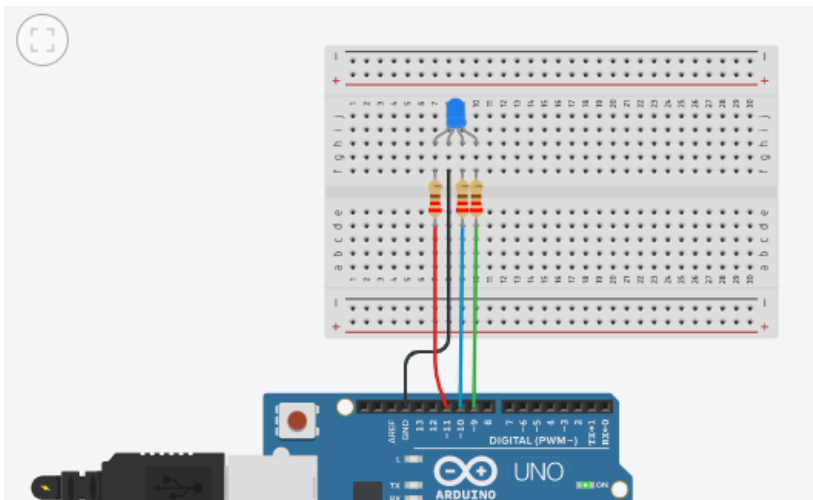
Proyek 8 – LED RGB

Dalam proyek terakhir kami melihat bahwa kami dapat menyesuaikan kecerahan LED menggunakan kemampuan PWM dari chip Atmega. Kami sekarang akan memanfaatkan kemampuan ini dengan menggunakan LED merah, hijau, dan biru dan dengan mencampurkan warnanya untuk membuat warna apa pun yang kami inginkan.

Apa yang Anda butuhkan

| | |
|---------------------------|---|
| LED RGB Katoda |  |
| Resistor 220 Ω x 3 |  |

Membuat Rangkaian



Ambil selembar kertas seukuran A5, gulung menjadi silinder lalu rekatkan agar tetap seperti itu. Kemudian tempatkan silinder di atas LED RGB.

Masukkan kode

```
// Proyek 8 - LED RGB
float RGB1[3];
float RGB2[3];
float INC[3];

int red, green, blue;

int RedPin = 11;
int GreenPin = 10;
int BluePin = 9;

void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));

  RGB1[0] = 0;
  RGB1[1] = 0;
  RGB1[2] = 0;

  RGB2[0] = random(256);
  RGB2[1] = random(256);
  RGB2[2] = random(256);
}

void loop()
{
  randomSeed(analogRead(0));

  for (int x = 0; x < 3; x++) {
    INC[x] = (RGB1[x] - RGB2[x]) / 256;

    for (int x = 0; x < 256; x++) {
      red = int(RGB1[0]);
      green = int(RGB1[1]);
      blue = int(RGB1[2]);

      analogWrite (RedPin, red);
      analogWrite (GreenPin, green);
```

```

    analogWrite (BluePin, blue);
    delay(100);

    RGB1[0] -= INC[0];
    RGB1[1] -= INC[1];
    RGB1[2] -= INC[2];
}

for (int x = 0; x < 3; x++) {
    RGB2[x] = random(556) - 300;
    RGB2[x] = constrain(RGB2[x], 0, 255);
    delay(1000);
}
}

```

Saat Anda menjalankan ini, Anda akan melihat warna perlahan berubah. Anda baru saja membuat lampu RGBAnda sendiri.

Proyek 8 - Ikhtisar Kode

LED yang membentuk lampu RGB berwarna merah, hijau, dan biru. Dengan cara yang sama seperti monitor komputer Anda terdiri dari titik-titik kecil merah, hijau dan biru (RGB), peta dapat menghasilkan warna yang berbeda dengan menyesuaikan kecerahan masing-masing dari 3 LED sedemikian rupa untuk memberi kita nilai RGB yang berbeda. .

Nilai RGB 255, 0, 0 akan memberi kita warna merah murni. Nilai 0, 255, 0 akan menghasilkan hijau murni dan 0, 0, 255 biru murni. Dengan mencampur ini kita bisa mendapatkan warna apapun yang kita suka dengan Ini adalah model warna aditif. Jika Anda baru saja menyalakan atau mematikan LED (mis.Tidak memiliki kecerahan berbeda), Anda masih akan mendapatkan warna berbeda seperti pada tabel ini.

| Red | Green | Blue | Colour |
|-----|-------|------|--------|
| 255 | 0 | 0 | Red |

| | | | |
|-----|-----|-----|---------|
| 0 | 255 | 0 | Green |
| 0 | 0 | 255 | Blue |
| 255 | 255 | 0 | Yellow |
| 0 | 255 | 255 | Cyan |
| 255 | 0 | 255 | Magenta |
| 255 | 255 | 255 | White |

Dengan menyesuaikan kecerahan menggunakan PWM, kita juga bisa mendapatkan warna lain di antaranya. Dengan menempatkan LED berdekatan dan dengan mencampurkan nilainya, spektrum cahaya dari 3 warna yang ditambahkan menjadi satu warna. Dengan menyebarkan cahaya dengan silinder kertas kami, kami memastikan warna tercampur dengan baik. LED dapat ditempatkan di objek apapun yang akan meredakan cahaya atau Anda dapat memantulkan cahaya dari diffuser reflektif. Coba letakkan lampu di dalam bola pingpong atau botol plastik putih kecil (semakin tipis plastiknya semakin baik). Rentang total warna yang bisa kita dapatkan menggunakan PWM dengan rentang 0 hingga 255 adalah 16.777, 216 warna (256x256x256) yang jauh lebih banyak dari yang kita butuhkan. Di dalam kode, kita mulai dengan mendeklarasikan beberapa array titik mengambang dan juga beberapa variabel integer yang akan menyimpan nilai RGB serta nilai kenaikan.

```
float RGB1[3];
float RGB2[3];
float INC[3];

int red, green, blue;
```

Dalam fungsi pengaturan yang kami miliki

```
randomSeed(analogRead(0));
```

Perintah `randomSeed` digunakan untuk membuat nomor acak (sebenarnya pseudo-random). Chip komputer tidak dapat menghasilkan angka yang benar-benar acak sehingga mereka cenderung melihat data di bagian memorinya yang mungkin berbeda atau melihat tabel dengan nilai yang berbeda dan menggunakannya sebagai angka acak semu. Dengan mengatur 'seed', Anda dapat memberitahu komputer di mana dalam memori atau di tabel itu untuk mulai menghitung. Dalam hal ini nilai yang kami berikan ke `randomSeed` adalah nilai yang dibaca dari Analog Pin 0. Karena kami tidak memiliki apa pun yang terhubung ke Pin Analog 0 semua yang akan kita baca adalah angka acak yang dibuat oleh noise analog.

Setelah kita menetapkan 'seed' untuk nomor acak kita, kita dapat membuatnya menggunakan fungsi `random()`. Kami kemudian memiliki dua set nilai RGB yang disimpan dalam array 3 elemen. RGB1 adalah nilai RGB yang kita inginkan untuk memulai lampu (dalam hal ini semua nol atau mati).

```
RGB1[0] = 0;  
RGB1[1] = 0;  
RGB1[2] = 0;
```

Kemudian larik RGB2 adalah sekumpulan nilai RGB acak yang kita inginkan untuk transisi lampu,

```
RGB2[0] = random(256);  
RGB2[1] = random(256);  
RGB2[2] = random(256);
```

Dalam hal ini kami telah mengaturnya ke nomor acak yang ditetapkan secara acak (256) yang akan memberikan angka antara 0 dan 255 inklusif (karena nomor akan selalu berkisar dari nol ke atas).

Jika Anda meneruskan satu angka ke fungsi random () maka itu akan mengembalikan nilai antara 0 dan 1 kurang dari angka, mis. random (1000) akan mengembalikan angka antara 0 dan 999. Jika Anda memberikan dua angka sebagai parameternya maka itu akan mengembalikan angka acak antara angka yang lebih rendah termasuk dan angka maksimum (-1). Misalnya. random (10,100) akan mengembalikan angka acak antara 10 dan 99.

Dalam loop program utama kita pertama-tama melihat nilai RGB awal dan akhir dan mencari tahu nilai apa yang dibutuhkan sebagai peningkatan untuk maju dari satu nilai ke nilai lainnya di 256 langkah (karena nilai PWM hanya boleh antara 0 dan 255). Kami melakukan ini dengan

```
for (int x = 0; x < 3; x++) {  
    INC[x] = (RGB1[x] - RGB2[x]) / 256;}
```

Ini untuk loop menetapkan nilai Increment untuk saluran R, G dan B dengan mengerjakan perbedaan antara dua nilai kecerahan dan membaginya dengan 256.

Kami kemudian memiliki loop for yang lain

```
for (int x = 0; x < 256; x++) {  
    red = int(RGB1[0]);  
    green = int(RGB1[1]);  
    blue = int(RGB1[2]);  
  
    analogWrite (RedPin, red);  
    analogWrite (GreenPin, green);  
    analogWrite (BluePin, blue);  
    delay(100);  
  
    RGB1[0] -= INC[0];  
    RGB1[1] -= INC[1];  
    RGB1[2] -= INC[2];  
}
```

dan ini menyetel nilai merah, hijau, dan biru ke nilai dalam larik RGB1, menulis nilai tersebut ke pin 9, 10, dan 11, lalu mengurangi nilai kenaikan lalu mengulangi proses ini 256 kali untuk memudar perlahan dari satu warna acak ke warna berikutnya . Penundaan 100 ms di antara setiap langkah memastikan perkembangan yang lambat dan stabil. Anda tentu saja dapat menyesuaikan nilai ini jika Anda menginginkannya lebih lambat atau lebih cepat atau Anda dapat menambahkan trimpot untuk memungkinkan pengguna mengatur kecepatan.

Setelah kita mengambil 256 langkah lambat dari satu warna acak ke warna berikutnya, larik RGB1 akan memiliki nilai yang sama (hampir) seperti larik RGB2. Kita sekarang perlu memutuskan satu set 3 nilai acak yang siap untuk waktu berikutnya. Kami melakukan ini dengan for loop lainnya

```
for (int x = 0; x < 3; x++) {  
    RGB2[x] = random(556) - 300;  
    RGB2[x] = constrain(RGB2[x], 0, 255);  
    delay(1000);  
}
```

Nomor acak dipilih dengan memilih nomor acak antara 0 dan 556 (256 + 300) dan kemudian dikurangi 300. Alasan kami melakukannya adalah untuk mencoba dan memaksa warna primer dari waktu ke waktu untuk memastikan kami tidak selalu hanya mendapatkan warna pastel. Kami memiliki 300 peluang dari 556 untuk mendapatkan angka negatif dan oleh karena itu memaksa bias terhadap satu atau lebih dari dua saluran warna lainnya. Perintah selanjutnya memastikan bahwa angka yang dikirim ke pin PWM tidak negatif dengan menggunakan fungsi constrain ().

Fungsi kendala membutuhkan 3 parameter - x, a dan b seperti dalam kendala (x, a, b) di mana x adalah angka yang ingin kita batasi, a adalah ujung bawah rentang dan b adalah ujung yang lebih tinggi. Jadi, fungsi kendala melihat nilai x dan memastikannya berada dalam

rentang a hingga b. Jika lebih rendah dari a maka set ke a, jika lebih tinggi dari b set ke b. Dalam kasus kami, kami memastikan bahwa jumlahnya antara 0 dan 255 yang merupakan rentang atau keluaran PWM kami.

Karena kami menggunakan random (556) -300 untuk nilai RGB kami, beberapa dari nilai tersebut akan lebih rendah dari nol dan fungsi kendala memastikan bahwa nilai yang dikirim ke PWM tidak lebih rendah dari nol.

Memaksakan bias terhadap satu atau lebih dari dua saluran lainnya memastikan nuansa warna yang lebih hidup dan tidak terlalu pastel dan juga memastikan bahwa dari waktu ke waktu satu atau beberapa saluran dimatikan sepenuhnya sehingga memberikan perubahan cahaya (atau RGB) yang lebih menarik.

Proyek 9 - LED Efek Kebakaran

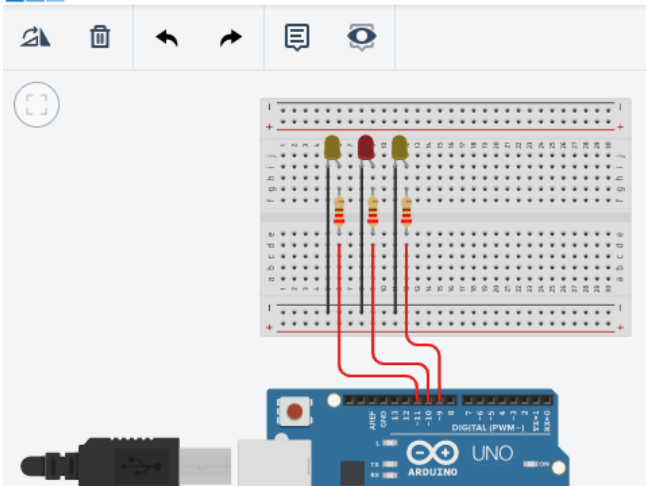
Proyek 9 akan menggunakan LED dan efek cahaya acak berkedip-kedip, menggunakan PWM lagi, untuk menciptakan kembali efek api yang berkedip-kedip.

Apa yang Anda butuhkan

| | |
|---------------------------------------|---|
| Led Merah - Nyala Merah 3mm (1 pcs) |  |
| Led Kuning - Nyala Kuning 3mm (2 pcs) |  |
| Resistor 220Ω (3 pcs) |  |

Membuat Rangkaian

Sekarang, pertama-tama pastikan Arduino Anda dimatikan. Anda dapat melakukan ini baik dengan mencabut kabel USB atau dengan mencabut Jumper Pemilih Daya pada papan Arduino. Kemudian hubungkan semuanya seperti ini:



Ketika Anda senang semuanya telah terhubung dengan benar, nyalakan Arduino Anda dan hubungkan kabel USB.

Masukkan kode

Sekarang, buka Arduino IDE dan ketik kode berikut:

```
// Proyek 9 - LED Efek Kebakaran
int ledPin1 = 9;
int ledPin2 = 10;
int ledPin3 = 11;

void setup()
{
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
}

void loop()
{
  analogWrite(ledPin1, random(120) + 135);
```

```
    analogWrite(ledPin2, random(120) + 135);  
    analogWrite(ledPin3, random(120) + 135);  
    delay(random(100));  
}
```

Sekarang tekan tombol Verify / Compile di bagian atas IDE untuk memastikan tidak ada kesalahan dalam kode Anda. Jika ini berhasil, Anda sekarang dapat mengklik tombol Unggah untuk mengunggah kode ke Arduino Anda.

Jika Anda telah melakukan semuanya dengan benar, Anda sekarang akan melihat LED berkedip-kedip secara acak untuk mensimulasikan efek kebakaran atau kebakaran.

Sekarang mari kita lihat kode dan perangkat kerasnya dan temukan bagaimana keduanya bekerja..

Proyek 9 – Ikhtisar Kode

Jadi mari kita lihat kode untuk proyek ini. Pertama kita mendeklarasikan dan menginisialisasi beberapa variabel integer yang akan menyimpan nilai untuk Pin Digital yang akan kita hubungkan dengan LED kita.

```
int ledPin1 = 9;  
int ledPin2 = 10;  
int ledPin3 = 11;
```

Kita kemudian mengaturnya menjadi keluaran.

```
pinMode(ledPin1, OUTPUT);  
pinMode(ledPin2, OUTPUT);  
pinMode(ledPin3, OUTPUT);
```

Loop program utama kemudian mengirimkan nilai acak antara 0 dan 120, dan kemudian menambahkan 135 padanya untuk mendapatkan kecerahan LED penuh, ke pin PWM 9, 10 dan 11.

```
analogWrite(ledPin1, random(120) + 135);  
analogWrite(ledPin2, random(120) + 135);  
analogWrite(ledPin3, random(120) + 135);
```

Lalu akhirnya kami memiliki penundaan acak antara pada dan 100 ms.

```
delay(random(100));
```

Loop utama kemudian mulai lagi menyebabkan efek cahaya berkedip yang Anda lihat.

Pantulkan cahaya dari kartu putih atau cermin ke dinding Anda dan Anda akan melihat efek nyala api yang sangat realistis.

Karena perangkat kerasnya sederhana dan kita harus memahaminya sekarang kita akan langsung masuk ke Proyek 10.

Proyek 10 – LED RGB Terkendali Serial

Kami sekarang akan menggunakan sirkuit yang sama seperti di Proyek 8, tetapi sekarang akan mempelajari dunia komunikasi serial dan kendalikan lampu kami dengan mengirimkan perintah dari PC ke Arduino menggunakan Serial Monitor di Arduino IDE.

Proyek ini juga memperkenalkan bagaimana kita memanipulasi string teks. Jadi biarkan perangkat keras diatur sama seperti sebelumnya dan masukkan kode baru.

Masukkan kode

```
// Project 10 - Serial controlled RGB Lamp

char buffer[18];
int red, green, blue;

int RedPin = 11;
int GreenPin = 10;
int BluePin = 9;

void setup()
{
    Serial.begin(9600);
    Serial.flush();
    pinMode(RedPin, OUTPUT);
    pinMode(GreenPin, OUTPUT);
    pinMode(BluePin, OUTPUT);
}

void loop()
{
    if (Serial.available() > 0) {
        int index = 0;
        delay(100); // let the buffer fill up
        int numChar = Serial.available();
        if (numChar > 15) {
            numChar = 15;
        }
        while (numChar-- > 0) {
```

```

        buffer[index++] = Serial.read();
    }
    splitString(buffer);
}
}

void splitString(char* data) {
    Serial.print("Data entered: ");
    Serial.println(data);
    char* parameter;
    parameter = strtok (data, " ,");
    while (parameter != NULL) {
        setLED(parameter);
        parameter = strtok (NULL, " ,");
    }

    // Clear the text and serial buffers
    for (int x = 0; x < 16; x++) {
        buffer[x] = '\0';
    }
    Serial.flush();
}

void setLED(char* data) {
    if ((data[0] == 'r') || (data[0] == 'R')) {
        int Ans = strtol(data + 1, NULL, 10);
        Ans = constrain(Ans, 0, 255);
        analogWrite(RedPin, Ans);
        Serial.print("Red is set to: ");
        Serial.println(Ans);
    }
    if ((data[0] == 'g') || (data[0] == 'G')) {
        int Ans = strtol(data + 1, NULL, 10);
        Ans = constrain(Ans, 0, 255);
        analogWrite(GreenPin, Ans);
        Serial.print("Green is set to: ");
        Serial.println(Ans);
    }
    if ((data[0] == 'b') || (data[0] == 'B')) {
        int Ans = strtol(data + 1, NULL, 10);
        Ans = constrain(Ans, 0, 255);
        analogWrite(BluePin, Ans);
        Serial.print("Blue is set to: ");
        Serial.println(Ans);
    }
}
}

```

Setelah Anda memverifikasi kodenya, unggah ke Arduino Anda. Sekarang ketika Anda mengunggah program sepertinya tidak ada yang terjadi. Ini karena program sedang menunggu masukan Anda. Mulai Monitor Serial dengan mengklik ikonnya di bilah tugas Arduino IDE.

Di jendela teks Serial Monitor, Anda sekarang dapat memasukkan nilai R, G, dan B untuk masing-masing 3 LED secara manual dan LED akan berubah menjadi warna yang Anda masukkan.

Misalnya. Jika Anda memasukkan R255, LED Merah akan ditampilkan dengan kecerahan penuh.

Jika Anda memasukkan R255, G255, maka LED merah dan hijau akan ditampilkan dengan kecerahan penuh.

Sekarang masukkan R127, G100, B255 dan Anda akan mendapatkan warna keunguan yang bagus.

Jika Anda mengetik, r0, g0, b0 semua LED akan mati.

Teks input dirancang untuk menerima R, G dan B huruf kecil atau besar dan kemudian nilai dari 0 hingga 255. Nilai apa pun di atas 255 akan diturunkan ke maksimum 255. Anda dapat memasukkan koma atau spasi di antara parameter dan Anda dapat memasukkan 1, 2 atau 3 nilai LED pada satu waktu. Misalnya.

```
r255  b100  
r127  b127  g127  
G255, B0  
B127, R0, G255
```

DII

Proyek 10 - Ikhtisar Kode

Proyek ini memperkenalkan sejumlah besar konsep baru, termasuk komunikasi serial, pointer dan manipulasi string. Jadi, ini akan membutuhkan banyak penjelasan.

Pertama kita menyiapkan array char (karakter) untuk menampung string teks kita. Kami telah membuatnya 18 karakter, yang mana lebih panjang dari maksimum 16, kami akan mengizinkan untuk memastikan kami tidak mendapatkan kesalahan "buffer overflow".

```
char buffer[18];
```

Kami kemudian mengatur bilangan bulat untuk menahan nilai merah, hijau dan biru serta nilai untuk pin digital.

```
int red, green, blue;
```

```
int RedPin = 11;
```

```
int GreenPin = 10;
```

```
int BluePin = 9;
```

Dalam fungsi setup, kita atur 3 pin digital menjadi output. Tapi, sebelumnya kita punya perintah Serial.begin.

```
void setup()
{
    Serial.begin(9600);
    Serial.flush();
    pinMode(RedPin, OUTPUT);
    pinMode(GreenPin, OUTPUT);
    pinMode(BluePin, OUTPUT);
}
```

Serial.begin memberi tahu Arduino untuk memulai komunikasi serial dan nomor di dalam tanda kurung, dalam hal ini 9600, menetapkan baud rate (karakter per detik) yang akan digunakan untuk berkomunikasi dengan jalur serial.

Perintah `Serial.flush` akan menghapus semua karakter yang kebetulan ada di baris serial sehingga kosong dan siap untuk input / output. Jalur komunikasi serial hanyalah cara bagi Arduino untuk berkomunikasi dengan dunia luar, dalam hal ini ke dan dari PC dan Monitor Serial Arduino IDE.

Di loop utama kami memiliki pernyataan `if`. Kondisi yang sedang diperiksa adalah

```
if (Serial.available() > 0) {
```

Perintah `Serial.available` memeriksa untuk melihat apakah ada karakter yang telah dikirim ke baris serial. Jika ada karakter yang telah diterima maka kondisi terpenuhi dan kode dalam blok kode pernyataan `if` sekarang dijalankan.

```
    if (Serial.available() > 0) {  
        int index = 0;  
        delay(100); // let the buffer fill up  
        int numChar = Serial.available();  
        if (numChar > 15) {  
            numChar = 15;  
        }  
        while (numChar-- > 0) {  
            buffer[index++] = Serial.read();  
        }  
        splitString(buffer);  
    }
```

Integer yang disebut `index` dideklarasikan dan diinisialisasi sebagai nol. Integer ini akan menahan posisi pointer ke karakter dalam array karakter.

Kami kemudian menetapkan penundaan sebesar 100/ `delay(100)`. Tujuannya adalah untuk memastikan bahwa buffer serial (tempat di memori tempat penyimpanan data serial yang diterima sebelum diproses) sudah penuh sebelum kami melanjutkan dan memproses data. Jika kita tidak melakukannya, ada kemungkinan fungsi tersebut

akan mengeksekusi dan mulai memproses string teks, sebelum kita menerima semua datanya. Jalur komunikasi serial sangat lambat dibandingkan dengan kecepatan eksekusi kode lainnya. Saat Anda mengirim string karakter, fungsi Serial.available akan segera memiliki nilai lebih tinggi dari nol dan fungsi if akan mulai dijalankan. Jika kita tidak memiliki pernyataan delay (100) di sana, ia dapat mulai mengeksekusi kode dalam pernyataan if sebelum semua string teks diterima dan data serial mungkin hanya beberapa karakter pertama dari baris teks yang dimasukkan.

Setelah kami menunggu 100ms untuk buffer serial terisi dengan data yang dikirim, kami kemudian mendeklarasikan dan menginisialisasi integer numChar menjadi jumlah karakter dalam string teks.

Misalnya. Jika kami mengirimkan teks ini di Serial Monitor:

```
R255, G255, B255
```

Maka nilai numChar akan menjadi 17. Ini adalah 17 dan bukan 16 karena pada akhir setiap baris teks ada karakter tak terlihat yang disebut karakter NULL. Ini adalah simbol 'tidak ada apa-apa' dan hanya memberitahu Arduino bahwa akhir baris teks telah tercapai.

Pernyataan if berikutnya memeriksa apakah nilai numChar lebih besar dari 15 atau tidak dan jika demikian maka set itu menjadi 15.

Ini memastikan bahwa kita tidak **overflow** `char buffer[18]` ;

Setelah ini muncul perintah sementara. Ini adalah sesuatu yang belum kami temui sebelumnya, jadi izinkan saya menjelaskan.

Kami telah menggunakan perulangan for, yang akan mengulang beberapa kali. Pernyataan while juga merupakan perulangan, tetapi yang dijalankan hanya jika kondisi benar.

Sintaksnya adalah

```
while (ekspresi) {  
    // pernyataan  
}
```

Dalam kode kami, loop sementara adalah

```
while (numChar--) {  
    buffer[index++] = Serial.read();  
}
```

Kondisi yang diperiksa hanyalah numChar, jadi dengan kata lain ia memeriksa bahwa nilai yang disimpan dalam integer numChar bukan nol. numChar memiliki - setelahnya. Inilah yang dikenal sebagai penurunan pasca. Dengan kata lain, nilainya dikurangi SETELAH digunakan. Jika kita telah menggunakan --numChar, nilai dalam numChar akan dikurangi (ada satu yang dikurangi darinya) sebelum dievaluasi. Dalam kasus kami, while loop memeriksa nilai numChar dan kemudian mengurangi satu darinya. Jika nilai numChar tidak nol sebelum penurunan, itu kemudian menjalankan kode dalam blok kodenya.

numChar diatur ke panjang string teks yang telah kita masukkan ke jendela Serial Monitor. Jadi, kode di dalam while loop akan dieksekusi berkali-kali.

Kode di dalam loop while adalah

```
buffer[index++] = Serial.read();
```

Yang menetapkan setiap elemen array penyangga ke setiap karakter yang dibaca dari baris Serial. Dengan kata lain, itu mengisi array penyangga dengan huruf-huruf yang telah kita masukkan ke dalam jendela teks Serial Monitor.

Perintah Serial.read () membaca data serial yang masuk, satu byte pada satu waktu.

Jadi sekarang array karakter kita telah diisi dengan karakter yang kita masukkan dalam Monitor Serial, loop sementara akan berakhir begitu numChar mencapai nol (yaitu Panjang string).

Setelah loop sementara yang kita miliki

```
splitString (buffer);
```

Yang merupakan panggilan ke salah satu dari dua fungsi yang telah kita buat dan disebut splitString (). Fungsinya terlihat seperti ini:

```
void splitString(char* data) {
    Serial.print("Data entered: ");
    Serial.println(data);
    char* parameter;
    parameter = strtok (data, " ,");
    while (parameter != NULL) {
        setLED(parameter);
        parameter = strtok (NULL, " ,");
    }

    // Clear the text and serial buffers
    for (int x = 0; x < 16; x++) {
        buffer[x] = '\0';
    }
    Serial.flush();
}
```

Kita dapat melihat bahwa fungsi tersebut tidak mengembalikan data, oleh karena itu tipe datanya telah disetel ke kosong. Kami melewati fungsi satu parameter dan itu adalah tipe data char yang kami sebut data. Namun, dalam bahasa pemrograman C dan C ++ Anda tidak diizinkan mengirim larik karakter ke suatu fungsi. Kami mengatasinya dengan menggunakan penunjuk. Kita tahu bahwa kita telah menggunakan penunjuk karena tanda bintang '*' telah ditambahkan ke nama variabel * data. Pointer adalah subjek yang cukup mahir dalam bahasa C jadi kami tidak akan membahas terlalu banyak detail tentangnya. Yang perlu Anda ketahui untuk saat ini adalah dengan mendeklarasikan 'data' sebagai penunjuk, itu hanyalah sebuah variabel yang menunjuk ke variabel lain.

Anda dapat mengarahkannya ke alamat yang menyimpan variabel dalam memori dengan menggunakan simbol &, atau dalam kasus kami, ke nilai yang disimpan di alamat memori tersebut menggunakan simbol *. Kami telah menggunakannya untuk 'mengecek' sistem, karena kami tidak diizinkan untuk mengirim larik karakter ke suatu fungsi. Namun kami diizinkan untuk mengirim pointer ke array karakter ke fungsi kami. Jadi, kita telah mendeklarasikan variabel tipe data Char dan menyebutnya data, tetapi simbol * sebelumnya berarti bahwa itu adalah 'menunjuk ke' nilai yang disimpan dalam variabel 'buffer'.

Ketika kami memanggil `splitString`, kami mengirimkannya konten 'buffer' (sebenarnya adalah penunjuk ke sana seperti yang kita lihat di atas).

```
splitString(buffer);
```

Jadi kita telah memanggil fungsi tersebut dan meneruskannya ke seluruh konten array karakter `buffer`.

Perintah pertama adalah

```
Serial.print("Data entered: ");
```

dan ini adalah cara kami mengirim data kembali dari Arduino ke PC. Dalam hal ini perintah cetak mengirimkan apa pun yang ada di dalam tanda kurung ke PC, melalui kabel USB, di mana kita dapat membacanya di jendela Serial Monitor. Dalam hal ini kami telah mengirimkan kata-kata "Data dimasukkan:". Teks harus diapit tanda kutip "". Baris berikutnya mirip

```
Serial.println(data);
```

dan lagi-lagi kami telah mengirimkan data kembali ke PC, kali ini kami mengirim variabel char yang disebut `data`. Variabel tipe Char yang kita panggil 'data' adalah salinan dari isi array karakter 'buffer' yang kita berikan ke fungsi. Jadi, jika string teks kita masuk dulu

Kemudian

```
Serial.println(data);
```

Command akan mengirim string teks itu kembali ke PC dan mencetaknya di jendela Serial Monitor (pastikan Anda telah mengaktifkan jendela Serial Monitor terlebih dahulu).

Kali ini perintah print telah di akhir untuk membuatnya menjadi println. Ini berarti 'print' dengan 'linefeed'.

Saat kami mencetak menggunakan perintah print, kursor (titik di mana simbol berikutnya akan muncul) tetap berada di akhir apa pun yang telah kami cetak. Ketika kita menggunakan perintah println, perintah linefeed dikeluarkan atau dengan kata lain teks dicetak dan kemudian kursor turun ke baris berikutnya.

```
Serial.print("Data entered: ");  
Serial.println(data);
```

Jika kita melihat pada dua perintah prints kita, yang pertama prints "Data entered:" dan kemudian kursor tetap berada di akhir teks itu. Perintah print berikutnya akan mencetak 'data', atau dengan kata lain isi dari array yang disebut 'buffer' dan kemudian mengeluarkan sebuah linefeed, atau menurunkan kursor ke baris berikutnya. Ini berarti bahwa jika kita mengeluarkan print atau println statement setelah ini, apa pun yang dicetak di jendela Serial Monitor akan muncul di baris berikutnya di bawah yang terakhir.

Kami kemudian membuat tipe data char baru yang disebut parameter

```
char* parameter;
```

dan karena kita akan menggunakan variabel ini untuk mengakses elemen dari array 'data' itu harus berjenis sama, oleh karena itu simbol

*. Anda tidak dapat meneruskan data dari satu variabel tipe data ke variabel lain karena data harus dikonversi terlebih dahulu. Variabel ini adalah contoh lain dari variabel yang memiliki 'elingkup lokal'. Ini hanya bisa 'terlihat' oleh kode dalam fungsi ini. Jika Anda mencoba mengakses variabel parameter di luar fungsi `splitString`, Anda akan mendapatkan error.

Kami kemudian menggunakan perintah `strtok`, yang merupakan perintah yang sangat berguna untuk memungkinkan kami memanipulasi string teks. `Strtok` mendapatkan namanya dari `String` dan `Token` karena tujuannya adalah untuk membagi string menggunakan token. Dalam kasus kami, token yang dicari adalah spasi atau koma. Ini digunakan untuk membagi string teks menjadi string yang lebih kecil.

Kami meneruskan array 'data' ke perintah `strtok` sebagai argumen pertama dan token (diapit dalam tanda kutip) sebagai argumen kedua. Karenanya `parameter = strtok (data, ",");`

Dan itu memisahkan string. Jadi kami menggunakannya untuk men-set 'parameter' menjadi bagian dari string hingga spasi atau koma.

Jadi, jika string teks kita adalah

```
R127 G56 B98
```

Kemudian setelah pernyataan ini nilai 'parameter' akan menjadi

```
R127
```

karena perintah `strtok` akan memisahkan string hingga kemunculan pertama dari spasi koma.

Setelah kita mengatur variabel 'parameter' ke bagian dari string teks yang ingin kita hapus (mis. Bit ke spasi pertama atau koma) kita kemudian masuk ke loop sementara yang kondisinya adalah parameter itu tidak kosong

(Yaitu Kami belum mencapai akhir string) menggunakan

```
while (parameter != NULL) {
```

Di dalam loop kita memanggil function setLED (parameter) kedua kita; Yang akan kita lihat nanti. Kemudian itu menetapkan variabel 'parameter' ke bagian string berikutnya hingga spasi atau koma berikutnya. Kami melakukan ini dengan meneruskan ke strtok parameter

```
parameter = strtok (NULL, " ,");
```

Ini memberitahu perintah strtok untuk melanjutkan di tempat terakhir tinggalkan.

Jadi ini seluruh bagian dari fungsinya

```
char* parameter;  
parameter = strtok (data, " ,");  
while (parameter != NULL) {  
    setLED(parameter);  
    parameter = strtok (NULL, " ,");  
}
```

cukup menghapus setiap bagian dari string teks yang dipisahkan oleh spasi atau koma dan mengirimkan bagian string tersebut ke fungsi berikutnya yang disebut setLED ().

Bagian terakhir dari fungsi ini hanya mengisi buffer array dengan karakter NULL, yang dilakukan dengan simbol / 0 dan kemudian membersihkan data Serial dari buffer Serial siap untuk set data berikutnya yang akan dimasukkan.

```
// Clear the text and serial buffers  
for (int x = 0; x < 16; x++) {  
    buffer[x] = '\0';  
}  
Serial.flush();
```


Fungsi setLED akan mengambil setiap bagian dari string teks dan mengatur LED yang sesuai ke warna yang telah kita pilih. Jadi, jika string teks yang kita masukkan adalah

G125 B55

Kemudian fungsi splitString () membaginya menjadi dua komponen terpisah

G125 B55

dan mengirim string teks yang dipersingkat itu ke fungsi setLED (), yang akan membacanya, memutuskan LED apa yang telah kita pilih dan mengaturnya ke nilai kecerahan yang sesuai.

Jadi mari kita lihat fungsi kedua yang disebut setLED ().

```
void setLED(char* data) {
    if ((data[0] == 'r') || (data[0] == 'R')) {
        int Ans = strtol(data + 1, NULL, 10);
        Ans = constrain(Ans, 0, 255);
        analogWrite(RedPin, Ans);
        Serial.print("Red is set to: ");
        Serial.println(Ans);
    }
    if ((data[0] == 'g') || (data[0] == 'G')) {
        int Ans = strtol(data + 1, NULL, 10);
        Ans = constrain(Ans, 0, 255);
        analogWrite(GreenPin, Ans);
        Serial.print("Green is set to: ");
        Serial.println(Ans);
    }
    if ((data[0] == 'b') || (data[0] == 'B')) {
        int Ans = strtol(data + 1, NULL, 10);
        Ans = constrain(Ans, 0, 255);
        analogWrite(BluePin, Ans);
        Serial.print("Blue is set to: ");
        Serial.println(Ans);
    }
}
```

Kita dapat melihat bahwa fungsi ini berisi 3 pernyataan if yang sangat mirip. Oleh karena itu kita akan melihat salah satunya karena 2 lainnya hampir identik.

```
if ((data[0] == 'r') || (data[0] == 'R')) {  
    int Ans = strtol(data + 1, NULL, 10);  
    Ans = constrain(Ans, 0, 255);  
    analogWrite(RedPin, Ans);  
    Serial.print("Red is set to: ");  
    Serial.println(Ans);  
}
```

Pernyataan if memeriksa bahwa karakter pertama dalam data string [0] adalah huruf r atau R (karakter huruf besar dan huruf kecil sama sekali berbeda sejauh menyangkut C. Kami menggunakan perintah logika OR yang simbolnya adalah || untuk memeriksa apakah huruf tersebut adalah r ATAU an R seperti yang akan dilakukan.

Jika itu adalah r atau R maka pernyataan if tahu bahwa kita ingin mengubah kecerahan LED Merah dan kode di dalamnya dijalankan. Pertama kita mendeklarasikan integer yang disebut Ans (yang memiliki cakupan lokal hanya untuk fungsi setLED) dan menggunakan perintah strtol (String to long integer) untuk mengubah karakter setelah huruf R menjadi integer. Perintah strtol mengambil 3 parameter dan ini adalah string yang kita berikan, penunjuk ke karakter setelah integer (yang tidak kita gunakan karena kita telah menghapus string menggunakan perintah strtok dan karenanya meneruskan karakter NULL) dan kemudian 'Base', yang dalam kasus kami adalah basis 10 karena kami menggunakan angka desimal normal (sebagai lawan dari biner, oktal atau heksadesimal yang masing-masing akan menjadi basis 2, 8 dan 16). Jadi dengan kata lain kita mendeklarasikan integer dan mengaturnya ke nilai string teks setelah huruf R (atau angka bit).

Selanjutnya kami menggunakan perintah kendala untuk memastikan bahwa Ans beralih dari 0 hingga 255 dan tidak lebih. Kami kemudian menjalankan perintah analogWrite ke pin merah dan mengirimkannya

nilai Ans. Kode kemudian mengirimkan “Merah diatur ke:” diikuti dengan nilai Ans kembali ke Serial Monitor. Dua pernyataan if lainnya persis sama tetapi untuk LED Hijau dan Biru.

Kami telah membahas banyak hal dan banyak konsep baru dalam proyek ini. Untuk memastikan Anda memahami dengan tepat apa yang terjadi dalam kode ini, saya akan mengatur kode proyek berdampingan dengan kode pseudo (bahasa komputer palsu yang pada dasarnya adalah bahasa komputer yang diterjemahkan ke dalam bahasa yang dapat dipahami manusia).

| Bahasa Pemrograman C. | Pseudo-Code |
|---|--|
| <pre>// Project 10 - Serial controlled RGB Lamp char buffer[18]; int red, green, blue; int RedPin = 11; int GreenPin = 10; int BluePin = 9;</pre> | <p>Komentar dengan nomor dan nama proyek</p> <p>Deklarasikan larik karakter yang terdiri dari 18 huruf Deklarasikan 3 bilangan bulat yang disebut merah, hijau dan biru Bilangan bulat yang pinnya akan digunakan untuk LED Merah " " Hijau " " Biru</p> |
| <pre>void setup() { Serial.begin(9600); Serial.flush(); pinMode(RedPin, OUTPUT); pinMode(GreenPin, OUTPUT); pinMode(BluePin, OUTPUT); }</pre> | <p>Fungsi pengaturan</p> <p>Setel komunikasi serial agar berjalan pada 9600 karakter per detik Hapus saluran serial Atur pin led merah menjadi pin keluaran Sama untuk warna hijau Dan biru</p> |
| <pre>void loop() { if (Serial.available() > 0) { int index = 0; delay(100); // let the buffer fill up int numChar = Serial.available(); if (numChar > 15) { numChar = 15;</pre> | <p>Loop program utama</p> <p>Jika data dikirim ke jalur serial ... Deklarasikan integer yang disebut index dan setel ke 0 Tunggu 100 milliseconds</p> <p>Setel numChar ke data yang masuk dari serial Jika numchar lebih dari 15 karakter ... Jadikan 15 dan tidak lebih</p> |

| | |
|--|---|
| <pre> } while (numChar--) { buffer[index++] = Serial.read(); } splitString(buffer); } } </pre> | <p>Meskipun numChar bukan nol (kurangi 1 darinya) Setel elemen [index] menjadi nilai yang dibaca (tambahkan 1) Panggil fungsi splitString dan kirimkan data dalam buffer</p> |
| <pre> void splitString(char* data) { Serial.print("Data entered: "); Serial.println(data); char* parameter; parameter = strtok (data, " ,"); while (parameter != NULL) { setLED(parameter); parameter = strtok (NULL, " ,"); } // Clear the text and serial buffers for (int x = 0; x < 16; x++) { buffer[x] = '\0'; } Serial.flush(); } </pre> | <p>Fungsi splitstring mereferensikan data buffer Cetak "Data dimasukkan:" Cetak nilai data dan kemudian tarik satu baris Deklarasikan parameter tipe data char Setel ke teks hingga spasi pertama atau koma Sementara konten parameter tidak kosong .. ! Panggil fungsi setLED Tetapkan parameter ke bagian berikutnya dari string teks</p> <p>Komentar lain Kami akan melakukan baris berikutnya 16 kali Setel setiap elemen buffer ke NULL (kosong)</p> <p>Hapus/ Flush komunikasi serial</p> |
| <pre> void setLED(char* data) { if ((data[0] == 'r') (data[0] == 'R')) { int Ans = strtol(data + 1, NULL, 10); Ans = constrain(Ans, 0, 255); analogWrite(RedPin, Ans); Serial.print("Red is set to: "); Serial.println(Ans); } if ((data[0] == 'g') (data[0] == 'G')) { int Ans = strtol(data + 1, NULL, 10); Ans = constrain(Ans, 0, 255); </pre> | <p>Fungsi yang disebut setLED melewati buffer Jika huruf pertama adalah r atau R ... Set integer Ans to number in next part of text Pastikan itu berada di antara 0 dan 255 Tuliskan nilai itu ke pin merah Cetak "Merah disetel ke:" Dan kemudian nilai Ans</p> <p>Jika huruf pertama adalah g atau G ... Atur bilangan bulat Ans ke angka di bagian teks selanjutnya Pastikan itu berada di antara 0 dan 255 Tuliskan nilai itu ke pin hijau</p> |

| | |
|---|--|
| <pre> analogWrite(GreenPin, Ans); Serial.print("Green is set to: "); Serial.println(Ans); } if ((data[0] == 'b') (data[0] == 'B')) { int Ans = strtol(data + 1, NULL, 10); Ans = constrain(Ans, 0, 255); analogWrite(BluePin, Ans); Serial.print("Blue is set to: "); Serial.println(Ans); } } </pre> | <p>Cetak "Hijau disetel ke:" Dan kemudian nilai Ans</p> <p>Jika huruf pertama adalah b atau B ... Atur bilangan bulat Ans ke angka di bagian teks selanjutnya Pastikan itu berada di antara 0 dan 255 Tuliskan nilai itu ke pin biru Cetak "Biru disetel ke:" Dan kemudian nilai Ans</p> |
|---|--|

Mudah-mudahan Anda dapat menggunakan 'pseudo-code' ini untuk memastikan Anda memahami dengan tepat apa yang terjadi dalam kode proyek ini.

Sekarang kita akan meninggalkan LED sebentar dan melihat bagaimana membuat suara dengan Arduino Anda.

Proyek 11 - Piezo Sounder Melody Player

Dalam proyek ini kita akan menggunakan cara super sederhana.

Saat Anda menjalankan kode ini, Arduino akan bermain sangat bagus, membawakan lagu dari 'Twinkle Twinkle Little Star'.

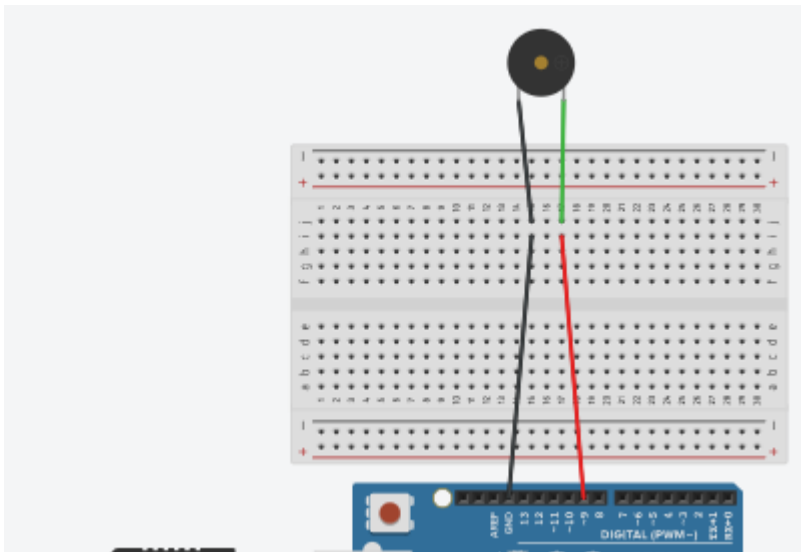
Apa yang Anda butuhkan

Universal passive buzzer electromagnetic impedance 16 ohms AC 2KHz

 digipart



Membuat Rangkaian



Masukkan kode

(courtesy of <http://www.arduino.cc/en/Tutorial/Melody>)

```
// Project 11 - Melody Player
int speakerPin = 9;
int length = 15; // the number of notes
char notes[] = "cgggaagffeeddc "; // a space represents
a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,
2, 4 };
int tempo = 300;
void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'
};
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136,
1014, 956 };
  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}

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

void loop() {
  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }
  }
}
```

```

    // pause between notes
    delay(tempo / 2);
  }
}

```

Proyek 11 - Ikhtisar Kode

Dalam proyek ini kami membuat suara menggunakan Passive buzzer. Untuk membunyikan Passive buzzer, kita harus memberikan tegangan sebentar dan mematikan lagi. Jadi untuk mendapatkan nada yang dapat kita dengar darinya, kita perlu membuatnya berbunyi klik berkali-kali dalam satu detik dengan cukup cepat sehingga menjadi nada yang dapat dikenali.

Program dimulai dengan menyiapkan variabel yang kita butuhkan. Kabel positif (merah) piezo sounders dipasang ke Pin 9.

```
int speakerPin = 9;
```

Lagu yang akan kita mainkan terdiri dari 15 nada.

```
int length = 15; // the number of notes
```

Not dari lagu disimpan dalam larik karakter sebagai string teks.

```
char notes[] = "ccggaagffeeddc "; // a space
represents a rest
```

Array lain, kali ini integer, diatur untuk menyimpan panjang setiap nada.

```
int beats[] = { 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
1, 2, 4 };
```

Dan akhirnya kami menetapkan tempo untuk lagu yang akan dimainkan,


```
int tempo = 300;
```

Selanjutnya Anda akan melihat bahwa kami mendeklarasikan dua fungsi sebelum fungsi `setup()` dan `loop()` kami. Tidak masalah jika kita meletakkan fungsi kita sendiri sebelum atau sesudah `setup()` dan `loop()`. Ketika program berjalan, kode di dalam dua fungsi ini tidak akan berjalan sebelum `setup()` berjalan karena kita belum memanggil fungsi tersebut.

Mari kita lihat fungsi `setup` dan `loop` sebelum kita melihat fungsi `playTone` dan `playNote`.

Semua yang terjadi di `setup()` adalah kita menetapkan pin speaker (9) sebagai output.

```
void setup() {  
  pinMode(speakerPin, OUTPUT);  
}
```

Dalam perulangan program utama kita memiliki pernyataan `if / else` di dalam perulangan `for`.

```
for (int i = 0; i < length; i++) {  
  if (notes[i] == ' ') {  
    delay(beats[i] * tempo); // rest  
  } else {  
    playNote(notes[i], beats[i] * tempo);  
  }  
  
  // pause between notes  
  delay(tempo / 2);  
}
```

Seperti yang Anda lihat, pernyataan `if` pertama memiliki kondisi seperti itu, bahwa elemen array `[i]` bahwa elemen tersebut berisi karakter spasi. `if (notes[i] == ' ') {`

Jika ini BENAR maka kode di dalamnya akan dieksekusi.

```
    delay(beats[i] * tempo); // rest
```

dan ini hanya menentukan nilai `beats[i] * tempo` dan menyebabkan penundaan selama itu menyebabkan not berhenti. Kami kemudian memiliki pernyataan lain.

```
else {  
    playNote(notes[i], beats[i] * tempo);
```

Setelah pernyataan `if` kita bisa memperpanjangnya dengan pernyataan lain. Pernyataan lain dilakukan jika kondisi dalam pernyataan `if` salah. Jadi misalnya. Katakanlah kita memiliki integer yang disebut `test` dan nilainya adalah 10 dan pernyataan `if / else` ini:

```
if (test == 10) {          digitalWrite(ledPin,  
HIGH)  
    } else {  
        digitalWrite(ledPin, LOW)    }
```

Kemudian jika ‘`test`’ memiliki nilai 10 (yang memang demikian) `ledPin` akan disetel ke TINGGI. Jika nilai `test` adalah selain 10, kode di dalam pernyataan lain akan dijalankan dan `ledPin` akan disetel ke LOW.

Pernyataan `else` memanggil fungsi yang disebut `playNote` dan meneruskan dua parameter. Parameter pertama adalah nilai `not [i]` dan yang kedua adalah nilai yang dihitung dari `beats[i] * tempo`.

```
    playNote(notes[i], beats[i] * tempo);
```

Setelah pernyataan `if / else` dilakukan, terjadi `delay` yang nilainya dihitung dengan membagi `tempo` dengan 2.

```
    delay(tempo / 2);
```

Sekarang mari kita lihat dua fungsi yang telah kita buat untuk proyek ini.

Fungsi pertama yang dipanggil dari loop program utama adalah `playNote`.

```
void playNote(char note, int duration) {
    char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'
};
    int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136,
1014, 956 };
    // play the tone corresponding to the note name
    for (int i = 0; i < 8; i++) {
        if (names[i] == note) {
            playTone(tones[i], duration);
        }
    }
}
```

Dua parameter telah diteruskan ke fungsi dan di dalam fungsi ini telah diberi nama `note` (karakter) dan `durasi` (integer).

Fungsi ini menyiapkan array variabel lokal dari tipe data `char` yang disebut `'names'`. Variabel ini memiliki cakupan lokal sehingga hanya dapat dilihat oleh fungsi ini dan bukan di luarnya.

Array ini menyimpan nama-nama not dari C tengah hingga C tinggi.

Kami kemudian membuat larik lain dari bilangan bulat tipe data dan larik ini menyimpan angka yang sesuai dengan frekuensi nada, dalam Kilohertz, dari setiap nada dalam array `names[]`.

```
int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136 ,
1014, 956 };
```

Setelah menyiapkan dua larik, ada perulangan `for` yang melihat melalui 8 catatan dalam array `names[]` dan membandingkannya dengan catatan yang dikirim ke fungsi.

```
for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
        playTone(tones[i], duration);
    }
}
```

```

    }
}

```

Nada yang dikirim ke fungsi ini adalah 'ccggaagffeeddc' jadi nada pertama akan menjadi tengah C. Perulangan for membandingkan catatan itu dengan catatan dalam array `names[]` dan jika ada yang cocok, memanggil fungsi kedua, dipanggil `playTone`, untuk memainkan nada yang sesuai menggunakan dalam array `tones[]` menggunakan panjang nada 'duration'. Fungsi kedua disebut `playTone`.

```

void playTone(int tone, int duration) {
    for (long i = 0; i < duration * 1000L; i += tone
* 2) {
        digitalWrite(speakerPin, HIGH);
        delayMicroseconds(tone);
        digitalWrite(speakerPin, LOW);
        delayMicroseconds(tone);
    }
}

```

Dua parameter diteruskan ke fungsi ini. Yang pertama adalah nada (dalam kilohertz) yang kita ingin agar speaker piezo mereproduksi dan yang kedua adalah durasi (dibuat dengan menghitung `beats[i] * tempo`. Fungsi ini memulai loop for

```

for (long i = 0; i < duration * 1000L; i += tone *
2)

```

Karena setiap loop for harus memiliki panjang yang berbeda untuk membuat setiap nada memiliki panjang yang sama (karena penundaan berbeda antara klik untuk menghasilkan frekuensi yang diinginkan) loop for akan berjalan ke 'duration' dikalikan 1000 dan kenaikan loop adalah nilai dari 'Tone' dikalikan dengan 2.

Di dalam loop for, kita cukup membuat pin yang terhubung ke speaker piezo menjadi tinggi, tunggu beberapa saat, lalu rendah, lalu tunggu beberapa saat lagi, lalu ulangi.

```
digitalWrite(speakerPin, HIGH);  
delayMicroseconds(tone);  
digitalWrite(speakerPin, LOW);  
delayMicroseconds(tone);
```

Klik berulang ini, dengan panjang yang berbeda dan dengan jeda yang berbeda (panjangnya hanya mikrodetik) di antara klik, membuat piezo menghasilkan nada dengan frekuensi yang berbeda-beda.

Proyek 11 – Ikhtisar Perangkat Keras

Satu-satunya perangkat keras yang digunakan dalam proyek ini adalah passive buzzer. Buzzer merupakan sebuah komponen elektronika yang masuk dalam keluarga transduser, yang dimana dapat mengubah sinyal listrik menjadi getaran suara. Nama lain dari komponen ini disebut dengan beeper.



Dalam kehidupan sehari – hari, umumnya digunakan untuk rangkaian alarm pada jam, bel rumah, perangkat peringatan bahaya, dan lain sebagainya.

Jenis – jenis yang sering ditemukan dipasaran yaitu tipe piezoelectric. Dikarenakan tipe ini memiliki kelebihan seperti harganya yang relatif murah, mudah diaplikasikan ke dalam rangkaian elektronika.

Terdapat 2 jenis yang terdapat dipasaran antara lain :

Passive buzzer yaitu yang tidak mempunyai suara sendiri, sehingga cocok untuk dipasangkan dengan arduino yang dapat diprogram tinggi rendah nadanya. Contoh dalam kehidupan sehari – hari yaitu speaker.

Active buzzer yaitu yang dapat berdiri sendiri atau standalone atau singkatnya sudah mempunyai suara tersendiri ketika diberikan catu daya.


Latihan :

1. Ubah not dan ketukan untuk membuat nada lain, seperti ‘Selamat Ulang tahun’ atau Garuda Pancasila’.
2. Menulis program untuk membuat nada naik turun dari passive buzzer, mirip dengan alarm mobil atau sirene polisi.

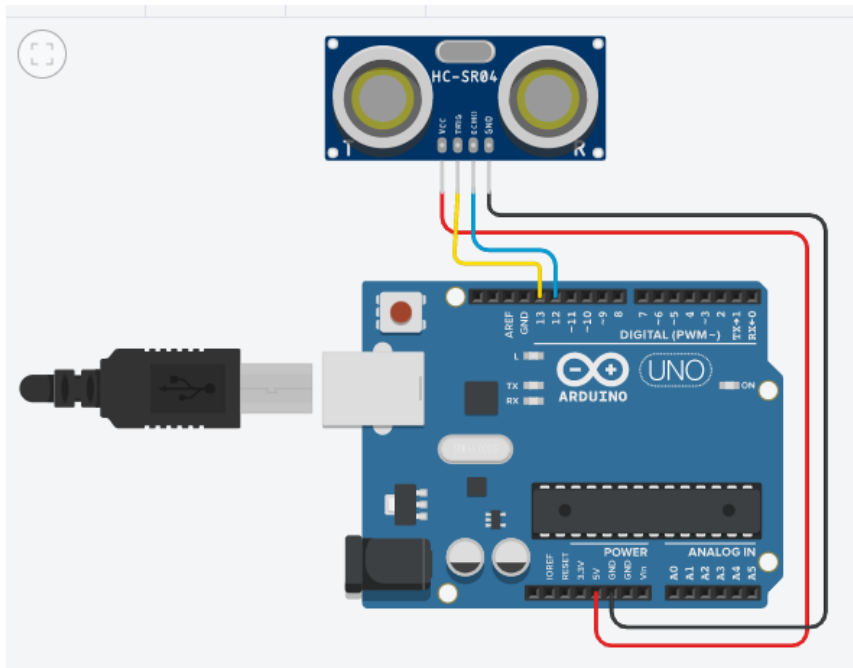
Proyek 12 - Sensor Jarak

Sekarang kita akan menggunakan Sensor Jarak di kit Anda, HC-SR04. Anda hanya membutuhkan satu komponen.

Apa yang Anda butuhkan

| | |
|---------|---|
| HC-SR04 |  |
|---------|---|

Membuat Rangkaian



Masukkan kode

```
/*
  HC-SR04 Ping distance sensor]
  VCC to arduino 5v GND to arduino GND
  Echo to Arduino pin 13 Trig to Arduino pin 12
*/

#define trigPin 13
#define echoPin 12

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  long duration, distance;
  digitalWrite(trigPin, LOW); // Added this line
  delayMicroseconds(2);       // Added this line
  digitalWrite(trigPin, HIGH);
  // delayMicroseconds(1000); - Removed this line
  delayMicroseconds(10);      // Added this line
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1;

  Serial.print(distance);
  Serial.println(" cm");
  delay(500);
}
```

Masukkan kodenya, lalu tekan tombol Serial Monitor di Arduino IDE. Anda sekarang akan mendapatkan bacaan setiap setengah detik yang menunjukkan pembacaan dari sensor jarak.

Proyek 12 – Ikhtisar Kode

Program pada Sketch di atas akan mengaktifkan pin Trigger selama 10 μ s, kemudian akan menunggu hingga pin Echo bernilai HIGH.

```
duration = pulseIn(echoPin, 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.

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.

Proyek 12 – Ikhtisar Perangkat Keras

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.

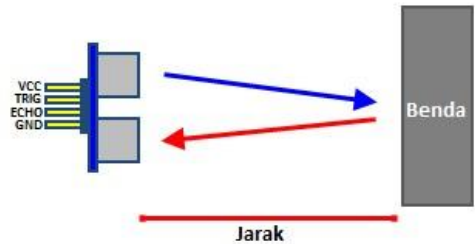
Berikut ini adalah beberapa aplikasi dari gelombang ultrasonik:

- 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 posisi sekelompok ikan.

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.

Dalam program tersebut, kami ingin menghitung jarak suatu benda di depan sensor ultrasonik. Sensor ini dapat mengirim "ping" pada saat tertentu dan menerima ping yang memantul kembali pada suatu objek pada saat tertentu.

Sebuah ping tidak lain adalah suara yang tidak bisa didengar manusia dan inilah mengapa sensor ini disebut "ultrasonik".

Sensor mengirim ping pada satu waktu t_1 dan menerima ping yang memantul pada saat t_2 .

Mengetahui kecepatan suara, perbedaan waktu $\Delta t = t_2 - t_1$ dapat memberikan gambaran tentang jarak suatu benda.

Contoh, jika $\Delta t = 500$ mikrodetik, kita tahu butuh 250 mikrodetik agar ping mengenai objek dan 250 mikrodetik lagi untuk kembali.

Perkiraan kecepatan suara di udara kering diberikan oleh rumus:

$$c = 331,5 + 0,6 * [\text{suhu udara dalam derajat Celcius}]$$

Pada 20 ° C, $c = 331,5 + 0,6 * 20 = 343,5 \text{ m / s}$

Jika kita mengubah kecepatan dalam sentimeter per mikrodetik, kita mendapatkan:

$$c = 343,5 * 100/1000000 = 0,03435 \text{ cm / dtk}$$

$$\text{Oleh karena itu, jaraknya, } D = (\Delta t / 2) * c$$

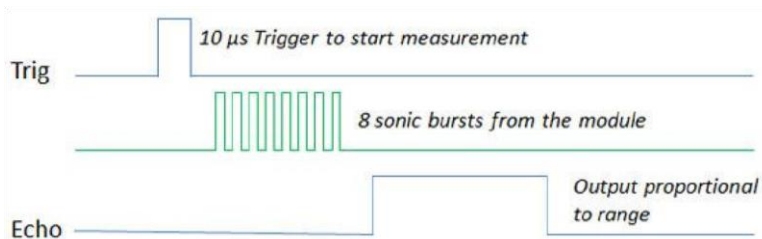
$$\text{atau } D = 250 * 0,03435 = 8,6 \text{ cm}$$

Selain menggunakan Kecepatan Suara, kita juga bisa menggunakan "Pace of Sound".

$$\text{Pace of Sound} = 1 / \text{Kecepatan Suara} = 1 / 0,03435 = 29,1 \text{ ss / cm}$$

Dalam hal ini persamaan untuk menghitung jarak menjadi: $D = (\Delta t / 2) / \text{Pace of sound}$




$$\text{dan untuk contoh di atas: } D = 250 / 29.1 = 8.6 \text{ cm}$$



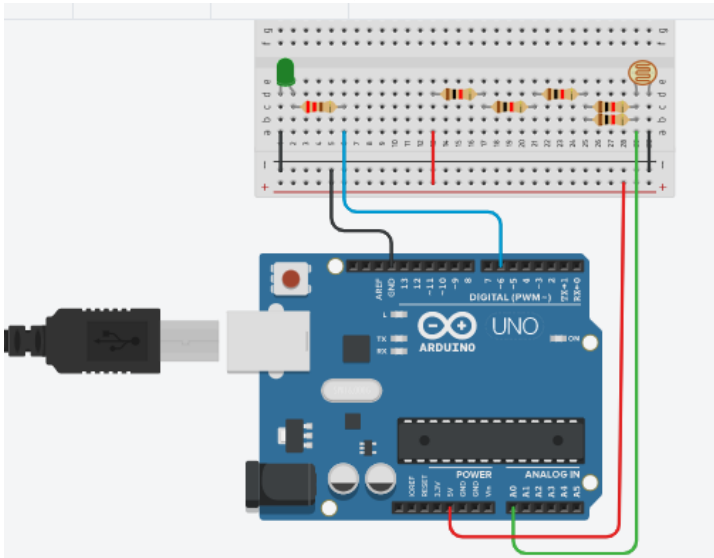
Proyek 13 - Sensor cahaya

Dalam proyek ini kita akan menggunakan Light Dependent Resistor (LDR) dalam kit kita untuk membaca nilai darinya dan menyesuaikan kecepatan LED berkedip.

Apa yang Anda butuhkan

| | |
|--------------------------------|---|
| Light Dependent Resistor (LDR) |  |
| Resistor 220 Ω |  |
| Resistor 1k Ω x 5 | |
| Led Hijau |  |

Membuat Rangkaiannya.



Masukkan kode

Masukkan kode tersebut, lalu unggah ke Arduino Anda. Anda akan melihat LED berkedip dan mati. Jika Anda menutupi LDR (Light Dependent Resistor), Anda akan melihat LED berkedip lebih lambat. Sekarang arahkan cahaya terang ke LDR dan Anda akan melihatnya terbang lebih cepat.

```
// Proyek 13 - Sensor cahaya

// Pin we will connect to LED
int ledPin = 6;    // Pin connected to LDR
int ldrPin = 0;    // Value read from LDR
int lightVal = 0;

void setup()
{
    // Set both pins as outputs
    pinMode(ledPin, OUTPUT);
}
void loop()
{
    // Read in value from LDR
    lightVal = analogRead(ldrPin);    // Turn LED on
    digitalWrite(ledPin, HIGH);       // Delay of length
    delay(lightVal);                   // Turn LED off
    digitalWrite(ledPin, LOW);        // Delay again
    delay(lightVal);
}
```

Proyek 13 – Ikhtisar Kode

Kode ini cukup sederhana dan Anda seharusnya sudah bisa mengerjakan sendiri apa yang dilakukannya sekarang.

Kode dimulai dengan menginisialisasi variabel yang terkait dengan Digital Pin 6, dimana LED terhubung dan Analog Pin 0, dimana LDR

terhubung. Kami juga menginisialisasi variabel yang disebut `lightVal` yang akan menyimpan nilai merah dari LDR.

```
int ledPin = 6;    // Pin connected to LDR
int ldrPin = 0;    // Value read from LDR
int lightVal = 0;
```

Fungsi pengaturan mengatur pinmode dari pin LED ke output.

```
pinMode(ledPin, OUTPUT);
```

Di loop utama program kita membaca nilai analog dari Analog Pin 0 dan menyimpannya dalam variabel '`lightVal`'.

```
lightVal = analogRead(ldrPin);    // Turn LED on
```

Kemudian LED dinyalakan dan dimatikan, dengan jeda sebesar nilai yang terbaca dari pin analog.

```
digitalWrite(ledPin, HIGH);
delay(lightVal);
digitalWrite(ledPin, LOW);
delay(lightVal);
```

Karena lebih banyak cahaya jatuh pada LDR, nilai yang dibaca dari Pin Analog 0 berkurang dan LED berkedip lebih cepat.

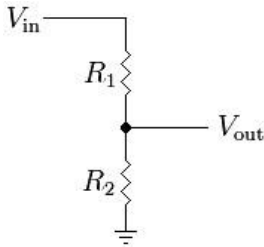
Mari kita cari tahu bagaimana sirkuit ini bekerja.

Proyek 13 – Ikhtisar Perangkat Keras

Satu-satunya komponen tambahan yang digunakan dalam rangkaian ini adalah LDR atau Light Dependent Resistor (terkadang disebut fotoresistor). Sebuah LDR awalnya memiliki resistansi yang sangat tinggi. Tapi, saat cahaya jatuh di atasnya, resistansi akan turun, memungkinkan lebih banyak arus masuk.



LDR kami dihubungkan secara seri dengan 3 x 1.5K Ω Resistor dan input ke Pin Analog 0 berada di antara 2. Inilah yang dikenal sebagai pembagi tegangan. Kami akan menjelaskan ini sebentar lagi.



Rangkaian seri 3 x 1.5K memberikan resistansi total 4500 Ω (4.5K Ω). Resistor dalam seri memiliki resistansi yang sama dengan jumlah resistansi masing-masing. Dalam hal ini nilainya adalah 3 x 1500 = 4500.

Pembagi tegangan adalah rangkaian yang terdiri dari dua resistansi di suplai tegangan. Output antara dua resistansi akan memberikan tegangan yang lebih rendah tergantung pada nilai kedua resistor.

Diagram di sebelah kiri menunjukkan pembagi tegangan yang terdiri dari dua resistor. Nilai Vout akan lebih rendah dari nilai Vin.

Untuk menghitung nilai Vout, kami menggunakan perhitungan berikut:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

Kami menyediakan 5 volt ke sirkuit jadi mari kita cari tahu nilai apa yang akan kita dapatkan. Dengan menggunakan multimeter, saya telah mengukur resistansi dari LDR dalam kondisi berbeda.

| Kondisi | Resistansi |
|--------------------------------|--------------|
| LDR Tertutup Jari | 8K Ω |
| Cahaya di kamar (hari mendung) | 1K Ω |
| Berada di bawah cahaya terang | 150 Ω |

Jadi dengan menggunakan nilai resistansi ini, tegangan input dan perhitungan yang kami cantumkan di atas, kira-kira. tegangan keluaran dapat dihitung dengan demikian:

| V in | R ₁ | R ₂ | Vout |
|------|----------------|----------------|--------|
| 5v | 4500 Ω | 8000 Ω | 3.2 v |
| 5v | 4500 Ω | 1000 Ω | 0.9 v |
| 5v | 4500 Ω | 150 Ω | 0.16 v |

Seperti yang Anda lihat, ketika resistansi LDR (R₂) berkurang, tegangan yang keluar dari pembagi tegangan juga berkurang, membuat nilai yang terbaca dari Pin Analog lebih rendah dan oleh karena itu mengurangi penundaan sehingga LED berkedip lebih cepat.




Rangkaian pembagi tegangan juga dapat digunakan untuk menurunkan tegangan ke yang lebih rendah jika Anda menggunakan 2 resistor standar, daripada resistor dan LDR (yang merupakan resistor variabel). Sebagai alternatif, Anda dapat menggunakan trimpot sehingga Anda dapat mengatur voltase keluar dengan memutar kenop.

Proyek 14 - Shift Register 8-Bit Binary Counter

Baik, kita sekarang akan mempelajari beberapa hal yang cukup canggih sehingga Anda mungkin ingin minuman kopi sebelum melangkah lebih jauh.

Dalam proyek ini kita akan menggunakan IC tambahan (Sirkuit Terpadu) dalam bentuk Shift Register/ Register Geser, untuk memungkinkan kita menggerakkan LED untuk menghitung dalam Biner (kami akan menjelaskan apa itu biner segera). Dalam proyek ini kita akan menggerakkan 8 LED secara mandiri hanya dengan menggunakan 3 pin keluaran dari Arduino.

Apa yang Anda butuhkan

| | |
|--------------------------------------|--|
| IC 74HC595 Shift Registers |  |
| Led Kuning - Nyala Merah 3mm (8 pcs) |  |
| Resistor 220Ω (8 pcs) |  |

Membuat Rangkaian

Periksa diagram dengan cermat. Hubungkan 3.3v ke rel atas Breadboard Anda dan Ground ke bawah. IC ada cekungan di salah satu ujungnya, Pin 1 di kiri bawah cekungan,

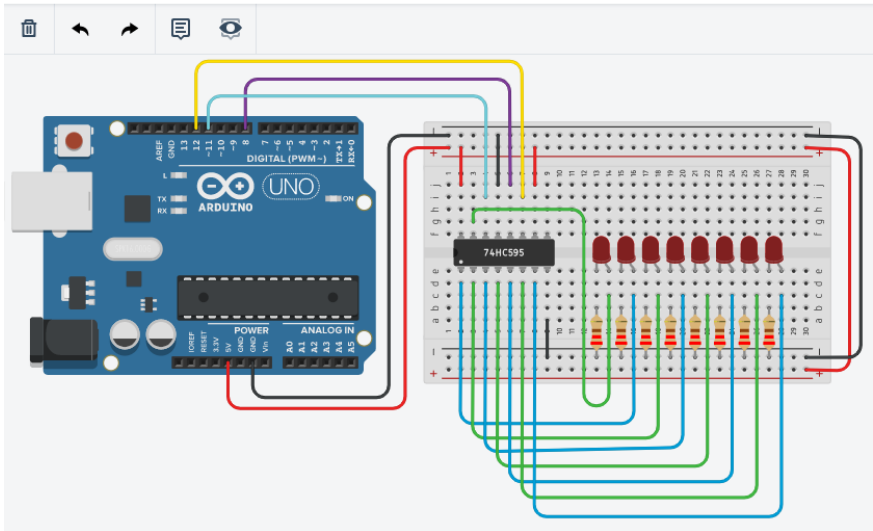
Pin 8 di kanan bawah, Pin 9 di kanan atas dan Pin 16 di kiri atas.

Anda sekarang membutuhkan kabel untuk menghubungkan dari suplai 3.3v ke Pin 10 & 16. Juga, kabel dari Ground ke Pin 8 & 13.

Sebuah kabel dihubungkan dari Pin Digital 8 ke Pin 12 pada IC. Satu lagi beralih dari Digital Pin 12 ke Pin 14 dan terakhir satu dari Digital Pin 11 ke Pin 11.

8 LED memiliki resistor 240 Ω antara katoda dan ground, kemudian anoda LED 1 menuju Pin 15. Anoda LED 2 sampai 8 menuju Pin 1 sampai 7 pada IC.

Proyek 14 - Shift Register 8-Bit Binary Counter



Setelah Anda menghubungkan semuanya, lakukan pemeriksaan terakhir bahwa kabel Anda benar dan IC serta LED berada di jalur yang benar. Kemudian masukkan kode berikut.

Masukkan kode

Masukkan kode berikut dan unggah ke Arduino Anda. Setelah kode dijalankan, Anda akan melihat LED menyala dan mati secara individual saat LED dihitung dalam Biner dari 0 hingga 255, lalu mulai lagi.

```
// Project 14

//Pin connected to Pin 12 of 74HC595 (Latch)
int latchPin = 8;
//Pin connected to Pin 11 of 74HC595 (Clock)
int clockPin = 12; //Pin connected to Pin 14 of 74HC595
(Data)
int dataPin = 11;

void setup() {
  //set pins to output
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {
  //count from 0 to 255
  for (int i = 0; i < 256; i++) {
    //set latchPin low to allow data flow
    digitalWrite(latchPin, LOW);
    shiftOut(i);
    //set latchPin to high to lock and send data
    digitalWrite(latchPin, HIGH);
    delay(500);
  }
}

void shiftOut(byte dataOut) {
  // Shift out 8 bits LSB first,
  // on rising edge of clock

  boolean pinState;

  //clear shift register ready for sending data
  digitalWrite(dataPin, LOW);
```

```

    digitalWrite(clockPin, LOW); // for each bit in
dataOut send out a bit

    for (int i = 0; i <= 7; i++) {
        //set clockPin to LOW prior to sending bit
        digitalWrite(clockPin, LOW);

        // if the value of DataOut and (logical AND) a
bitmask
        // are true, set pinState to 1 (HIGH)
        if ( dataOut & (1 << i) ) {
            pinState = HIGH;
        }
        else {
            pinState = LOW;
        }

        //sets dataPin to HIGH or LOW depending on pinState
        digitalWrite(dataPin, pinState);
        //send bit out on rising edge of clock
        digitalWrite(clockPin, HIGH);
    }
    //stop shifting out data
    digitalWrite(clockPin, LOW);
}

```

Sistem Bilangan Biner

Sekarang sebelum kita melihat kode dan perangkat keras untuk Proyek 15, sekarang saatnya untuk melihat Sistem Bilangan Biner, karena sangat penting untuk memahami Biner agar dapat berhasil memprogram mikrokontroler.

Manusia menggunakan Basis 10, atau sistem bilangan Desimal, karena kita memiliki 10 jari di tangan kita. Komputer tidak memiliki jari sehingga cara terbaik bagi komputer untuk menghitung adalah dengan menggunakan jari yang setara, yang dalam keadaan ON atau OFF (1 atau 0). Perangkat logika, seperti komputer, dapat mendeteksi apakah ada tegangan (1) atau jika tidak (0) dan karenanya

menggunakan sistem bilangan biner, atau basis 2 karena sistem bilangan ini dapat dengan mudah direpresentasikan dalam rangkaian elektronik dengan status tegangan tinggi atau rendah.

Dalam sistem bilangan kita, basis 10, kita memiliki 10 digit mulai dari 0 hingga 9. Ketika kita menghitung ke digit berikutnya setelah 9 digit kembali ke nol, tetapi 1 bertambah ke kolom puluhan di sebelah kirinya. Setelah kolom puluhan mencapai 9, menambahkannya dengan 1 akan mengembalikannya ke nol, tetapi menambahkan 1 ke kolom ratusan ke kiri, dan seterusnya.

000.001.002.003.004.005.006.007.008.009
010.011.012.013.014.015.016.017.018.019 020.021.023

Dalam Biner hal yang persis sama terjadi, kecuali digit tertinggi adalah 1 sehingga menambahkan 1 ke 1 hasil di digit reset ke nol dan 1 ditambahkan ke kolom di sebelah kiri.

000,001
010,011
100,101 ...3.

Angka 8 bit (atau byte) direpresentasikan seperti ini

| 2⁷ | 2⁶ | 2⁵ | 2⁴ | 2³ | 2² | 2¹ | 2⁰ |
|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

Angka di atas dalam Biner adalah 1001011 dan masuk Desimal ini adalah 75.

Ini dikerjakan seperti ini:

$$1 \times 1 = 1$$

$$1 \times 2 = 2$$

$$1 \times 8 = 8$$

$$1 \times 64 = 64$$

Tambahkan semuanya dan Anda mendapatkan 75.

Berikut beberapa contoh lainnya:

| Dec | 2^7 128 | 2^6 64 | 2^5 32 | 2^4 16 | 2^3 8 | 2^2 4 | 2^1 2 | 2^0 1 |
|-----|--------------|-------------|-------------|-------------|------------|------------|------------|------------|
| 75 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 27 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 100 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 127 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 255 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

...dan seterusnya.

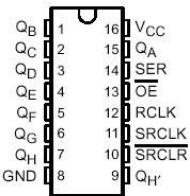
Proyek 14 – Ikhtisar Perangkat Keras

Jadi sekarang setelah Anda memahami biner (atau setidaknya saya harap Anda mengerti), pertama-tama kita akan melihat perangkat kerasnya, sebelum melihat kodenya.

Kami akan melakukan hal yang sebaliknya untuk proyek ini dan melihat perangkat keras sebelum kami melihat kodenya.

Kami menggunakan Shift Register. Khususnya tipe Shift Register 74HC595. Jenis Register Geser ini adalah register geser 8-bit serial-in, serial atau paralel-out dengan kait keluaran. Ini berarti Anda dapat mengirim data ke Register Geser secara seri dan mengirimkannya secara paralel. Secara seri berarti 1 bit dalam satu waktu. Paralel berarti banyak bit (dalam hal ini 8) dalam satu waktu. Jadi Anda memberikan data Register Geser (dalam bentuk 1's dan 0's) satu bit pada satu waktu, kemudian mengirimkan 8 bit semuanya pada waktu yang sama. Setiap bit didorong seiring dengan bit berikutnya yang dimasukkan. Jika bit ke-9 dimasukkan sebelum Latch diatur ke TINGGI, maka bit pertama yang dimasukkan akan dihilangkan dari akhir baris dan hilang selamanya.

Register Geser biasanya digunakan untuk konversi data serial ke paralel. Dalam kasus kami, karena data yang merupakan output adalah 1's dan 0's (atau 0v dan 3.3v), kita dapat menggunakannya untuk menghidupkan dan mematikan bank yang terdiri dari 8 LED.



Shift Register, untuk proyek ini, hanya membutuhkan 3 input dari Arduino. Output dari Arduino dan input dari 595 adalah sebagai berikut:

| Arduino Pin | 595 Pin | Description |
|-------------|---------|------------------------------|
| 8 | 12 | Storage Register Clock Input |
| 11 | 14 | Serial Data Input |
| 12 | 11 | Shift Register Clock Input |

Kita akan merujuk ke Pin 12 sebagai Pin Jam, Pin 14 sebagai Pin Data dan Pin 11 sebagai Pin Latch.

Bayangkan Latch sebagai gerbang yang akan memungkinkan data keluar dari 595. Ketika gerbang diturunkan (LOW) data di 595 tidak bisa keluar, tetapi data bisa dimasukkan. Saat gerbang dinaikkan (TINGGI) data tidak bisa lagi dimasukkan, tetapi data di Register Geser dilepaskan ke 8 Pin (QA-QH). Jam hanyalah pulsa 0's dan 1's dan Data Pin adalah tempat kami mengirim data dari Arduino ke 595.

Untuk menggunakan Shift Register, Pin Latch dan Pin Jam harus disetel ke LOW. Pin Latch akan tetap RENDAH sampai 8 bit telah disetel. Hal ini memungkinkan data untuk dimasukkan ke dalam Register Penyimpanan (register penyimpanan hanyalah tempat di dalam IC untuk menyimpan 1 atau 0). Kami kemudian menyajikan sinyal TINGGI atau RENDAH di Pin Data dan kemudian mengatur Pin Jam ke TINGGI. Dengan mengatur Pin Jam ke TINGGI, ini menyimpan data yang disajikan di Pin Data ke dalam Register Penyimpanan. Setelah ini selesai, kami mengatur Jam ke LOW lagi, lalu menyajikan bit data berikutnya di Pin Data. Setelah kami melakukan ini 8 kali, kami telah mengirimkan nomor 8 bit penuh ke 595. Pin Latch sekarang dimunculkan yang mentransfer data dari Storage Register ke dalam Shift Register dan mengeluarkannya dari QA ke QH (Pin 15, 1 sampai 7).

Saya telah menghubungkan Logic Analyzer (perangkat yang memungkinkan Anda melihat 1 dan 0 keluar dari perangkat digital) ke 595 saya saat program ini berjalan dan gambar di bagian bawah halaman menunjukkan hasilnya.

Urutan kejadian di sini adalah:

| Pin | State | Description |
|-------|-------|---|
| Latch | LOW | Latch lowered to allow data to be entered |
| Data | HIGH | First bit of data (1) |
| Clock | HIGH | Clock goes HIGH. Data stored. |
| Clock | LOW | Ready for next Bit. Prevent any new data. |
| Data | HIGH | 2 nd bit of data (1) |
| Clock | HIGH | 2 nd bit stored |
| ... | ... | ... |
| Data | LOW | 8 th bit of data (0) |
| Clock | HIGH | Store the data |
| Clock | LOW | Prevent any new data being stored |
| Latch | HIGH | Send 8 bits out in parallel |

Pada gambar di bawah, Anda dapat melihat bahwa bilangan biner 00110111 (membaca dari kanan ke kiri) atau Desimal 55 telah dikirim ke chip.

Jadi untuk meringkas penggunaan satu Shift Register dalam proyek ini, kami memiliki 8 LED yang terpasang pada 8 keluaran Register. Latch diatur ke LOW untuk mengaktifkan entri data. Data dikirim ke Pin Data, sedikit demi sedikit, Pin CLock diatur ke TINGGI untuk menyimpan data itu, lalu kembali ke bawah untuk siap untuk bit berikutnya. Setelah semua 8 bit dimasukkan, kait diatur ke TINGGI yang mencegah entri data lebih lanjut dan menetapkan 8 pin keluaran ke Tinggi (3,3 v atau RENDAH (0 volt) tergantung pada status Register.

Jika Anda ingin membaca lebih lanjut tentang register geser yang Anda miliki di kit Anda, lihat nomor seri pada IC (misalnya 74HC595N atau SN74HC595N, dll.) Dan masukkan ke Google. Anda kemudian dapat menemukan lembar data khusus untuk IC dan membaca lebih lanjut tentangnya.

Saya penggemar berat 595 chip. Ini sangat serbaguna dan tentu saja dapat meningkatkan jumlah pin keluaran digital yang dimiliki Arduino. Arduino standar memiliki 19 Output Digital (6 Pin Analog juga dapat digunakan sebagai Pin Digital bernomor 14 hingga 19). Menggunakan Register Geser 8-bit Anda dapat memperluasnya menjadi 49 (6 x 595's ditambah satu pin cadangan yang tersisa). Mereka juga beroperasi sangat cepat, biasanya pada 100MHz. Artinya Anda dapat mengirim data sekitar. 100 juta kali per detik jika Anda mau. Ini berarti Anda juga dapat mengirim sinyal PWM melalui perangkat lunak ke IC dan mengaktifkan kontrol kecerahan LED juga.

Karena output hanya ON dan OFF dari tegangan output, mereka juga dapat digunakan untuk menghidupkan dan mematikan perangkat bertenaga rendah (atau bahkan perangkat bertenaga tinggi lainnya dengan menggunakan transistor atau relai) atau untuk mengirim data ke perangkat (mis. printer dot matrix lama atau perangkat serial lainnya).



Proyek 15 – LCD Display Character 1602

Library LiquidCrystal memungkinkan Anda untuk mengontrol tampilan LCD yang kompatibel dengan driver Hitachi HD44780. LCD ini mempunyai antarmuka 16-pin.

Contoh sketsa ini mencetak "Hello World!" ke LCD dan menunjukkan waktu dalam hitungan detik sejak Arduino direset.

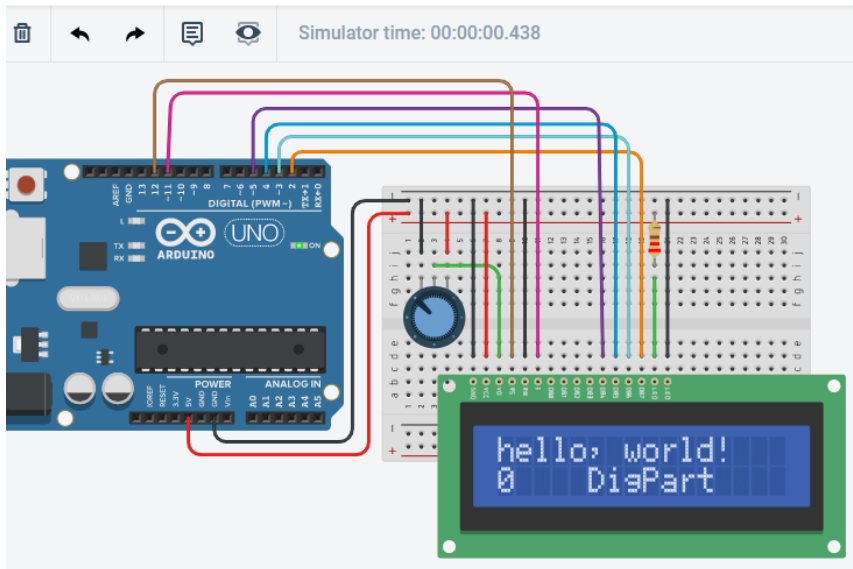


keluaran sketch pada LCD 16x2 output of the sketch on a 16x2 LCD

Apa yang Anda butuhkan

| | |
|---|--|
| LCD 2x16 16x2 1602 Display Module Arduino | |
| Trimpot 10K | |
| Resistor 220 ohm | |

Membuat Rangkaian



Sebelum memasang kabel layar LCD ke papan Arduino Anda, kami sarankan untuk menyolder strip header pin ke konektor jumlah pin 14 (atau 16) pada layar LCD. Untuk menyambungkan layar LCD Anda ke papan, sambungkan pin berikut:

- Pin LCD RS ke pin digital 12
- Pin LCD Enable ke pin digital 11
- Pin LCD D4 ke pin digital 5
- Pin LCD D5 ke pin digital 4
- Pin LCD D6 ke pin digital 3
- Pin LCD D7 ke pin digital 2
- Pin LCD R / W ke GND
- Pin LCD VSS ke GND
- Pin LCD VCC ke 5V
- LCD LED + ke 5V melalui resistor 220 ohm
- LCD LED - ke GND

Selain itu, sambungkan pot 10k ke + 5V dan GND, dengan wiper (output) ke pin VO layar LCD (pin3).

Masukkan kode

```
/*
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/LiquidCrystalHelloWorld
*/

// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD
interface pin
// with the arduino pin number it is connected to
const int rs=12, en=11, d4=5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting
  begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

Proyek 15 – Ikhtisar Perangkat Keras

LCD memiliki antarmuka paralel, artinya mikrokontroler harus memanipulasi beberapa pin antarmuka sekaligus untuk mengontrol tampilan. Antarmuka terdiri dari pin berikut:

Pin register select (RS) yang mengontrol di mana dalam memori LCD Anda menulis data. Anda dapat memilih register data, yang menyimpan apa yang terjadi di layar, atau register instruksi, di mana pengontrol LCD mencari instruksi tentang apa yang harus dilakukan selanjutnya.

Pin Read/Write (R/W) yang memilih mode membaca atau mode menulis

Pin Enable yang memungkinkan penulisan ke register

8 pin data (D0 -D7). Status pin ini (tinggi atau rendah) adalah bit yang Anda tulis ke register saat Anda menulis, atau nilai yang Anda baca saat Anda membaca.

Ada juga pin kontras tampilan (Vo), pin catu daya (+ 5V dan Gnd), dan pin Lampu Latar LED (BKlt + dan BKlt-) yang dapat Anda gunakan untuk memberi daya pada LCD, mengontrol kontras tampilan, dan menyalakan serta mematikan LED lampu latar, masing-masing.

Proses pengontrolan tampilan melibatkan penempatan data yang berupa gambar dari apa yang ingin ditampilkan ke dalam register data, kemudian memasukkan instruksi pada register instruksi. Library LiquidCrystal menyederhanakan ini untuk Anda sehingga Anda tidak perlu mengetahui instruksi tingkat rendah.

LCD yang kompatibel dengan Hitachi dapat dikontrol dalam dua mode: 4-bit atau 8-bit. Mode 4-bit membutuhkan tujuh pin I / O

dari Arduino, sedangkan mode 8-bit membutuhkan 11 pin. Untuk menampilkan teks di layar, Anda dapat melakukan hampir semua hal dalam mode 4-bit, jadi contoh menunjukkan cara mengontrol LCD 16x2 dalam mode 4-bit.

Latihan :

1. Gabungkan rangkaian sensor jarak dengan rangkaian LCD sehingga menjadi alat pengukur jarak portabel
2. Gabungkan rangkaian sensor cahaya dengan rangkaian LCD sehingga menjadi alat pengukur cahaya portabel

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.

Kemajuan suatu negeri salah satunya tergantung teknologi yang dikuasainya. Dunia komputer dan mikrokontroller merupakan cikalbakal 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 Memajukan Bangsa dengan Teknologi###