SIMULASI MIKROKONTROLER ARDUINO BERBASIS TINKERCAD

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

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

Pembatasan Pelindungan Pasal 26

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

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

Sanksi Pelanggaran Pasal 113

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

SIMULASI MIKROKONTROLER ARDUINO BERBASIS TINKERCAD

Ahmad Zarkasi

Mulawarman University PRESS

SIMULASI MIKROKONTROLER ARDUINO BERBASIS TINKERCAD

Penulis : Ahmad Zarkasi
Desain Cover : Syaiful Anwar
Tata Letak : Gofur Dyah Ayu
Proofreader : Aditya Timor Eldian

ISBN : 978-623-5262-07-9

Cetakan Pertama : Juni 2022

Ahmad Zarkasi. 2022. *Simulasi Mikrokontroler Arduino Berbasis Tinkercad.* Mulawarman University Press, Samarinda.

Penerbit:
Mulawarman University PRESS
Member of IKAPI & APPTI
Gedung LP2M Universitas Mulawarman
Jl. Krayan, Kampus Gunung Kelua
Samarinda – Kalimantan Timur – Indonesia 75123
Telp/Faks: (0541) 747432, E-mail: mup@lppm.unmul.ac.id

Dicetak oleh:
PENERBIT DEEPUBLISH
(Grup Penerbitan CV BUDI UTAMA)
Anggota IKAPI (076/DIY/2012)
Jl.Rajawali, G. Elang 6, No 3, Drono, Sardonoharjo, Ngaglik, Sleman
Jl.Kaliurang Km.9,3 – Yogyakarta 55581
Telp/Faks: (0274) 4533427, E-mail: cs@deepublish.co.id
Website: www.deepublish.co.id / www.penerbitdeepublish.com

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah Swt. atas rahmat dan karunia-Nya, sehingga buku *Simulasi Mikrokontroler Arduino Berbasis Tinkercad* ini dapat terselesaikan. Penyusunan buku ini dilatarbelakangi oleh perlunya memperkenalkan media alternatif bagi kalangan yang ingin mempelajari mikrokontroler Arduino bahkan dari *basic* sekalipun. Media tersebut berupa *platform* simulasi yang bernama Tinkercad. Tinkercad merupakan *platform* simulasi *online* berbasis *website*. Oleh karena itu, pengguna tidak perlu melakukan instalasi di komputer. Karena menggunakan sistem *online*, maka Tinkercad juga dapat diakses melalui *smartphone*, *tablet*, dan lain sebagainya, sehingga menjadi lebih fleksibel.

Selain berisi petunjuk simulasi mikrokontroler, buku ini dilengkapi dengan beberapa materi penunjang yang penting khususnya bagi para pemula seperti ulasan tentang Mikroprosesor, Mikrokontroler dan Arduino; instalasi *software* Arduino IDE dan *platform* Tinkercad; serta beberapa *syntax* dasar pada pemrograman Arduino. Percobaan-percobaan yang dimuat pada buku ini disusun berdasarkan komponen-komponen yang tersedia pada *platform* Tinkercad dan tentu saja mudah didapatkan di pasaran.

Rampungnya penulisan buku ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu penulis menyampaikan terima

kasih, khususnya kepada Bapak Dr. Djayus, M.T. (Ketua Jurusan Fisika FMIPA UNMUL), Ibu Dr. Rahmawati Munir, M.Si. (Koordinator Program Studi Fisika FMIPA UNMUL), dan Dr. Syahrir, M.Si. (Kepala Lab. Instrumentasi Fisika FMIPA UNMUL) yang turut memberikan saran dan masukan yang berharga.

Terakhir namun yang terpenting, penulis menyampaikan ucapan terima kasih yang mendalam dan sebesar-besarnya kepada Ibunda dan Ayahanda tercinta (Ibu Zeniyah dan Bapak Haji Marjan), serta kepada kakak dan adik-adik penulis (Hilmi, Zayyin, dan Julia), atas dukungan dan doa sehingga penulis tetap konsisten dalam menulis buku ini hingga selesai. Akhirnya, semoga buku ini bermanfaat bagi banyak orang.

Samarinda, Juni 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
MIKROKONTROLER PLATFORM ARDUINO

1.1. MIKROKONTROLER

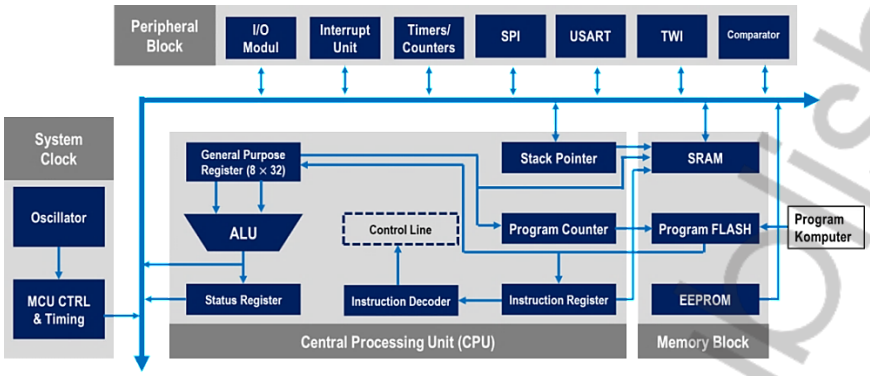
Pada dasarnya Mikrokontroler merupakan sebuah sistem Mikroprosesor lengkap berupa *chip* atau IC yang dapat diprogram. Mikrokontroler berbeda dengan Mikroprosesor serba guna yang digunakan dalam sebuah PC, karena di dalam sebuah Mikrokontroler umumnya juga telah berisi komponen pendukung sistem Mikroprosesor seperti inti prosesor, memori, dan antarmuka I/O. Sedangkan, di dalam Mikroprosesor umumnya hanya berisi CPU saja. Mikrokontroler populer digunakan pada berbagai keperluan otomatisasi seperti sistem kontrol mobil, perangkat medis, pengendali jarak jauh, mesin perkantoran, dan lain-lain. Penggunaan Mikrokontroler lebih ekonomis dibandingkan sebuah desain sistem yang berisikan Mikroprosesor, memori, dan perangkat input/output terpisah. Tabel 1.1 memperlihatkan perbedaan Mikrokontroler dengan Mikroprosesor secara umum.

Tabel 1.1 Perbedaan Mikrokontroler dengan Mikroprosesor

Mikrokontroler	Mikroprosesor
Mikrokontroler digunakan untuk mengeksekusi sebuah fungsi khusus	Mikroprosesor dapat mengeksekusi berbagai fungsi secara bersamaan
Desain dan *hardware* yang ekonomis	Desain dan *hardware* yang relatif mahal

Mikrokontroler	Mikroprosesor
Mudah diganti	Sulit diganti
Telah tertanam teknologi CMOS yang membutuhkan daya yang kecil untuk beroperasi	Membutuhkan konsumsi daya yang tinggi karena harus mengontrol keseluruhan sistem
Terdiri dari CPU, RAM, ROM, dan *port* I/O	Tidak tersusun atas RAM, ROM, dan *port* I/O. Mikroprosesor menggunakan pinnya untuk terhubung dengan perangkat periferal

Di pasaran saat ini telah tersedia berbagai jenis Mikrokontroler yang terus mengalami perkembangan. Salah satu jenis mikrokontroler yang banyak digunakan saat ini adalah Mikrokontroler AVR. AVR merupakan Mikrokontroler RISC (*Reduce Instruction Set Computing*) 8 bit berdasarkan arsitektur Harvard. Kelebihan Mikrokontroler jenis AVR bila dibandingkan dengan Mikrokontroler lain yaitu AVR memiliki kecepatan eksekusi program yang lebih cepat. Hal ini disebabkan karena sebagian besar instruksi dieksekusi dalam satu siklus *clock*. Selain itu, Mikrokontroler AVR memiliki fitur lengkap seperti ADC internal, EEPROM internal, *Timer*/*Counter*, PWM, *port* I/O, komunikasi serial, dan lain-lain. Arsitektur AVR secara garis besar terdiri dari empat diagram blok yaitu CPU, *memory block*, *system clock*, dan *peripheral block*. Gambar 1.1 memperlihatkan arsitektur ATmega16 yang merupakan salah satu *chip* AVR dari keluarga Atmel.



Gambar 1.1. Arsitektur AVR ATmega16

Fungsi dari masing-masing blok pada Gambar 1.1 adalah:

a. **CPU** bertugas untuk memproses semua instruksi atau program yang ada di Mikrokontroler.

b. ***Memory Block*** bertugas menyimpan data dan program. *Program flash* berfungsi untuk menyimpan kode program, sedangkan SRAM dan EEPROM berfungsi menyimpan data.

c. ***System Clock*** bertugas menangani semua *clock* pada CPU, memori, ADC, input/output, dan lain-lain.

d. ***Peripheral Block*** berfungsi untuk menghubungkan Mikrokontroler dengan *environment* di luar Mikrokontroler seperti I/O Modul, *Interrupt Unit*, *Timer/Counter*, SPI, USART, TWI, dan *Comparator*.

1.2. ARDUINO

Arduino merupakan *board* Mikrokontroler yang menggunakan *chip* AVR ATmega yang dirilis oleh Atmel dan bersifat *open source*. Saat ini Arduino menjadi salah satu proyek perangkat keras *open source* yang paling populer. Sifat Arduino yang *open source* membuat Arduino berkembang dengan sangat cepat. Arduino banyak

diminati karena kemudahan dalam penggunaan dan penulisan programnya. Jika pada *board* Mikrokontroler yang lain masih membutuhkan perangkat keras secara terpisah, tidak demikian dengan Arduino yang hanya membutuhkan kabel USB untuk dapat mulai digunakan. Selain itu, Arduino IDE yang merupakan *software* sekaligus *compiler* Arduino menggunakan bahasa pemrograman Bahasa C/C++ dengan versi yang lebih sederhana. *Compiler* yang digunakan Arduino pada dasarnya adalah *compiler* avr-gcc yang sama dengan yang digunakan avr-studio, namun *plugin* yang tersedia di Arduino memungkinkan siapa saja menambahkan *compiler* lain selain avr-gcc, bahkan dukungan untuk prosesor lain selain Atmel AVR.

Berbagai kemudahan serta fleksibilitas yang ditawarkan Arduino menarik antusias banyak orang untuk membangun kreativitas dengan membuat berbagai jenis proyek meski tidak memiliki latar belakang di bidang elektronika, teknik, maupun komputasi. Seperti Mikrokontroler kebanyakan, Arduino lahir dan berkembang kemudian muncul dengan berbagai tipe seperti:

a. Arduino Uno

Menggunakan *chip* ATmega328 sebagai Mikrokontrolernya, memiliki 14 pin I/O digital dan 6 input analog. Untuk pemrogramannya cukup menggunakan koneksi USB *type A to type B*.

b. Arduino Due

Tidak seperti kebanyakan Arduino, Arduino Due menggunakan *chip* yang lebih tinggi yaitu ARM Cortex CPU. Memiliki 54 pin I/O digital dan 12 pin input analog. Pemrogramannya menggunakan Micro USB.

c. Arduino Mega

Hampir mirip dengan Arduino Uno yang sama-sama menggunakan USB *type A to type B* namun menggunakan *chip* lebih tinggi berupa ATmega2560. Memiliki 54 pin I/O digital dan 16 pin input analog.

d. Arduino Leonardo

Menggunakan *chip* ATmega32U4 sebagai Mikrokontrolernya dengan jumlah pin I/O digital sebanyak 20 pin yang mana 7 diantaranya dapat digunakan sebagai PWM dan 12 pin sebagai input analog. Pemrogramannya membutuhkan kabel Micro USB.

e. Arduino Fio

Board Mikrokontroler ini memiliki jumlah pin I/O digital dan input analog yang sama dengan Arduino Uno. Akan tetapi, Arduino Fio memiliki Socket XBee yang memungkinkan Arduino tipe ini dapat diapakai untuk keperluan *project* yang berhubungan dengan *wireless*.

f. Arduino Lilypad

Menggunakan *chip* ATmega168 untuk versi lama dan ATmega328 untuk versi terbarunya. Memiliki 14 pin I/O digital dan 6 pin input analog. Arduino Lilypad berbentuk melingkar yang biasanya digunakan untuk produk pakaian dan tekstil.

g. Arduino Nano

Sudah dilengkapi dengan FTDI untuk pemrograman melalui Micro USB. Arduino tipe Nano menggunakan *chip* ATmega168 dan juga ada yang menggunakan ATmega328. Memiliki 14 pin I/O digital dan 8 pin input analog.

h. Arduino Mini

Arduino Mini memiliki fasilitas yang sama dengan Arduino Nano, hanya saja tidak dilengkapi dengan Micro USB untuk pemrogramannya.

i. Arduino Micro

Berbasis *chip* ATmega32U4 dengan 20 pin I/O digital dan 12 pin input analog.

j. Arduino Ethernet

Arduino tipe ini sudah dilengkapi dengan fasilitas Ethernet yang memungkinkan untuk pembuatan *project* yang berhubungan dengan jaringan LAN pada komputer. Adapun fasilitas pin I/O yang tersedia sama dengan Arduino Uno.

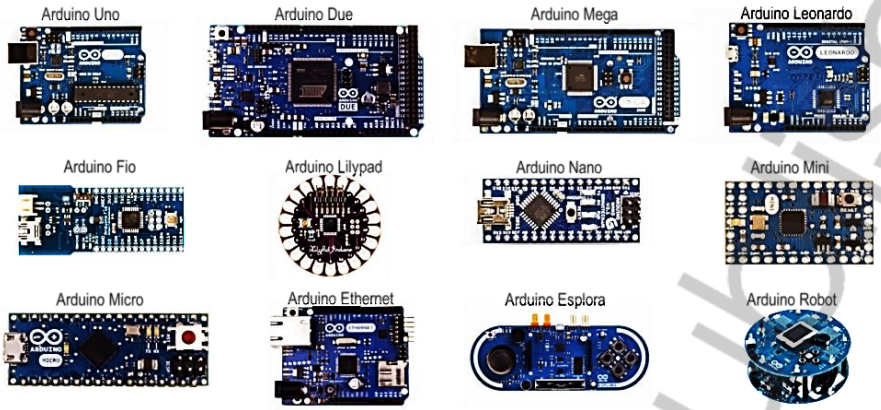
k. Arduino Esplora

Arduino tipe ini sudah dilengkapi dengan beberapa sensor seperti sensor suhu, sensor *accelerometer*, dan sensor cahaya. Selain itu, Arduino ini dilengkapi dengan tombol *joystick*, *linear potensiometer*, 4 *push button*, LED, dan *microphone*.

l. Arduino Robot

Arduino tipe ini merupakan paket komplit Arduino yang sudah berbentuk robot dengan dilengkapi LCD, *speaker*, roda, sensor inframerah, dan lain-lain.

Selain tipe-tipe Arduino di atas, masih terdapat berbagai tipe Arduino lain seperti Arduino BT, Arduino Duemilanove, Arduino Nouve Generazione, Arduino Extreme, dan lain-lain. Gambar 1.2 memperlihatkan tampilan fisik beberapa jenis Arduino yang banyak digunakan.



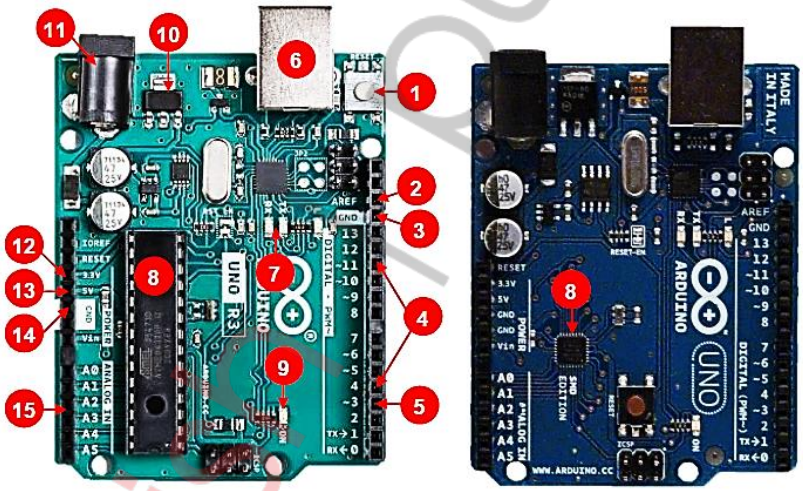
Gambar 1.2. Beberapa tipe *board* Arduino

Dari beberapa tipe *board* Arduino yang telah diuraikan di atas, Arduino Uno merupakan salah satu tipe Arduino yang paling sering digunakan. Arduino jenis ini menjadi pilihan banyak orang termasuk pemula karena ketersediaan referensi yang cukup melimpah baik di buku maupun internet. Selain itu, Arduino Uno memiliki jumlah pin I/O yang sudah mencukupi untuk berbagai keperluan pembuatan *project*. Penggunaan Arduino Uno dapat menjadi fondasi awal yang tepat sebelum menggunakan jenis Arduino yang lebih *advance* seperti Arduino Robot, Arduino Ethernet, Arduino Esplora dan yang lainnya. Berikut adalah spesifikasi dari Arduino Uno:

- Mikrokontroler : ATmega328P
- Tegangan Operasi : 5 V
- Tegangan Input : 7-12V (Rekomendasi)
- Pin I/O Digital : 14 pin (6 pin PWM)
- Pin Input Analog : 6 *channel*
- Arus DC Tiap Pin I/O : 20 mA
- Arus DC pada Pin 3.3V : 50 mA

- Flash Memory : 32 KB (ATmega328)
- SRAM : 2 KB (ATmega328)
- EEPROM : 1 KB (ATmega328)
- Clock Speed : 16 MHz

Berdasarkan dimensinya, *chip* ATmega328P pada Arduino Uno terbagi atas tipe DIP (*Dual Inline Package*) dan SMD (*Surface Mount Device*). Gambar 1.3 memperlihatkan *board* Arduino dengan *chip* bertipe DIP dan SMD.



(a) (b)

Gambar 1.3. Arduino dengan *chip* (a) tipe DIP dan (b) tipe SMD

Bagian-bagian utama pada Arduino Uno berdasarkan Gambar 1.3 adalah sebagai berikut:

1) **Reset**. Fungsi tombol ini bukan untuk menghapus program melainkan menjalankan ulang program dari awal. Untuk me-*reset* program juga dapat dilakukan dengan menghubungkan pin *reset* dengan *ground* secara singkat.

2) **AREF**. Merupakan referensi tegangan untuk input analog.

3) **GND**. *Ground* atau pin negatif dalam sirkuit elektronik.
4) **I/O Digital**. Arduino Uno memiliki 14 pin I/O digital yang tersedia pada pin 0-13 yang berfungsi memberikan nilai logika 0 atau 1 atau dapat berfungsi layaknya saklar. Logika 1 bernilai 5 V, sementara logika 0 bernilai 0 V. Pin berlabel "~" berfungsi sebagai PWM.
5) **PWM**. Menyediakan output PWM sebesar 8-bit (0-255) pada pin 3, 5, 6, 9, 10, dan 11.
6) **Power USB**. Berfungsi menghubungkan *board* Arduino ke komputer melalui koneksi USB sebagai suplai listrik ke *board* atau untuk pemrograman Arduino.
7) **TX/RX (*Transmit*/*Receive*)**. LED TX/RX berkedip saat pemrograman di *chip* atau *board* Arduino berlangsung.
8) **ATmega328P**. Merupakan pusat kontrol Arduino. *Chip* ini diprogram oleh *board* Arduino melalui *software* Arduino IDE.
9) **LED Indikator Power**. Akan menyala saat *board* Arduino diberikan suplai listrik.
10) **Voltage Regulator**. IC ini berfungsi untuk mengatur/menstabilkan tegangan eksternal yang terhubung melalui *power jack*.
11) **Power Jack (DC)**. Untuk koneksi Arduino dengan sumber listrik dengan input DC sebesar 5-12V.
12) **Pin 3,3V**. Sumber tegangan output 3,3 V.
13) **Pin 5V**. Sumber tegangan output 5 V.
14) **GND**. Ground atau pin negatif dalam sirkuit elektronik.
15) **Pin Analog**. Arduino Uno memiliki 6 *channel* input analog yang tersedia pada pin A0-A5. Pin ini berfungsi menerima input analog dari berbagai komponen seperti sensor untuk selanjutnya diubah menjadi bentuk digital menggunakan ADC.

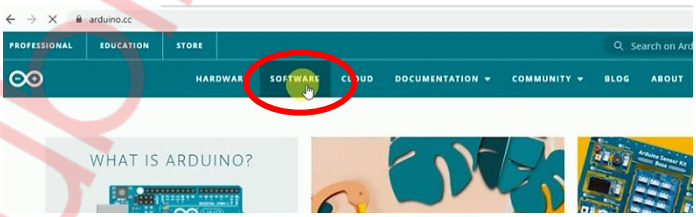
1.3. ARDUINO IDE

Arduino IDE merupakan *software* pemrograman pada *board* Arduino yang menggunakan bahasa C/C++ namun dengan versi yang sudah disederhanakan. IDE merupakan singkatan dari *Integrated Development Environment* yang secara sederhana dapat diartikan sebagai lingkungan terintegrasi yang digunakan untuk keperluan pengembangan program. Disebut sebagai *environment*/lingkungan karena melalui *software* Arduino IDE tersebut dilakukan pemrograman agar dapat melakukan fungsi-fungsi yang dibenamkan melalui *syntax* pemrograman yang telah tersedia. Kode program pada Arduino IDE biasa disebut *sketch* yang jika sudah selesai dibuat pada Arduino IDE dapat langsung di-*compile* dan di-*upload* ke *board* Arduino.

a. Proses *Download* dan Instalasi Arduino IDE

Berikut adalah *step by step* tahapan *download* dan instalasi Arduino IDE menggunakan sistem operasi Windows. Pada sistem operasi lain seperti Linux dan Mac OS juga tidak jauh berbeda.

1) Tahapan *Download*
   - *Software* Arduino bersifat *free access* dan dapat di-*download* di https://www.arduino.cc
   - Pada bagian menu pilih ***software***

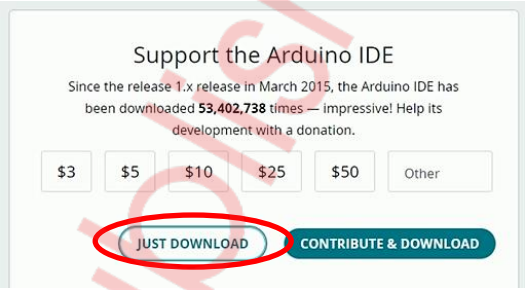


Gambar 1.4. Menu pada arduino.cc

- Pada ***download options***, pilih *installer* sesuai sistem operasi yang digunakan, dalam hal ini misalkan menggunakan sistem operasi Windows.

Gambar 1.5. Opsi *download*

- Jika sudah muncul opsi *download* seperti Gambar 1.6, klik ***Just Download*** atau ***Contribute & Download*** jika ingin berkontribusi. Setelah itu, proses *download* akan berlangsung. Tunggu beberapa saat hingga *file* ter-*download* dengan sempurna.



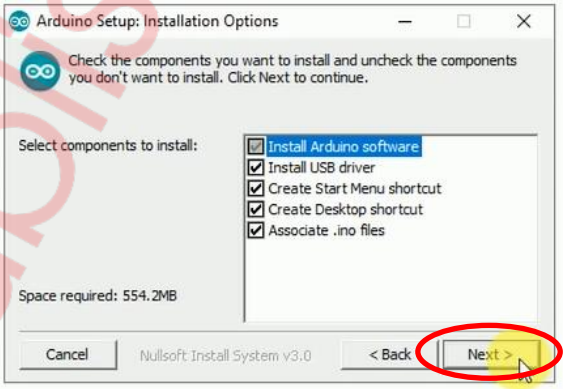
Gambar 1.6. Memulai proses *download*

2) Tahapan Instalasi

- *Double* klik atau ***run as administrator*** pada *file installer* yang telah diunduh. Tunggu beberapa saat hingga muncul ***License Agreement***. Klik ***I Agree***.

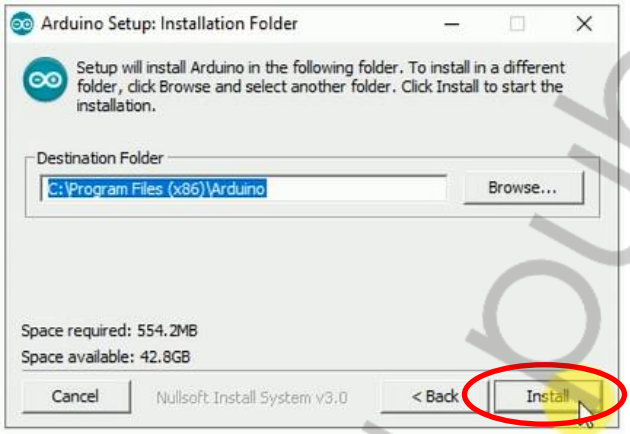
Gambar 1.7. *License Agreement*

- Untuk ***Installation Option*** centang semua opsi dan klik tombol ***Next***.



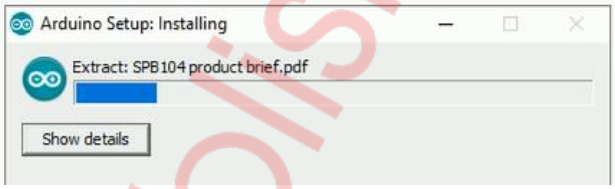
Gambar 1.8. *Installation Options*

- Pada ***Installation Folder*** dapat dipilih folder apa yang akan digunakan sebagai tempat menyimpan program. Selanjutnya klik ***Install***.



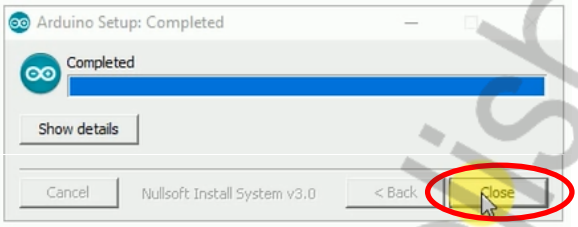
Gambar 1.9. *Installation Folder*

- Proses ekstraksi dan instalasi dimulai. Proses ini biasanya memakan waktu beberapa menit.



Gambar 1.10. Proses ekstraksi dan instalasi

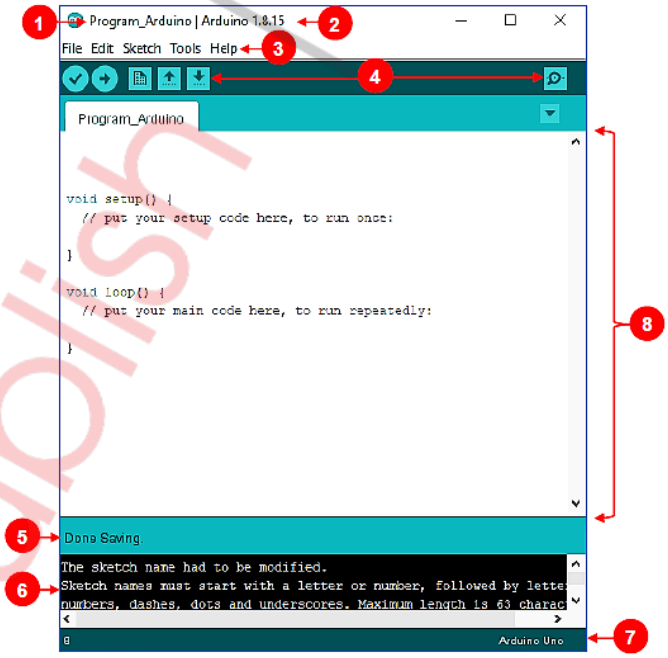
- Tunggu beberapa saat sampai proses instalasi selesai kemudian klik ***close***. *Software* Arduino IDE siap digunakan.



Gambar 1.11. Proses instalasi selesai

b. Jendela *Software* Arduino IDE

Jendela *software* Arduino IDE terdiri dari beberapa bagian seperti menu, tombol *toolbar*, *text editor* untuk penulisan *sketch*, dan *console* sebagai informasi umpan balik. Gambar 1.11 memperlihatkan tampilan jendela *software* Arduino IDE.



Gambar 1.12. Jendela *software* Arduino IDE

Bagian-bagian pada *software* Arduino IDE berdasarkan Gambar 1.11 di atas adalah sebagai berikut:

1) **Nama *file***: *File* disimpan dengan ekstensi *file*.ino.
2) **Versi Arduino IDE**: *Software* Arduino IDE juga memiliki versi yang bersifat *portable* yang dapat digunakan tanpa melalui tahapan instalasi terlebih dahulu.
3) **Menu**
   - ***File***: Berisi beberapa opsi untuk membuat *file* baru (**new**), membuka *file* yang telah disimpan (**open**), menggunakan contoh *sketch* yang tersimpan pada Arduino IDE (**examples**), menyimpan program (**save**, **save as**), dan lain-lain.
   - **Edit**: Berisi opsi untuk membatalkan perintah sebelumnya dan sebaliknya (**Undo**, **Redo**), memperbesar atau memperkecil huruf (**Increase-Decrease Font Size**), menyalin *sketch* sebagai teks HTML, dan berbagai opsi lainnya.
   - **Sketch**: Dapat digunakan untuk memverifikasi dan melakukan *compile* program (**Verify**/**Compile**), meng-*upload* program, memasukkan *library* baru, dan masih banyak lagi termasuk opsi untuk mengubah bentuk *file* biner (**Export Compiled Binary**).
   - **Tools**: Menyediakan fasilitas untuk mengatur format penulisan program agar lebih rapi (**Auto Format**), mengatur *library* (**Manage Libraries**), menampilkan jendela serial (**Serial Monitor**), juga untuk mengatur penggunaan *board* dan *port* yang harus diatur saat menggunakan Arduino (**Board, Port**).
   - **Help**: Menyediakan informasi seputar *troubleshooting*, informasi serta pertanyaan yang sering muncul tentang Arduino (**About Arduino, FAQ**), dan tersedia juga *link* situs lengkap tentang Arduino (**Visit Arduino.cc**).

4) **Toolbar**
   - **Verify**  : Pada versi lama dikenal dengan **Compile**. Tombol ini berfungsi untuk memverifikasi *sketch* yang dibuat apakah sudah sesuai atau masih terdapat kesalahan. Proses verifikasi diperlukan untuk mengubah *sketch* ke bentuk *binary code* sebelum di-*upload* ke Mikrokontroler.
   - **Upload**  : Berfungsi untuk meng-*upload* sketch ke *board* Arduino. Jika program belum diverifikasi menggunakan tombol **Verify**, maka tombol upload ini akan meng-*compile sketch* dan langsung di-*upload* ke *board* Arduino.
   - **New Sketch**  **:** Digunakan untuk membuat *window* dan *sketch* baru.
   - **Open Sketch**  **:** Berfungsi untuk membuka *sketch* yang pernah dibuat dengan ekstensi file.ino.
   - **Save Sketch**  **:** Berfungsi untuk menyimpan *file***.**
   - **Serial Monitor**  **:** Digunakan untuk membuka *interface* komunikasi serial.
5) **Keterangan Aplikasi**: Notifikasi yang dikerjakan oleh *software* seperti "**Compiling**" dan "**Uploading**" saat memverifikasi dan meng-*upload* program ke Arduino akan muncul pada bagian ini.
6) **Console**: Rincian pesan-pesan yang dikerjakan *software* beserta pesan-pesan tentang *sketch* akan muncul pada bagian ini. Contohnya jika terjadi kesalahan penulisan program, maka akan muncul informasi *error*. Hal ini memudahkan pengguna dalam memperbaiki program yang telah dibuat.
7) **Informasi *Board* dan *Port***: Tipe *board* Arduino dan *port* yang digunakan akan ditampilkan pada bagian ini.

8) **Tempat Penulisan Program/*Sketch***: Semua kegiatan pemrograman pada Arduino dilakukan pada bagian ini.

c. *Sketch* Arduino IDE

Struktur *sketch* pada Arduino IDE dikelompokkan menjadi 3 blok utama yaitu **Header**, **Setup**, dan **Loop**. Namun, pada program yang lebih kompleks biasanya dibuat blok-blok lain untuk menyediakan fungsi-fungsi pendukung.

1) **Header**

Bagian ini digunakan sebagai tempat untuk mendeskripsikan atau mendeklarasikan variabel tertentu yang nantinya akan digunakan pada program utama. Selain itu, *library* juga dapat dimasukkan melalui bagian ini. Program yang ditulis pada bagian header hanya dieksekusi sekali saja yaitu pada saat proses *compile*. Berikut contoh kode program untuk memasukkan *library* sensor temperatur LM35 dan LED yang terhubung dengan pin 2.

```
#include < LM35.h>
int Led = 2;
```

2) **Setup**

Bagian ini dijalankan sekali saja yaitu pada saat program baru dieksekusi atau saat program di-*reset*. Bagian ini dapat digunakan untuk mengatur mode pin (input atau *output*), mengatur *baudrate* (kecepatan transfer data), mengatur *timer*, dan lain-lain. Deklarasi/inisialisasi variabel juga dapat dilakukan pada bagian ini. Berikut adalah contoh penulisan kode program pada bagian *setup* berupa pengaturan mode pin dan *baudrate*.

```
void setup()
{
 pinMode (Led, OUTPUT);
 Serial.begin(9600);
}
```

3) Loop

Bagian loop merupakan fungsi utama program yang akan dijalankan secara berulang-ulang dan akan berhenti ketika *board* Arduino sudah tidak menerima daya/power. Berikut adalah contoh kode program untuk menghidupkan dan mematikan LED secara berulang-ulang dengan delay 1 detik.

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

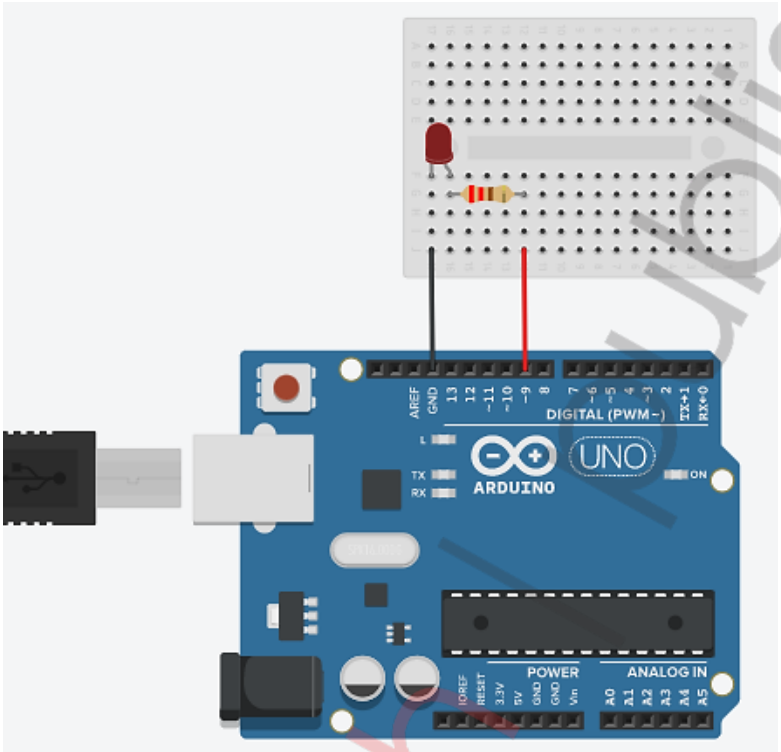
1.4. TAHAPAN PERCOBAAN MENGGUNAKAN ARDUINO

Pada bagian ini akan dibahas mengenai prosedur melakukan percobaan sederhana berupa *blinking LED* (LED berkedip) menggunakan *board* Arduino Uno dengan Arduino IDE sebagai media pembuatan programnya.

a. Siapkan alat dan bahan, berupa:

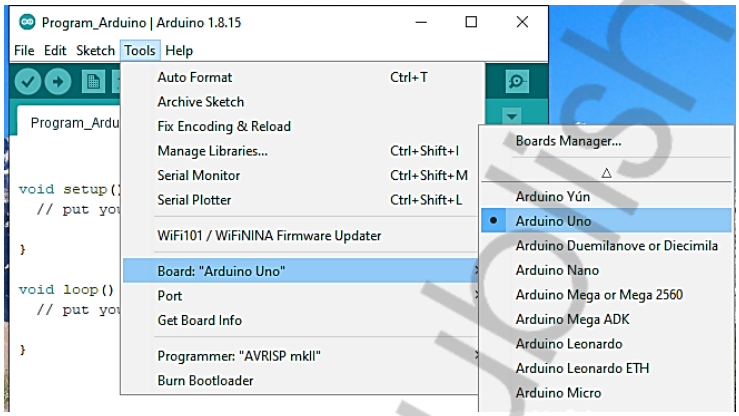
1) 1 unit Arduino Uno beserta kabel USB *type A to type B*
2) 1 buah LED
3) 1 buah resistor 220 Ohm
4) 2 buah kabel *male to male*
5) 1 buah papan rangkaian
6) 1 unit PC yang sudah terinstal Arduino IDE.

b. Buatlah rangkaian seperti Gambar 1.13 berikut:



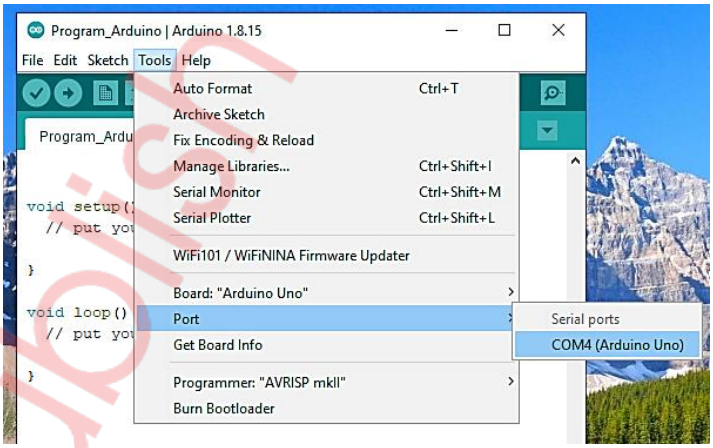
Gambar 1.13. Contoh rangkaian LED sederhana

c. Hubungkan *board* Arduino ke PC melalui kabel USB *type A to type B*. Pastikan LED Power pada *board* Arduino menyala untuk memastikan bahwa *board* sudah terhubung dengan PC.
d. Buka *software* Arduino IDE yang sudah terinstal di PC.
e. Pilih *board* Arduino yang digunakan, dalam hal ini adalah Arduino Uno dengan mengakses **Tools** > **Board** > **Arduino Uno**.



Gambar 1.14. Memilih *board* Arduino

f. Pilih *port* yang akan digunakan melalui **Tools** > **Port**. Biasanya Arduino IDE akan mendeteksi *port* secara otomatis jika *board* Arduino dihubungkan ke PC.



Gambar 1.15. Memilih *Port*

g. Jika sudah mengatur *board* dan *port*, selanjutnya ketik *sketch* berikut di Arduino IDE.

```
int Led = 9;

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

void loop() {
 digitalWrite (Led, HIGH);
 delay (500);
 digitalWrite (Led, LOW);
 delay (500);
}
```

Header

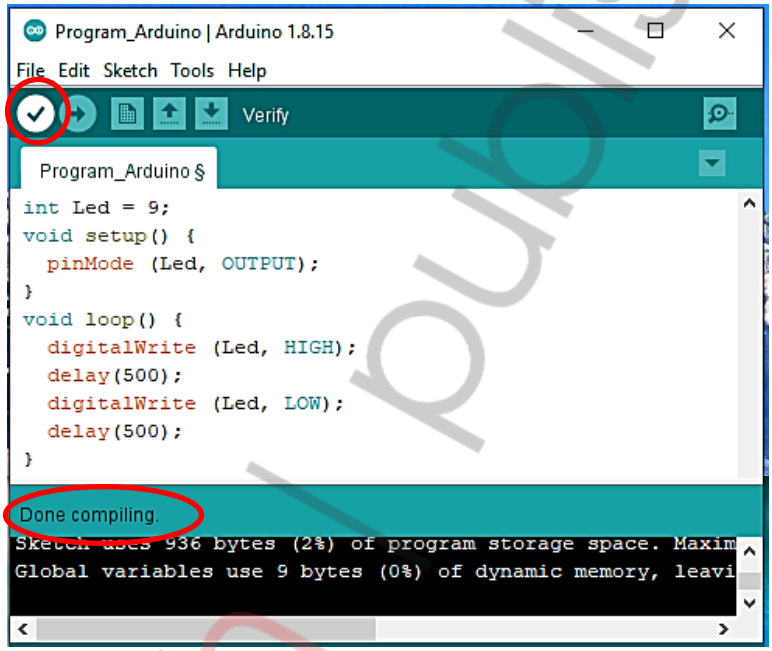
Setup

Loop

Berikut penjelasan masing-masing blok pada *sketch*.

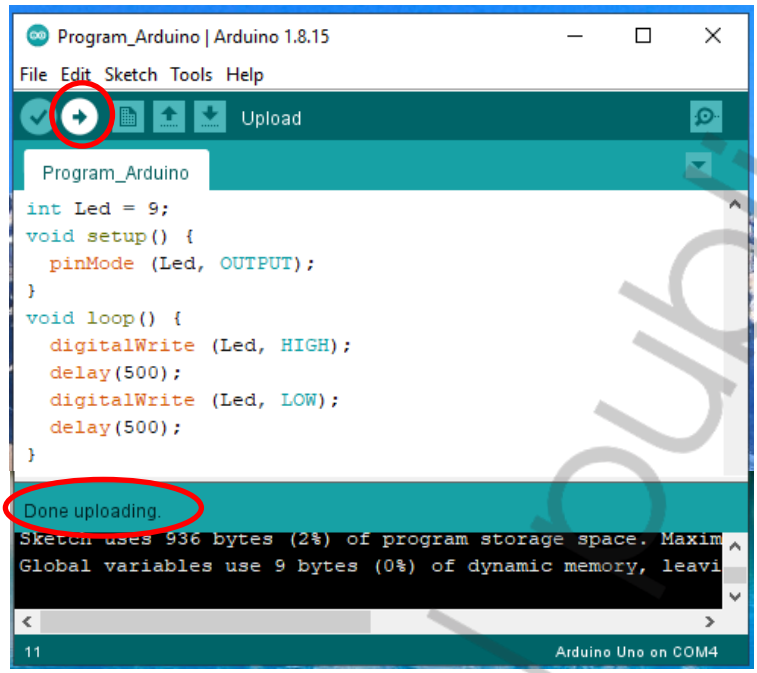
- Header. Berisi inisialisasi variabel **Led** dengan tipe data angka satuan (integer) bernilai 9, artinya pin 9 pada Arduino Uno dijadikan sebagai input/output.
- Setup: Kode **pinMode(Led,OUTPUT)** berfungsi untuk memberi tahu Arduino bahwa pin 9 akan dijadikan sebagai output. Kode ini dipanggil hanya sekali.
- Loop: Bagian ini berisi program utama yang dipanggil dan dijalankan berulang kali. Kode **digitalWrite (Led,HIGH)** berarti LED akan mendapat logika HIGH atau 1 yang berarti menerima tegangan 5V, sehingga LED menyala. Sedangkan, **digitalWrite (Led,LOW)** berarti LED mendapat logika LOW atau 0 yang berarti menerima tegangan 0V dan menyebabkan LED padam. LED akan menyala dan padam secara terus menerus dengan *delay* 500ms sesuai kode **delay(500).**

h. Simpan dan *compile* program dengan mengklik **Verify**. Tunggu beberapa saat sampai muncul keterangan **Done compiling**.



Gambar 1.16. Proses *compile* program

i. *Upload* program ke *board* Mikrokontroler dengan mengklik tombol **Upload**. Proses upload mulai berjalan biasanya ditandai dengan berkedipnya LED TX/RX pada *board* Arduino. Tunggu sampai muncul keterangan **Done uploading**.

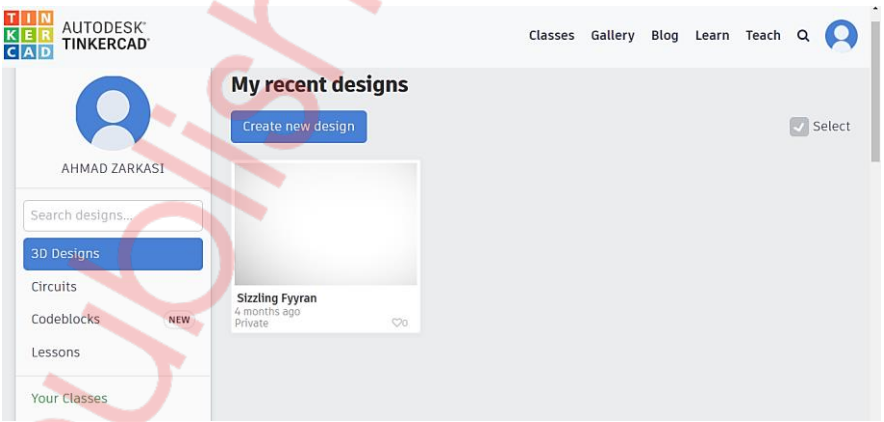


Gambar 1.17. Proses *upload* program

BAB II
PLATFORM TINKERCAD

2.1. PENGENALAN TINKERCAD

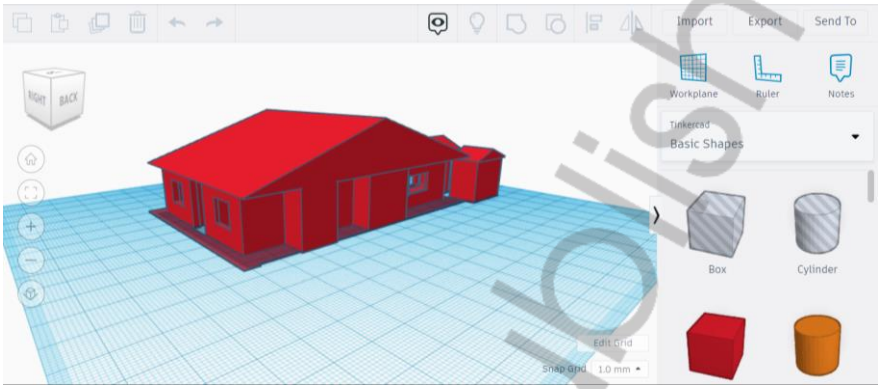
Tinkercad merupakan *platform* desain berbasis *website* yang menyediakan berbagai sarana untuk kebutuhan desain seperti desain 3D, *codeblock*, dan rangkaian elektronika. *Platform* ini dirilis oleh Autodesk sudah cukup lama yaitu pada tahun 2011. Meski demikian, pengembangannya terus berlanjut hingga saat ini. Salah satu kelebihan *platform* berbasis *website* seperti Tinkercad adalah mudah diakses oleh siapa pun tanpa perlu melakukan instalasi *software* di PC. Tak hanya itu, Tinkercad juga dapat digunakan melalui *smartphone* dan *tablet* sehingga menjadi lebih fleksibel dan praktis. Gambar 2.1 memperlihatkan tampilan awal web Tinkercad.



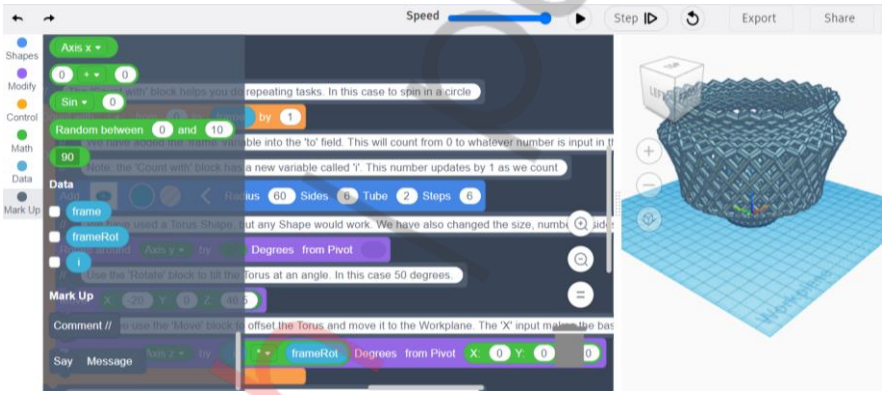
Gambar 2.1. Tampilan awal web Tinkercad

Tinkercad pada dasarnya merupakan *platform* yang bersifat *open source* sehingga siapa saja dapat mengembangkannya dengan mudah. Web yang dimiliki Tinkercad juga dapat dijadikan media pembelajaran secara daring (*online*) karena hampir sama dengan web penyedia layanan pembelajaran *online* seperti Edmodo, Google Classroom, dan Schoology. Hanya saja, Tinkercad lebih dikhususkan untuk kebutuhan desain. Pada web Tinkercad tersedia menu **Classes** yang dapat digunakan oleh pendidik untuk memantau hasil kerja siswa/mahasiswa seperti yang terlihat pada Gambar 2.1. Siswa/mahasiswa yang ingin mengikuti pembelajaran melalui Tinkercad bahkan tidak harus membuat akun atau melakukan registrasi. Cukup dengan memasukkan kode unik serta nickname yang diberikan *educator*, maka siswa/mahasiswa bersangkutan akan langsung dapat bergabung.

Secara umum, menu desain yang disediakan oleh Tinkercad terdiri dari 3 menu utama yaitu **3D Designs**, **Codeblocks**, dan **Circuits**. Penggunaan menu **3D Designs** dan **Codeblocks** pada dasarnya sama yaitu membuat objek 3 dimensi. Perbedaannya terletak pada proses dan tujuan pembuatannya. Pembuatan objek menggunakan **3D Designs** dilakukan dengan memanfaatkan *shapes* yang tersedia di Tinkercad dengan berbagai bentuk yang dapat diatur langsung. Sedangkan, pada **Codeblocks**, pembuatan objek dilakukan dengan menyusun kode yang tersedia pada blok-blok di Tinkercad seperti menyusun sebuah kode program. Kelebihan menggunakan Codeblocks ini adalah dapat disimulasikan setiap tahapan pembuatannya. Gambar 2.2 memperlihatkan perbedaan **3D Designs** dan **Codeblocks**.



(a)

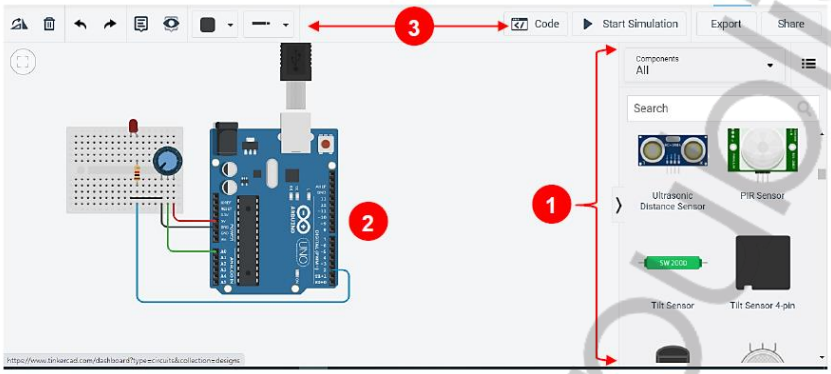


(b)

Gambar 2.2. Contoh desain 3D menggunakan
(a) 3D Designs dan (b) Codeblocks

Selain menyediakan menu untuk kebutuhan desain objek 3 dimensi, Tinkercad juga menyediakan menu **Circuits** untuk melakukan simulasi rangkaian elektronika, yang mana akan menjadi fokus pada buku ini. Menu **Circuits** dapat dijadikan sebagai media simulasi Mikrokontroler karena di dalamnya sudah tersedia Arduino Uno yang dapat diprogram menggunakan kode program yang sama persis dengan Arduino IDE. Tak hanya itu, Tinkercad juga telah menyediakan *serial monitor* untuk membaca

nilai serial. Contoh tampilan pada menu Circuits dapat dilihat pada Gambar 2.3.



Gambar 2.3. Contoh rangkaian sederhana di menu Circuits

Bagian-bagian pada Gambar 2.3 secara garis besar dapat dijelaskan sebagai berikut:

1) ***Components***

Bagian ini berisi beberapa komponen elektronika yang dapat diakses melalui proses *drag and drop*.

2) **Tempat Membuat Rangkaian**

Merupakan *space* yang disediakan Tinkercad untuk membuat rangkaian dengan menggunakan komponen yang tersedia. Rangkaian yang telah dibuat dapat digeser atau di-*zoom* menggunakan *mouse*.

3) **Toolbar**

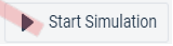
- **Rotate**

  Berfungsi memutar komponen searah jarum jam

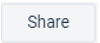
- **Delete**

  Berfungsi menghapus komponen

- **Undo/Redo** 
  Berfungsi untuk membatalkan eksekusi/perintah saat pembuatan rangkaian dan sebaliknya.
- **Notes tool** 
  Digunakan untuk membuat catatan pada komponen atau rangkaian.
- **Toggle notes visibility** 
  Berfungsi untuk menampilkan atau tidak menampilkan notes.
- **Wire color** 
  Berfungsi mengatur warna kabel/kawat
- **Wire type**
  Berfungsi mengatur jenis kabel/kawat
- **Toggle code editor** 

  Berfungsi menyediakan area penulisan *sketch* seperti yang dimiliki oleh Arduino IDE. Selain itu juga disediakan serial monitor untuk melihat data serial dari rangkaian.
- **Start/stop simulation** 

  Berfungsi memulai dan menghentikan simulasi rangkaian. Tombol ini akan berfungsi hanya jika *sketch* yang dibuat sudah benar, artinya dapat digunakan sebagai tombol verifikasi seperti pada Arduino IDE.
- **Export file** 

  Digunakan jika ingin membuat PCB dari rangkaian. Ekstensi *file* yang dihasilkan adalah .brd yang dapat dibuka melalui *software* Eagle.
- **Share this design** 

  Project simulasi yang telah dibuat dapat dibagikan melalui *link* tertentu.

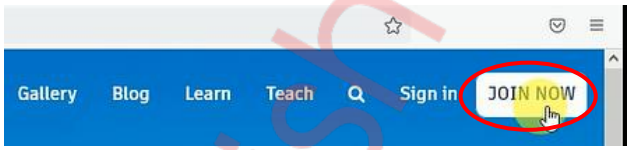
2.2. REGISTRASI, PEMBUATAN CLASS, DAN JOIN CLASS DI TINKERCAD

Sebelum menggunakan Tinkercad, para pengguna harus mengetahui bagaimana cara masuk ke *platform* Tinkercad baik melalui tahapan registrasi ataupun melalui **class**.

a. Registrasi/Pembuatan Akun di Tinkercad

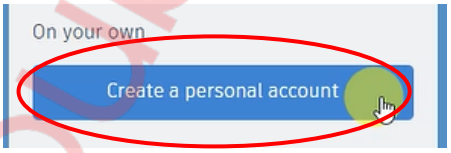
Tahapan registrasi penting untuk diketahui oleh mahasiswa, para pendidik (guru/dosen), asisten praktikum, ketua kelompok *project* tertentu, atau siapa saja yang ingin memanfaatkan Tinkercad secara pribadi. Karena ketika seorang pengguna sudah memiliki akun, maka ia dapat menentukan apakah ingin menjadi *students*, *teacher/educator*, atau yang lainnya sesuai kebutuhan. Berikut akan dijelaskan tahapan registrasi atau pembuatan akun di Tinkercad.

1) *Platform* Tinkercad bersifat *open source* yang dapat diakses melalui https://tinkercad.com/
2) Pada bagian menu pilih **Join Now.**



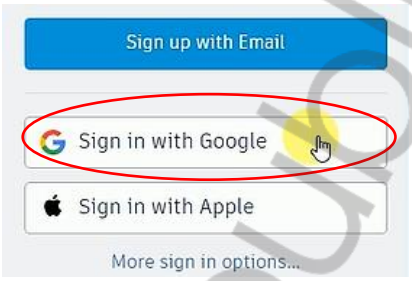
Gambar 2.4. Menu pada tampilan awal Tinkercad

3) Pada pilihan **Start Tinkering**, pilih **Create a personal account.**



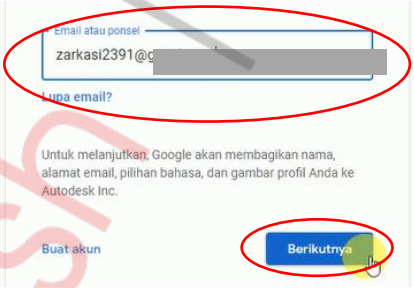
Gambar 2.5. Create a personal account di Tinkercad

4) Pilih metode pendaftaran yang diinginkan. Contoh jika ingin mendaftar menggunakan akun gmail, maka pilih **Sign in with Google**.



Gambar 2.6. Pilihan mode untuk pendaftaran Tinkercad

5) Masukkan alamat email, lalu klik **Berikutnya**.

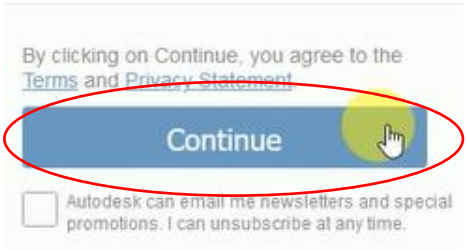


Gambar 2.7. Memasukkan alamat email

6) Masukkan *password* kemudian tekan **Berikutnya**.

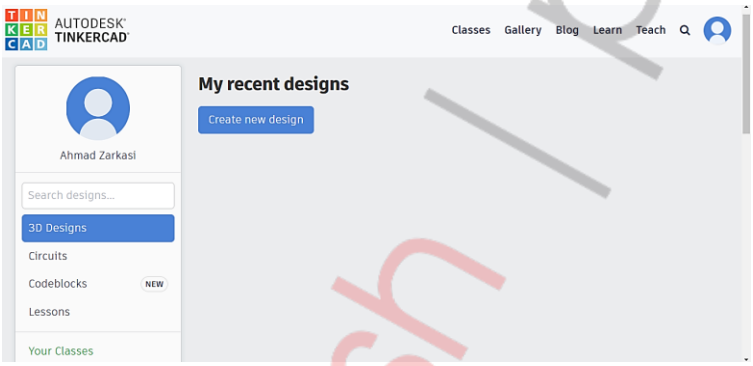
Gambar 2.8. Memasukkan *password*

7) Jika muncul tampilan seperti ini, klik **Continue**.



Gambar 2.9. Konfirmasi promosi dari Autodesk

8) Jika sudah berhasil maka akan muncul tampilan seperti ini.



Gambar 2.10. Tampilan awal pada akun Tinkercad

Setelah melakukan registrasi, maka untuk masuk ke akun Tinkercad dapat dilakukan dengan mengikuti langkah 1) sampai 4) atau bisa juga melalui menu **sign in**.

b. Pembuatan Classes

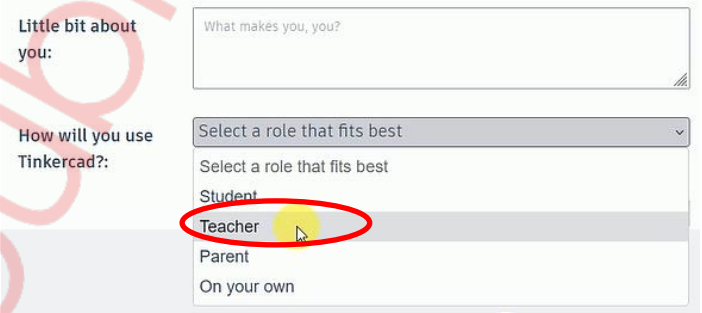
Tinkercad menyediakan fitur berupa **Classes/Class** yang dapat dimanfaatkan untuk kegiatan pembelajaran secara *online* layaknya Google Classroom, Edmodo, dan Schoology. Fitur ini dapat

dimanfaatkan untuk keperluan penugasan yang dilakukan di Tinkercad baik oleh dosen, guru, asisten praktikum, atau mungkin ketua kelompok suatu *project*. Seorang ketua kelompok dapat dengan mudah melakukan monitoring hasil pekerjaan anggotanya melalui akun yang ia miliki. Penggunaan fitur Classes juga memungkinkan anggota tim untuk dapat mengerjakan tugas atau membuat *project* tanpa perlu membuat akun di Tinkercad. Cukup dengan **class code** dan **nickname** yang dikirimkan oleh ketua kelompok, anggota tim dapat dengan mudah mengakses Tinkercad. Berikut tahapan pembuatan class di Tinkercad.

1) Ubah status menjadi **Teacher**, dengan cara:
   - Masuk ke akun Tinkercad. Kemudian klik profil.

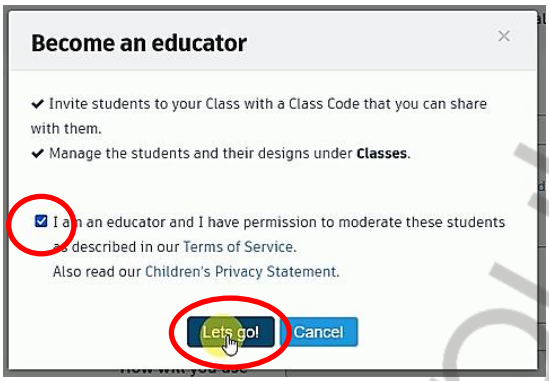
Gambar 2.11. Klik profil

   - Pada **Account Settings** di bagian paling bawah, klik pilihan **Select a role that fits best**, kemudian pilih **Teacher.**



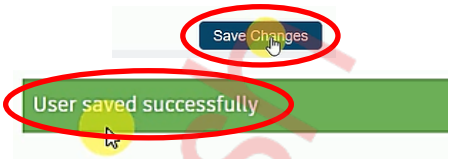
Gambar 2.12. Select role

- Centang kotak **Become an educator**, kemudian klik **Lets go**! (Gambar 2.13).



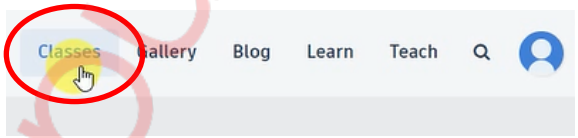
Gambar 2.13. Become an educator

- Klik **Save changes** sampai muncul keterangan **User saved successfully.**



Gambar 2.14. Keterangan berhasil mengubah status

2) Setelah menjadi **Teacher/Educator**, barulah dapat dilakukan pembuatan **Class**, dengan cara:
   - Kembali ke *dashboard* akun, kemudian klik **Classes**.

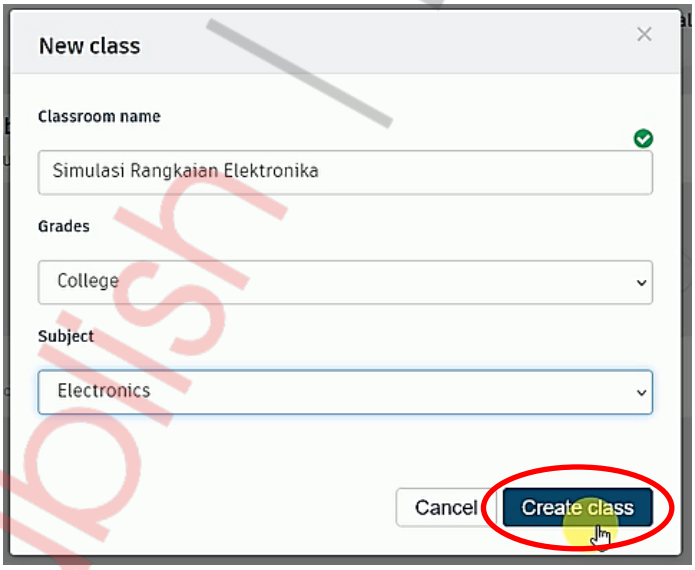


Gambar 2.15. Fitur Classes

- Klik **Create new class**.

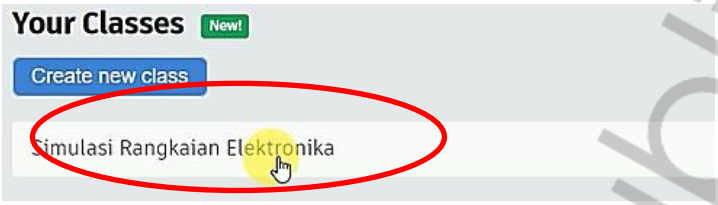
Gambar 2.16. Create new classes

- Pada **New class**, isi **Classroom name**, **Grades**, dan **Subject**, kemudian klik **Create class**. Berikut adalah contoh class.



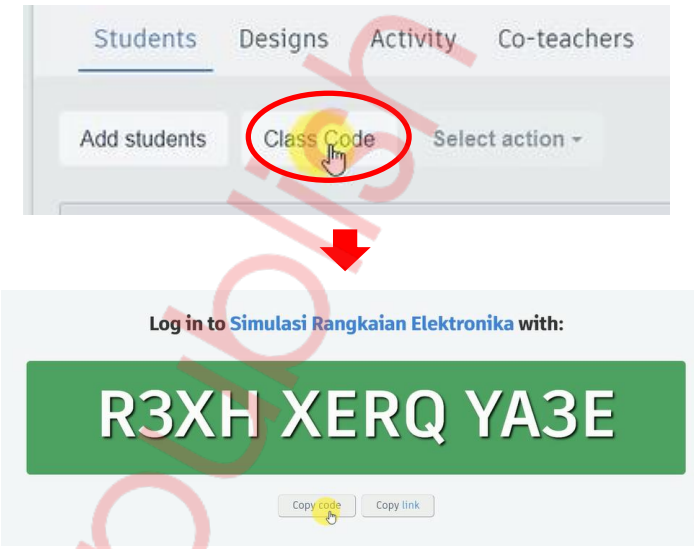
Gambar 2.17. Contoh class

- Jika sudah membuat class maka akan muncul tampilan seperti ini.



Gambar 2.18. Contoh class yang sudah dibuat

- Setelah class selesai dibuat, maka akan muncul class *code* yang dapat digunakan oleh mahasiswa/*member* untuk mengakses Tinkercad. Untuk melihat class code, klik nama class (Gambar 2.18), kemudian klik **class code**, maka akan muncul kode unik yang dapat di-*copy*.

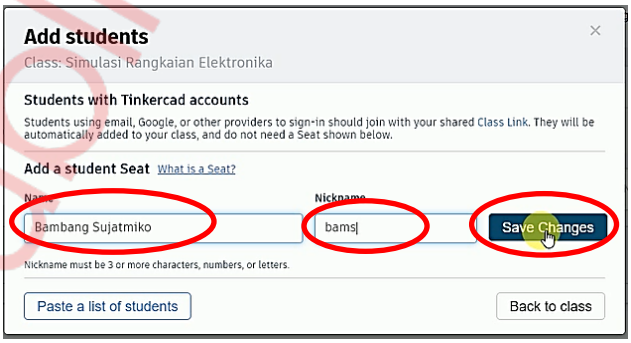


Gambar 2.19. Contoh class code

3) Selain menggunakan class code, mahasiswa/*member* suatu class harus memiliki **nickname** untuk dapat bergabung di suatu kelas. Nickname dibuat oleh *Teacher/Educator* dengan cara:
   - Klik nama class seperti Gambar 2.18, kemudian pilih **Add students**.

Gambar 2.20. Add students

   - Pada kotak **Add students**, masukkan nama dan *nickname* pada Kolom **Name** dan **Nickname**. Kemudian klik **Save Changes.** Untuk diketahui jika terdapat banyak nama yang ingin dimasukkan, maka tidak perlu diketik satu per satu melainkan bisa di-*copy* dari *file* lain kemudian di-*paste* ke kolom Name. Untuk nickname biasanya terisi secara otomatis, namun *Teacher/Educator* dapat mengubah sesuai keinginan seperti Gambar 2.21 di bawah.

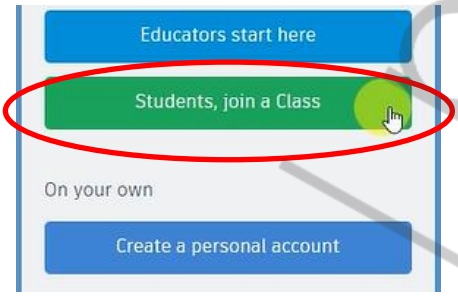


Gambar 2.21. Contoh pengisian Name dan Nickname

c. Join class

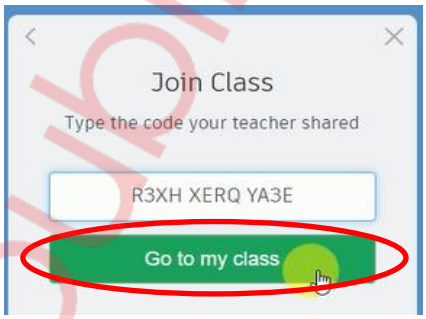
Setelah *Teacher/Educator* memberikan class code dan nickname, maka anggota dari suatu class dapat menggunakannya untuk *join* langsung ke class bahkan tanpa harus melakukan registrasi sebelumnya. Adapun cara *join* di class adalah sebagai berikut:

1) Masuk ke *link* https://tinkercad.com/ kemudian pada menu awal Tinkercad klik **Join Now** (seperti Gambar 2.4). Selanjutnya pilih **Students, join a Class**.



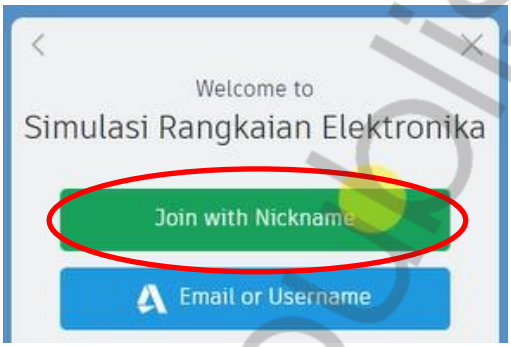
Gambar 2.22. Join class di Tinkercad

2) Masukkan class code. Contoh gunakan class code sesuai Gambar 2.19 kemudian klik **Go to my class.**



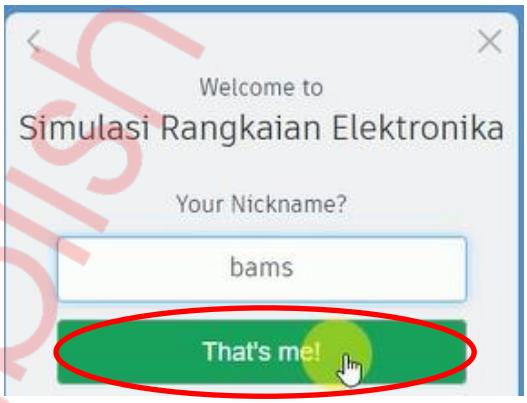
Gambar 2.23. Memasukkan class code

3) Setelah memasukkan class code akan muncul nama class. Selanjutnya pilih **Join with Nickname**.



Gambar 2.24. Muncul nama class

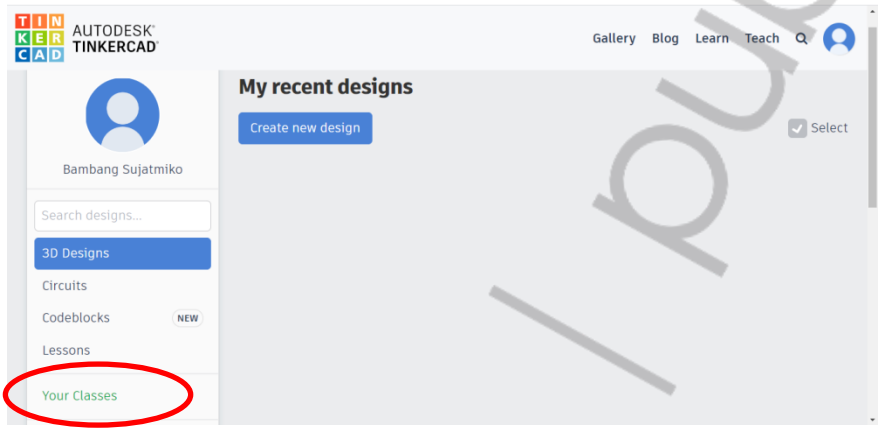
4) Masukkan nickname ke kolom **Your Nickname** (contoh gunakan nickname sesuai Gambar 2.21), kemudian klik **That's me**.



Gambar 2.25. Masukkan nickname

5) Jika sudah maka anggota class sudah dapat menggunakan Tinkercad. Hasil pengerjaan anggota class dapat dipantau

langsung melalui akun *Teacher/Educator* yang membuat class. Gambar 2.26 memperlihatkan tampilan Tinkercad yang diakses melalui *join class*. Jika diperhatikan tidak ada yang berbeda dengan tampilan yang dimiliki oleh *Tecaher/Educator* selain ketiadaan menu **Classes**. Anggota class dapat melihat nama class dan *Teacher/Educator* pada bagian **Your Classes**.

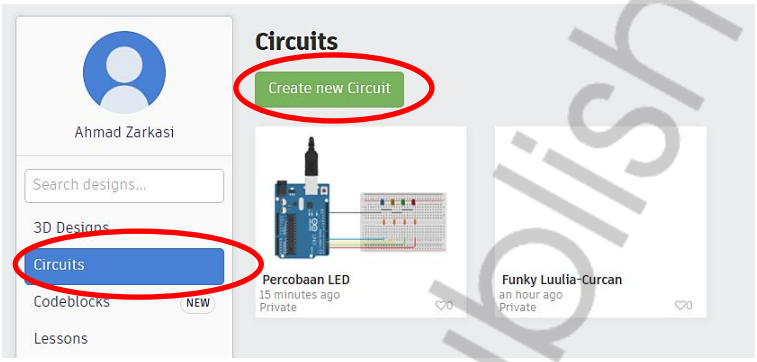


Gambar 2.22. Tampilan Tinkercad melalui *join class*

2.3. PERCOBAAN ARDUINO DI TINKERCAD

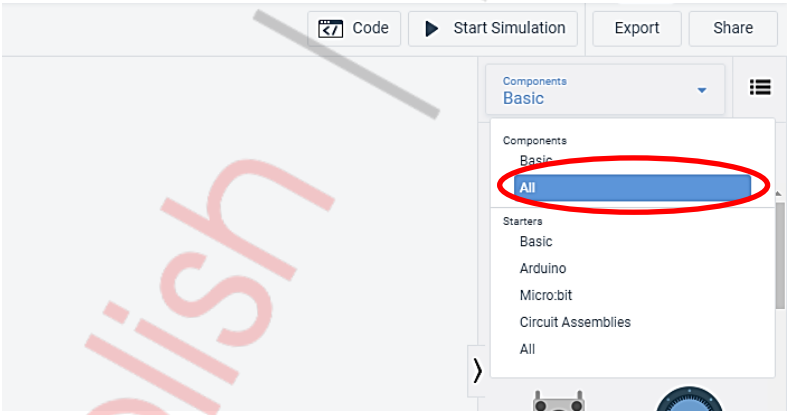
Pada bagian ini akan dijelaskan mengenai tahapan pembuatan simulasi berbasis Mikrokontroler Arduino melalui Tinkercad. Percobaan yang akan dilakukan berupa percobaan sederhana menggunakan 4 buah LED dengan tahapan sebagai berikut:

a. Masuk ke akun Tinkercad yang sudah dibuat atau melalui *join class* menggunakan class code dan nickname. Selanjutnya pilih menu **Circuits** dan klik **Create new Circuit**.



Gambar 2.23. Create new Circuit

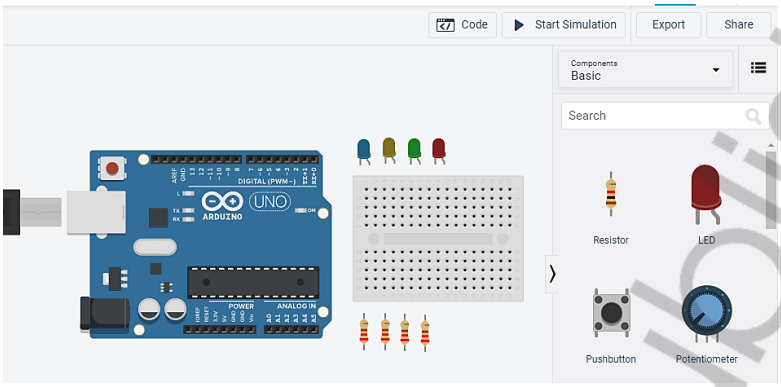
b. Klik bagian **Components** kemudian pilih **All**. Hal ini bertujuan untuk dapat melihat semua pilihan komponen yang tersedia.



Gambar 2.24. Components

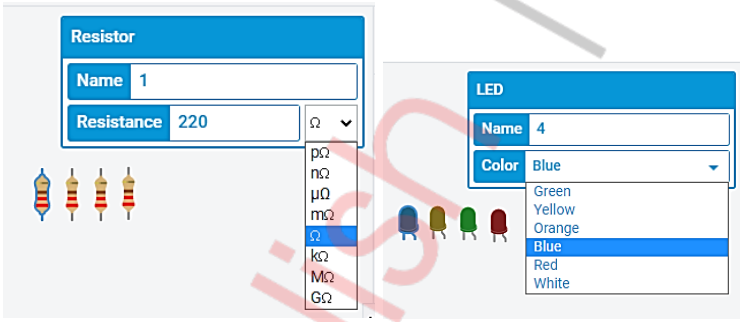
c. Siapkan komponen yang diperlukan cara drag and drop. Adapun komponen yang diperlukan berupa:

1) 1 unit *board* Arduino R3
2) 1 buah bread *board*
3) 4 buah LED
4) 4 buah resistor 220 Ohm



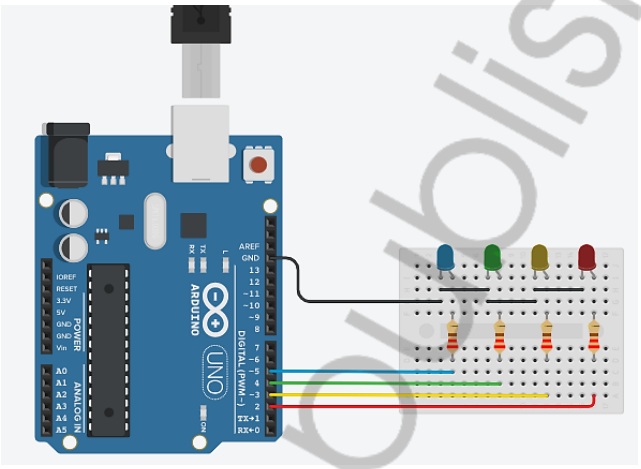
Gambar 2.25. Komponen yang diperlukan untuk simulasi

d. Untuk mengubah nilai resistor maupun warna LED dapat dilakukan melalui **properties** yang tersedia pada masing-masing komponen.



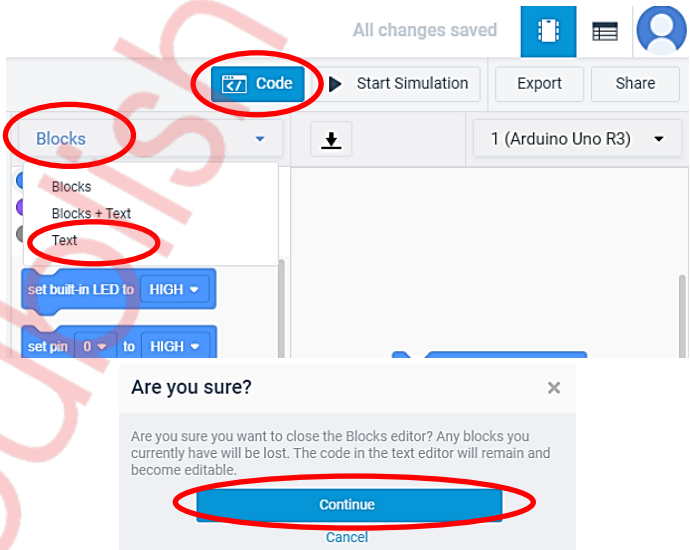
Gambar 2.26. Mengubah nilai resistor dan warna LED

e. Susun rangkaian berdasarkan Gambar 2.31 berikut.



Gambar 2.27. Rangkaian simulasi

f. Klik **Code > Blocks** > **Text,** lalu klik **Continue.**

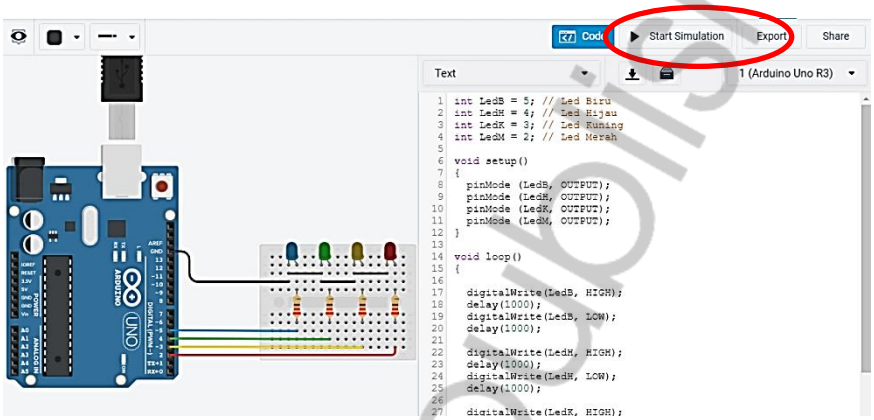


Gambar 2.28. Memunculkan text editor

g. Pada **Text** ketikkan *sketch* berikut:

```
int LedB = 5;//Led Biru
int LedH = 4;//Led Hijau
int LedK = 3;//Led Kuning
int LedM = 2;//Led Merah


void setup()
{
pinMode (LedB, OUTPUT);
pinMode (LedH, OUTPUT);
pinMode (LedK, OUTPUT);
pinMode (LedM, OUTPUT);
}


void loop()
{
digitalWrite(LedB, HIGH);
delay(1000);
digitalWrite(LedB, LOW);
delay(1000);

digitalWrite(LedH, HIGH);
delay(1000);
digitalWrite(LedH, LOW);
delay(1000);

digitalWrite(LedK, HIGH);
delay(1000);
digitalWrite(LedK, LOW);
delay(1000);


digitalWrite(LedM, HIGH);
delay(1000);
digitalWrite(LedM, LOW);
delay(1000);

}
```

h. Program siap dijalankan melalui **Start Simulation**.



Gambar 2.29. Start Simulation

●●●

BAB III
SYNTAX DASAR ARDUINO

Bahasa pemrograman yang digunakan untuk membuat *sketch* (program) pada Arduino adalah bahasa C++ namun dengan versi yang jauh lebih sederhana dan disesuaikan dengan kinerja hardware. Berikut beberapa instruksi dan *syntax* dasar dalam memrogram Arduino.

3.1. STRUCTURE (STRUKTUR)

Struktur dasar bahasa pemrograman Arduino cukup sederhana dan berjalan setidaknya dengan memuat dua bagian utama yaitu **setup** dan **loop**.

```
void setup()
{
 statements;//Pernyataan
}

void loop()
{
 statements; //Pernyataan
}
```

Blok **setup( )** berisi persiapan program yang dieksekusi hanya sekali, sedangkan **loop( )** berisi perintah yang dieksekusi secara berulang-ulang. Kedua bagian ini diperlukan agar program dapat berjalan. Fungsi **setup( )** biasanya berisi deklarasi variabel, pengaturan **pinMode**, dan inisialisasi komunikasi serial. Sedangkan, fungsi **loop( )** berisi kode yang dieksekusi secara terus

menerus dan merupakan inti dari semua program Arduino karena sebagian besar program dieksekusi melalui fungsi ini.

a. Setup( )

Fungsi **setup( )** dipanggil sekali ketika program dijalankan. Biasanya **setup( )** digunakan untuk mengatur mode dan komunikasi serial. Fungsi ini harus diikutkan meskipun tidak mengandung pernyataan/perintah apapun.

```
void setup()
{
 pinMode (pin, OUTPUT);//Mengatur pin
                       //sebagai output
}
```

b. Loop()

Setelah memanggil fungsi **setup()**, fungsi **loop()** melakukan eksekusi secara terus-menerus dan dimulai dari program yang paling atas ke program yang paling bawah.

```
void loop()
{
 digitalWrite (pin, HIGH);//Memberi tegangan
                          //5V ke pin
 delay (1000); //Berhenti 1 detik

 digitalWrite (pin, LOW);//Memberi tegangan
                         //0V pin
 delay (1000); //Berhenti 1 detik
}
```

c. Functions (Fungsi)

Fungsi adalah kumpulan baris program yang dibuat untuk mengerjakan tugas tertentu. Pada dasarnya, fungsi dibuat untuk mengurangi penulisan kode program yang berulang. Berikut adalah struktur dari sebuah fungsi:

```
type function_name (parameters)
{
 body_of_the_function;
}
```

- **type**: Berupa tipe data seperti **int**, **float**, **bool**, dan lain-lain
- **function_name**: nama dari sebuah fungsi yang tidak boleh sama dengan *syntax* yang tersedia di Arduino IDE
- **parameter**: untuk memberikan nilai pada sebuah fungsi
- **function body**: merupakan tubuh dari fungsi tempat menulis baris-baris perintah yang ingin digunakan

Berikut contoh sebuah fungsi untuk menjumlahkan dua buah variabel. Fungsi **jumlahkan** menerima dua buah nilai berupa **num1** dan **num2** melalui **parameter**.

```
int jumlahkan (int num1, int num2){
{
 int hasil = num1 + num2;
 return hasil;
}
```

Agar dapat digunakan, suatu fungsi harus dipanggil. Misal:

```
jumlahkan (23, 91);
```

Untuk lebih memahami penggunaan fungsi, perhatikan contoh lain berikut ini.

```
int nilaiDelay()
{
 int v; //Membuat variabel 'v'
 v = analogRead (pot); //Membaca nilai
                          //potensiometer
 v/= 4; //Konversi 0-1023 menjadi 0-255
 return; //Mengembalikan nilai ke program
}
```

Fungsi di atas merupakan fungsi konversi angka 0-1023 menjadi 0-255 yang digunakan untuk mengatur delay dalam program dengan membaca nilai potensiometer. Pertama digunakan sebuah tipe data integer dengan nama fungsi **nilaiDelay**. Sebuah variabel **v** digunakan untuk membaca nilai analog dari potensiometer. Nilai potensiometer yang berkisar antara 0-1023 kemudian dibagi 4 untuk memperoleh nilai akhir antara 0-255. Terakhir, nilai tersebut dikembalikan ke program utama.

d. Curly Brackets (kurung kurawal) { }

Kurung kurawal mendefinisikan awal dan akhir blok suatu fungsi atau pernyataan seperti fungsi **void loop()**, **for**, **if**, dan lain-lain. Namun, pada kenyataannya, suatu fungsi bisa saja tidak mengandung pernyataan sama sekali. Walaupun demikian, fungsi yang tidak mengandung pernyataan sekalipun harus menggunakan kurung kurawal.

```
type function ()
{
 statements;
}
```

Kurung kurawal buka '{' harus diikuti oleh kurung kurawal tutup '}', jika tidak maka akan menyebabkan terjadinya error pada saat melakukan *compile*.

e. Semicolon (Titik Koma);

Tanda titik koma harus digunakan untuk mengakhiri pernyataan dan memisahkan elemen program. Titik koma juga digunakan untuk memisahkan elemen dalam *for loop*.

```
int x = 13;//Mendeklarasikan variabel x
          //sebagai integer 13
```

Lupa dalam mengakhiri baris dengan titik koma akan menyebabkan kesalahan pada saat melakukan *compile*. Biasanya kesalahan semacam ini dapat dideteksi melalui keterangan yang muncul pada konsol *text editor* di Arduino IDE.

f. Block Comments (Komentar Blok) /*…*/

Komentar blok disebut juga komentar *multi line* merupakan area teks yang diabaikan oleh program. Komentar blok dipergunakan untuk mendeskripsikan suatu program sehingga memudahkan orang lain dalam memahami sebuah program. Penggunaan komentar blok ini memungkinkan penulisan banyak komentar (terdiri dari banyak baris).

```
/*
Ini merupakan contoh block comments yang tidak dieksekusi
namun sangat bermanfaat untuk memahami sebuah program
*/
```

g. Line Comments (Komentar Baris) //

Komentar baris dimulai dengan tanda '//' dan diakhiri dengan kode berikutnya. Seperti komentar blok, komentar baris juga

diabaikan oleh program dan tidak memakan ruang memori. Komentar baris hanya digunakan untuk membuat keterangan yang hanya terdiri dari satu baris. Komentar baris biasanya dimanfaatkan untuk memberi keterangan suatu baris program agar mudah diingat.

```
//Ini adalah contoh komentar dalam satu baris
```

3.2. VARIABLES (VARIABEL)

Variabel adalah kode program yang digunakan untuk menyimpan suatu nilai pada sebuah nama. Seperti namanya, variabel adalah angka yang nilainya dapat diubah. Untuk menggunakannya, sebuah variabel perlu dideklarasikan dan secara opsional ditetapkan ke nilai yang perlu disimpan. Kode berikut mendeklarasikan variabel yang disebut **inputVariable** dan memberikan nilai yang diperoleh pada pin analog 2.

```
int inputVariable = 0; //Mendeklarasikan variabel
                       //dan memberikan nilai 0
inputVariable = analogRead(2);//Mengatur var ke nilai
 //pin Analog 2
```

inputVariable merupakan nama variabel. Baris pertama menyatakan bahwa variabel tersebut akan berisi **int** (integer). Baris kedua menetapkan variabel ke nilai pada pin analog 2. Hal ini membuat nilai pada pin 2 dapat diakses di tempat lain dalam kode program.

Setelah suatu variabel ditetapkan, dapat dilakukan uji nilai untuk melihat apakah variabel tersebut memenuhi kondisi tertentu atau tidak. Di samping itu, nilai yang ada di dalam variabel juga dapat digunakan secara langsung. Berikut adalah contoh kode program untuk mengatur *delay* berdasarkan nilai sebuah variabel.

```
if (inputVariable < 100)//Menguji variabel jika kurang
                  //dari 100
{
inputVariable = 100;//Jika benar maka berikan nilai 100
}
delay(inputVariable);//Menggunakan variabel sebagai
 //delay
```

Jika nilai **inputVariabel** kurang dari 100 maka akan diatur agar memiliki nilai 100. Jika **inputVariable** sudah memiliki nilai 100, maka nilai tersebut akan digunakan untuk mengatur waktu *delay*.

a. Variabel Declaration (Deklarasi Variabel)

Semua variabel harus dideklarasikan sebelum dapat digunakan. Mendeklarasikan variabel berarti mendefinisikan tipe nilainya seperti dalam int, long, float, byte, dan lain-lain. Deklarasi variabel juga dilakukan untuk menetapkan nama tertentu dan secara opsional menetapkan nilai awalnya. Pada program, deklarasi variabel hanya perlu dilakukan sekali saja. Contoh berikut menyatakan bahwa **inputVariable** menggunakan tipe data int (*integer* atau bilangan bulat) yang nilainya sama dengan nol. Hal ini dinamakan *assignment* sederhana.

```
int inputVariable = 0;
```

b. Variable Scope (Cakupan Variabel)

Variabel dapat dideklarasikan pada tiga lokasi yang berbeda. Pertama, di bagian *header* atau di awal program sebelum **void setup( )**, variabel ini disebut *global variable*. Kedua, di dalam fungsi atau blok yang disebut *local variable*. Ketiga, di dalam definisi parameter fungsi yang disebut dengan *formal parameters*. Contoh berikut menunjukkan cara mendeklarasikan beberapa jenis variabel yang berbeda dan menunjukkan visibilitas setiap variabel.

```
int value;//variabel 'value' berlaku pada semua
 //fungsi
void setup()
{
 //Dapat dikosongkan
}
void loop()
{
 for (int i=0; i<20;)//'i' hanya berlaku di
 //dalam for loop
 {
 i++;
 }
 float f;//'f' hanya berlaku dalam loop
}
```

3.3. DATA TYPES (TIPE DATA)

Tipe data adalah klasifikasi jenis data yang diperuntukkan agar *compiler* dapat mengetahui bagaimana sebuah data akan digunakan.

a. Byte

Byte menyimpan nilai numerik sebesar 8-bit tanpa titik desimal dengan jangkauan 0-255.

```
byte variabel1 = 180;//Deklarasi 'variabel1'
 //dengan tipe byte
```

b. Integers (int)

Integer atau bilangan bulat adalah tipe data utama untuk penyimpanan angka tanpa titik desimal dengan ukuran 16-bit. Nilai tipe data ini berkisar antara-32.768 hingga 32.767.

```
int variabel2 = 10;//Deklarasi 'variabel2'
 //dengan tipe integer
```

c. Long

Long merupakan tipe data integer dengan ukuran yang diperluas menjadi 32-bit. Nilai long berkisar antara-2.147.483.648 sampai 2.147.483.647.

```
long variabel3 = 900000;//Deklarasi 'variabel3'
                  //dengan tipe long
```

d. Float

Tipe data ini digunakan untuk angka-angka desimal. Tipe data float memiliki resolusi yang lebih tinggi dari pada integer yaitu berukuran 32-bit dengan kisaran nilai -3,40282235E+38 sampai 3,40282235E+38

```
float variabel4 = 3.14;   //Deklarasi 'variabel4'
                 //dengan tipe float
```

e. Array

Array adalah kumpulan nilai yang diakses dengan nomor indeks. Nilai apapun dalam array dapat dipanggil dengan memanggil nama array dan indeks nilainya. Indeks pada array dimulai dari 0. Array perlu dideklarasikan dan diberi nilai secara opsional sebelum digunakan.

```
int arrayKu [ ] = {nilai0, nilai1, nilai2...}
```

Demikian juga dimungkinkan untuk mendeklarasikan array dengan mendeklarasikan tipe dan ukuran array serta menetapkan posisi indeks.

```
int arrayKu[5];//Deklarasi array bertipe integer
          //pada posisi ke-6

arrayKu[3]=10; //Memberikan indeks 4 ke nilai 10
```

Array sering digunakan dalam *for loop* di mana penghitung kenaikan (*increment counter*) juga digunakan sebagai posisi indeks setiap nilai array. Contoh berikut menggunakan array untuk mengedipkan 4 buah LED secara bergantian.

```
int led[4]={5, 4, 3, 2};//Array led memiliki
                  //indeks 0, 1, 2, dan 3

void setup(){
 for (int i=0; i<=3; i++){//Mengatur indeks led
                  //yang akan dijadikan
 pinMode(led[i], OUTPUT);//sebagai output
 }
}


void loop(){
 for (int i=0; i<=3; i++){ //Mengedipkan led[0]
 //sampai led[3]
 digitalWrite(led[i], HIGH);
 delay(1000);
 digitalWrite(led[i], LOW);
 delay(1000);       //delay selama 1 detik
 }
}
```

Selain ke-5 tipe data di atas, masih banyak lagi tipe data lain seperti yang diperlihatkan Tabel 3.1.

Tabel 3.1. Jenis Tipe Data

Tipe data	Ukuran	Rentang Nilai
boolean	8 bit (1 byte)	1 atau 0 (True atau False)
byte	8 bit	0 sampai 255
char	8 bit	-128 sampai 127
unsigned char	8 bit	0 sampai 255

Tipe data	Ukuran	Rentang Nilai
int	16 bit	-32.768 sampai 32.767
unsigned int	16 bit	0 sampai 65.535
word	16 bit	0 sampai 65.535
long	32 bit	-2.147.483.648 sampai 2.147.483.647
float	32 bit	-3,4028235E+38 sampai 3,4028235E+38
double	64 bit	-3,4028235E+38 sampai 3,4028235E+38
string	1 byte + x	Array dari char
array	8 bit + x	Kumpulan variabel

3.4. OPERATOR

a. Arithmetic Operators (Operator Aritmatika)

Tabel 3.2 memperlihatkan beberapa jenis operator aritmatika beserta fungsinya.

Tabel 3.2. Operator Aritmatika

Operator	Nama Operator	Fungsi
+	Addition (penjumlahan)	Melakukan penjumlahan
–	Subtraction	Melakukan pengurangan
*	Multiplication	Melakukan perkalian
/	Division	Melakukan pembagian
%	Modulo	Mencari sisa hasil bagi
=	Assignment operator	Menyimpan nilai yang ada di sebelah kanan tanda sama dengan dalam variabel yang ada di sebelah kiri tanda sama dengan

Berikut contoh penulisan program menggunakan operator aritmetika.

```
int x = 9;//deklarasi variabel 'x' bertipe integer
int y = 4;//deklarasi variabel 'y' bertipe integer
int z;

z = x + y; //menghasilkan 13
z = x - y; //menghasilkan 5
z = x * y; //menghasilkan 36
z = x/y; //menghasilkan 2
z = x % y; //menghasilkan 1
```

Ingat bahwa operasi dilakukan menggunakan tipe data operan. Jadi untuk x/y atau 9/4 menghasilkan 2 dan bukan 2.25 karena 9 dan 4 adalah integer dan tidak dapat menggunakan titik desimal.

b. Comparison Operator (Operator Pembanding)

Biasanya operator pembanding digunakan pada pernyataan ***if*** untuk membandingkan suatu variabel/konstanta terhadap variabel/konstanta yang lain. Operator pembanding sering digunakan untuk menguji suatu kondisi apakah bernilai **true (benar)** atau **false (salah).**

Tabel 3.3. Operator Pembanding

Operator	Nama Operator	Fungsi
==	Equal	Mengecek apakah nilai dua operan sama atau tidak, jika **sama** maka hasilnya **True**
! =	Not equal	Mengecek apakah nilai dua operan sama atau tidak, jika **tidak** sama maha hasilnya **true**
<	Less than	Mengecek apakah nilai operan di sebelah kiri bernilai kurang dari

Operator	Nama Operator	Fungsi
		operan sebelah kanan, jika iya maka bernilai **true**
>	Greater than	Mengecek apakah nilai operan di sebelah kiri bernilai lebih besar dari operan sebelah kanan, jika iya maka bernilai **true**
<=	Less than or equal	Mengecek apakah nilai operan di sebelah kiri bernilai kurang dari atau sama dengan operan sebelah kanan, jika iya maka bernilai **true**
>=	Greater than or equal	Mengecek apakah nilai operan di sebelah kiri bernilai lebih besar dari atau sama dengan operan sebelah kanan, jika iya maka bernilai **true**

Berikut contoh penulisan program menggunakan operator pembanding.

```
int a = 9;        //deklarasi variabel a

int b = 4;        //deklarasi variabel b

bool c;  //deklarasi variabel c dengan tipe Boolean


 if(a == b) //Statement if ini menghasilkan
 //false (0)
 c = true;
 else
 c = false;

 if(a != b) //Statement if ini menghasilkan
 //true (1)
 c = true;
 else
 c = false;
```

```
if(a < b) //Statement if ini menghasilkan
//false (0)
c = true;
else
c = false;

if(a > b)//Statement if ini menghasilkan true (1)
c = true;
else
c = false;

if(a <= b)//Statement if ini menghasilkan false (0)
c = true;
else
c = false;

if(a >= b)//Statement if ini menghasilkan true (1)
c = true;
else
c = false;
```

c. Logical Operators (Operator Logika)

Operator logika biasanya merupakan cara untuk membandingkan dua ekspresi sehingga nilai **true** atau **false** tergantung pada operatornya. Operator logika pada Arduino terdiri dari logika **AND**, **OR**, dan **NOT**.

Tabel 3.4. Operator Logika

Operator	Nama Operator	Fungsi
&&	AND	Bernilai **true** jika kedua ekspresi bernilai **true**
||	OR	Bernilai **true** jika salah satu bernilai **true**
!	NOT	Bernilai **true** jika ekspresi bernilai **false**

Berikut contoh program menggunakan operator logika.

```
int a = 9,
b = 4
bool c;

 if((a > b)&& (b >= a)) //Menghasilkan false
 c = true;
 else
 c = false;

 if((a == b)|| (b < a)) //Menghasilkan true
 c = true;
 else
 c = false;

 if( !(a == b)&& (b < a))//Menghasilkan true
 c = true; //Gabungan dari NOT dan
 //AND
 else
 c = false;
```

d. Compound Operators (Operator Majemuk)

Operator majemuk menggabungkan operasi aritmetika dengan sebuah *variable assignment*. Operator ini biasanya digunakan pada *for loop* yang secara umum diperlihatkan pada Tabel 3.3.

Tabel 3.5. Operator Majemuk

Operator	Nama Operator	Fungsi
++	Increment	Menambahkan suatu nilai dengan 1
−−	Decrement	Mengurangi suatu nilai dengan 1
+=	Compound addition	Merupakan bentuk ringkas dari operasi penjumlahan
−=	Compound subtraction	Merupakan bentuk ringkas dari operasi pengurangan
*=	Compound multiplication	Cara untuk melakukan perkalian suatu variabel dengan konstanta atau variabel lain

Operator	Nama Operator	Fungsi
/=	Compound division	Cara untuk melakukan pembagian suatu variabel dengan konstanta atau variabel lain
% =	Compound modulo	Cara mudah menghitung sisa pembagian
& =	Compound bitwise AND	Cara mudah menggunakan operator bitwise AND (bitwise **&** digunakan untuk mengkonversi nilai operan desimal menjadi bentuk biner dan melakukan operasi AND pada biner tersebut)
| =	Compound bitwise OR	Cara mudah menggunakan operator bitwise OR (penggunaan bitwise bitwise | sama dengan **&** akan tetapi | digunakan pada operasi OR)

Berikut contoh program menggunakan operator majemuk:

```
/* Deklarasi variabel a, b, dan c dalam bentuk integer*/
int a = 10;
int b = 20;
int c;

//Penggunaan operator majemuk
 a++; //sama seperti a = a+1, bernilai 11
 a--; //sama seperti a = 1-1, bernilai 9
 b += a; //sama seperti b = b+a, bernilai 30
 b-= a; //sama seperti b = b-a, bernilai 10
 b *= a; //sama seperti b = b*a, bernilai 300
 b/= a; //sama seperti b = b/a, bernilai 2
 b %= a; //sama seperti b = b%a, bernilai 0
 a |= b; //bernilai 0
 a &= b; //bernilai 0
```

3.5. CONSTANTS (KONSTANTA)

Bahasa pemrograman pada Arduino memiliki beberapa nilai yang telah didefinisikan yang disebut konstanta. Konstanta digunakan untuk membuat program lebih mudah dibaca. Konstanta diklasifikasikan ke dalam beberapa kelompok.

a. True/False

Merupakan konstanta Boolean yang mendefinisikan level logika. Konstanta **false** didefinisikan sebagai 0 (nol), sedangkan **true** didefinisikan sebagai 1 (satu) atau bisa apa saja kecuali 0. Jadi, dalam Boolean, nilai-1, 2, 200 dan lain-lain (kecuali 0) didefinisikan sebagai **true**. Berikut contoh penggunaannya.

```
if (b == true) //Syarat agar perintah dijalankan
{
 lakukanSesuatu; //Berisi perintah yang ingin
 //dijalankan
}
```

b. High/Low

Konstanta ini menentukan level pin Arduino sebagai **HIGH** atau **LOW** dan digunakan saat membaca atau menulis nilai pin digital. Konstanta **HIGH** didefinisikan sebagai level logika 1, ON, atau 5 Volt. Sedangkan, konstanta **LOW** adalah level logika 0, OFF, atau 0 Volt.

```
digitalWrite (13, HIGH);//Pin 13 Arduino diberi
 //logika 1 yang berarti
                  //diberi tegangan 5V
digitalWrite (12, LOW); //Pin 13 Arduino diberi
 //logika 0 yang berarti
                  //diberi tegangan 0V
```

c. Input/Output

Pada Arduino terdapat pin I/O (Input/Output) yang dapat digunakan untuk mengakses atau menerima data, baik berupa data digital maupun data analog. Untuk menentukan apakah pin tersebut digunakan sebagai input atau output, perlu dilakukan pengaturan mode pin menggunakan fungsi **pinMode** dan konstanta **INPUT** atau **OUTPUT**.

```
pinMode (A0, INPUT); //Pin A0 sebagai input
pinMode (2, OUTPUT); //Pin 2 sebagai output
```

3.6. FLOW CONTROL

a. if

Pernyataan **if** digunakan untuk menguji apakah suatu kondisi telah terpenuhi atau tidak yang nantinya akan menentukan keputusan dalam mengeksekusi suatu perintah/pernyataan. Jika pernyataannya benar, maka perintah atau pernyataan apapun yang ada di dalam kurung akan dieksekusi. Jika salah, maka program akan melewatkan pernyataan tersebut. Pernyataan **if** biasanya menggunakan operator pembanding. Berikut adalah *syntax* penulisan program menggunakan **if**.

```
if (sebuahVariabel ?? sebuahNilai)
{
 lakukanSesuatu;//Berisi pernyataan/perintah
}
```

Contoh di atas membandingkan **sebuahVariabel** dengan nilai lain yang dapat berupa variabel atau konstanta. Jika perbandingan atau kondisi dalam tanda kurung **( )** benar, pernyataan di dalam tanda kurung kurawal **{ }** akan dijalankan. Jika tidak, program akan

melewatinya. Contoh penggunaan **if** dapat dilihat pada contoh penggunaan operator pembanding dan operator logika.

b. if else

Penggunaan **if else** berfungsi untuk menjalankan salah satu perintah atau pernyataan berdasarkan kesesuaian hasil pengujian kondisi pada tanda kurung **( ) if**. Misalnya jika ingin menghidupkan dua buah LED secara bergantian yang dikendalikan menggunakan sebuah push button.

```
if (pushButton == HIGH)//Jika push button
 //ditekan (HIGH)
 {
 digitalWrite (ledHijau, HIGH);//Led hijau menyala
 digitalWrite (ledMerah, LOW);//Led merah mati
 }
 else//Program di bawah dijalankan jika pushButton
 //== LOW
{
 digitalWrite (ledHijau, LOW);//Led hijau mati
 digitalWrite (ledMerah, HIGH);//Led merah menyala
 }
```

Else juga dapat digunakan sebelum pengujian **if** yang lain (**else if**), sehingga beberapa pengujian yang eksklusif dapat dijalankan pada saat yang bersamaan. Selain itu, penggunaan **else if** memungkinkan percabangan pengujian dengan jumlah yang sangat banyak (secara teori tak terbatas). Ingat bahwa hanya satu perintah atau pernyataan yang akan dijalankan tergantung pada hasil pengujian kondisi.

```
if (inputPin < 500)//Pengujian kondisi 1
{
doThingA; //Dieksekusi bila kondisi 1 terpenuhi
}
else if (inputPin >= 1000)//Pengujian kondisi 2
{
doThingB; //Dieksekusi bila kondisi 1 tidak
 //terpenuhi dan kondisi 2 terpenuhi
else
{
doThingC; //Dieksekusi bila kondisi 1 dan 2 tidak
 //terpenuhi
}
```

c. for

Pernyataan **for** digunakan untuk mengulang suatu pernyataan yang berada dalam kurung kurawal { }. Perhitungan yang digunakan dapat berupa penambahan ataupun pengurangan suatu bilangan dengan batasan tertentu untuk mengakhiri proses looping. **For** banyak digunakan pada operasi yang berulang-ulang dengan kombinasi bilangan tertentu. Penggunaan **for** dapat mempersingkat *sketch* sehingga memperkecil ukuran suatu program. **For** memiliki 3 bagian yang dipisahkan oleh tanda titik koma (**;**). Berikut *syntax* penulisan program menggunakan **for**.

```
for (inisialisasi, kondisi, ekspresi/operasi)
{
 lakukanSesuatu;//Berisi pernyataan/perintah
}
```

Perhitungan variabel lokal dan *increment* dijalankan sekali dan paling awal. Setiap kali melalui *loop* dilakukan pengujian kondisi. Jika kondisi benar, pernyataan dan ekspresi akan dieksekusi dan pengujian kondisi kembali terjadi. Ketika kondisi salah maka *loop* akan berakhir. Contoh berikut memulai bilangan bulat **i** pada 0 dan melakukan pengujian pada nilai **i** apakah masih kurang dari 20. Jika benar, maka nilai **i** akan ditambah sebanyak 1 dan

mengeksekusi sebuah perintah untuk menghidupkan dan mematikan LED sebanyak 20 kali (**i=0** sampai **i=19**).

```
/*Deklarasi 'I' bertipe integer. Kemudian menguji nilai 'i'
apakah i<20? Kemudian lakukan increment senilai 1*/
for (int i=0; i<20; i++)
{
 digitalWrite(13, HIGH); //pin 13 ON
 delay(250); //berhenti 1/4 detik
 digitalWrite(13, LOW); //pin 13 OFF
 delay(250); //berhenti 1/4 detik
}
```

d. while

Sebuah perintah pada **while,** *loop* akan dieksekusi secara terus menerus dan tanpa henti sampai terdapat kondisi di dalam kurung yang bernilai salah. Berikut adalah *syntax* penulisan **while**.

```
while (kondisi ?? nilai)
{
lakukanSesuatu;//Berisi pernyataan/perintah
}
```

Contoh berikut ini menguji apakah **suatuVariabel** memiliki nilai kurang dari 200. Jika benar maka akan dilakukan *increment* sejumlah 1 secara terus menerus sampai batas < 200.

```
while (suatuVariabel < 200)//Mengecek apakah nilai
{                  //'variabelKu' kurang
 //dari 200
 suatuVariabel++;//increment yang dapat juga
 //ditulis:
          //suatuVariabel = suatuVariabel + 1
}
```

e. do-while

Pernyataan perulangan **do while** hampir sama dengan pernyataan **while**. Perbedaannya yaitu jika pada pernyataan **while**,

kondisi diuji dahulu dan bila uji kondisi bernilai benar maka pernyataan yang ada dalam blok **while** akan dieksekusi. Sedangkan, pada **do while** kondisi menjadi terbalik yaitu pernyataan utama akan dieksekusi terlebih dahulu, setelah itu baru dilakukan uji kondisi. Jika kondisi benar maka pernyataan utama akan diulang, tetapi jika salah program akan keluar dari blok **do while**. Berikut *syntax* penulisan **do while**.

```
do
{
lakukanSesuatu;
} while (suatuVariabel ?? nilai);
```

Di bawah ini adalah contoh penggunaan **do while**. Pertama-tama program akan melakukan *delay* selama 50 milidetik menunggu hingga sensor stabil. Selanjutnya program akan membaca sensor dan menyimpannya dalam variabel **x**. Kemudian, variabel **x** akan diuji apakah bernilai lebih kecil dari 100 atau tidak. Jika nilai **x** lebih kecil dari 100, maka program akan melakukan pengulangan (kembali membaca sensor), tetapi jika nilai **x** lebih dari 100, maka program akan keluar dari blok **do while** untuk melakukan perintah yang lain.

```
do {
delay (50);
x = readSensors();//memasukkan nilai sensor ke
 //variabel 'x'
} while (x < 100);//looping terjadi saat 'x' kurang
 //dari 100
```

3.7. DIGITAL I/O

a. pinMode (pin, mode)

Syntax **pinMode (pin, mode)** digunakan dalam **void setup()** untuk mengkonfigurasi pin tertentu agar berperan sebagai input atau output seperti contoh di bawah.

```
pinMode (13, OUTPUT);//Mengatur pin 13 sebagai
 //output
pinMode (3, INPUT); //Mengatur pin 3 sebagai
 //input
```

Pin digital pada Arduino secara *default* berperan sebagai input, sehingga pada dasarnya tidak perlu dideklarasikan secara eksplisit sebagai input dengan **pinMode.** Pin yang dikonfigurasi sebagai input dikatakan dalam keadaan impedansi tinggi.

b. digitalRead (pin)

Syntax **digitalRead (pin)** digunakan untuk membaca nilai dari pin digital tertentu dengan hasil **HIGH** atau **LOW**. Pin dapat ditentukan sebagai variabel atau konstanta.

```
/* Mengatur variabel 'nilai' agar sama dengan pin input*/
nilai = digitalRead (pin);
```

c. digitalWrite (pin, nilai)

Syntax **digitalWrite (pin, nilai)** berfungsi memberikan logika **HIGH** atau **LOW** yang berarti memberikan nilai 1 (5V) atau 0 (0V) pada pin Arduino. Berikut contoh kode program untuk membaca nilai pada *push button* yang terhubung ke input digital dan menyalakan LED yang terhubung ke output digital ketika *push button* ditekan.

```
int led = 13; //Menghubungkan LED ke pin 13
int pb = 7; //Menghubungkan push button ke pin 7
int nilaiPb = 0;//Variabel untuk menyimpan nilai Pb

void setup(){
pinMode(led, OUTPUT);//Mengatur led sebagai output
pinMode(pb, INPUT); //Mengatur push button sebagai input
}
void loop(){
nilaiPb = digitalRead(pb);//Membaca nilai pb dan
 //menyimpannya ke 'nilaiPb'
digitalWrite(led, nilai);//Agar 'led'sesuai dengan
 //'nilaiPb'
}
```

3.8. ANALOG I/O

a. analogRead (pin)

Syntax **analogRead (pin)** digunakan untuk membaca nilai dari pin analog Arduino dengan resolusi ADC tertentu sesuai spesifikasi *board* Arduino yang digunakan. Sebagai contoh, Arduino Uno memiliki resolusi ADC sebesar 10 bit yang dapat diakses pada pin A0 sampai A5. Nilai integer yang dihasilkan berkisar antara 0 sampai 1023.

```
value = analogRead (pin);//Membaca nilai analog pada
 //'pin' dan menyimpannya
 //dalam variabel 'value'
```

b. analogWrite (pin, nilai)

Syntax **analogWrite (pin, nilai)** berfungsi untuk menulis nilai *pseudo-analog* melalui pin PWM. Pada masing-masing *board* Arduino, jumlah pin yang memiliki fitur PWM berbeda-beda tergantung jenis *chip* yang digunakan. Contohnya pada Arduino yang menggunakan *chip* ATmega328, fungsi PWM dapat bekerja pada pin 3, 5, 6, 9, 10, dan 11. Sementara pada Arduino yang lebih lama dengan *chip* ATmega8 hanya mendukung fungsi PWM pada pin 9, 10, dan 11. Kebanyakan Arduino memiliki PWM dengan

resolusi sebesar 8 bit yang artinya dapat mengeluarkan nilai dari 0 sampai 255.

```
analogWrite (pin, value);//Menulis nilai analog
 //sebesar 'value' ke 'pin'
```

Contoh berikut membaca nilai analog dari potensiometer sebagai input analog, kemudian nilai tersebut diubah dengan cara membaginya dengan 4, dan terakhir mengeluarkan sinyal PWM pada pin PWM.

```
int led = 10;//Menghubungkan led ke pin 10
int pot = A0;//Potensiometer terhubung dengan pin 0
int nilaiPot; //Membaca nilai potensio

void setup(){}//Tidak dilakukan setup
void loop()
{
nilaiPot = analogRead(pot);//Membaca nilai
 //potensiometer

nilaiPot/= 4; //Konversi 0-1023
//menjadi 0-255

analogWrite(led, nilaiPot);//Output PWM ke led
 //(berdasarkan
 //nilai potensiometer)
}
```

3.9. TIME

a. delay (ms)

Syntax **delay (ms)** digunakan untuk memberikan jeda pada program untuk jumlah waktu yang ditentukan dalam satuan milidetik. Berikut contoh penulisan *delay* untuk jeda waktu selama 1 detik atau 1000 milidetik.

```
delay (1000);//Memberi jeda selama 1000 milidetik
          //atau 1 detik
```

b. millis ( )

Syntax **millis ( )** berguna untuk menjalankan waktu internal setiap milidetik pada Arduino secara independen. Ketika **millis** dibaca, maka **millis** akan terus menghitung waktu walaupun Arduino sedang menjalankan program yang lain. Berikut adalah contoh penulisan **millis**.

```
value = millis();//Mengatur 'value' sama dengan
 //nilai millis
```

3.10. MATH

a. min (x, y)

Syntax min (x, y) berguna untuk menghitung minimal dua angka dari tipe data apapun dan mengembalikan angka yang lebih kecil.

```
value = min(value, 100);//Mengatur 'value' menjadi
 //lebih kecil dari 'value'
 //atau 100 dan memastikan
 //'value' tidak akan di
                          //atas 100
```

b. max (x, y)

Syntax **man (x, y)** merupakan kebalikan dari **min (x,y)**. *Syntax* ini digunakan untuk menghitung maksimal dua angka dari tipe data apapun dan mengembalikan angka yang lebih besar

```
value = min(value, 100);//Mengatur 'value' menjadi
 //lebih besar dari 'value'
 //atau 100 dan memastikan
 //'value' tidak akan di
                          //bawah 100
```

3.11. SERIAL

a. Serial.begin (rate)

Pada Arduino terdapat fitur serial monitor dan serial plotter yang dapat dimanfaatkan untuk melihat hasil komunikasi serial berupa output suatu sensor, *push button*, dan lain-lain. *Syntax* **Serial.begin (rate)** digunakan untuk membuka *port* serial dan mengatur baud rate (kecepatan transmisi) untuk transmisi data serial. Baud rate tipikal untuk berkomunikasi dengan komputer adalah 9600 bps meskipun kecepatan lainnya masih tetap mendukung. Saat menggunakan komunikasi serial, pin digital 0 (RX) dan 1 (TX) tidak dapat digunakan secara bersamaan. Berikut contoh pengaturan baudrate.

```
void setup()
{
 Serial.begin(9600);//Membuka port serial dengan &
 //mengatur baudrate sebesar
 //9600 bps
}
```

b. Serial.print (data) dan Serial.println (data)

Syntax **Serial.print (data)** berfungsi mengirimkan data ke *port* serial dan menampilkannya pada serial monitor dalam satu baris saja. Jika argumen yang dimasukkan ke dalam perintah, maka data yang dikirim akan menyesuaikan dengan format tersebut.

```
Serial.print (91); //Mencetak "91"
Serial.print ("Hello World!")//Mencetak "Hello World"
Serial.print (variabel1) //Mencetak nilai pada
 //"variabel1"
Serial.print (91, BIN); //Mencetak "01011011"
Serial.print (91, HEX); //Mencetak "5B"
```

Syntax lain yang fungsinya hampir sama dengan **Serial.print (data)** adalah **Serial.println (data).** *Syntax* ini digunakan juga untuk

mengirimkan data yang selanjutnya ditampilkan di layar serial monitor. Jika dengan **Serial.print** data ditampilkan pada satu baris, maka dengan menggunakan **Serial.println** akan memunculkan baris baru.

```
Serial.println (variabel2)//Mencetak nilai pada
 //"variabel2"
```

Perlu diingat bahwa sebelum menggunakan *syntax* **Serial.print (data)**, *port* serial harus dibuka dan kecepatan transmisi data harus diatur menggunakan *syntax* **Serial.begin (rate)**.

●●●

BAB IV
PERCOBAAN MIKROKONTROLER ARDUINO

Percobaan-percobaan yang dimuat pada buku ini melibatkan komponen-komponen sederhana dan sangat mudah didapatkan. Pada buku ini terdapat 8 buah percobaan yang dimulai dari percobaan paling sederhana menggunakan LED, sampai pada pembuatan *project* dengan menggunakan Arduino Uno sebagai *board* Mikrokontrolernya. *Platform* Tinkercad berperan menyediakan sarana simulasi yang interaktif karena memiliki *interface* yang sangat mudah digunakan dengan tampilan yang hampir mirip dengan rangkaian aslinya. Capaian akhir yang diinginkan setelah melakukan percobaan-percobaan ini adalah agar pengguna dapat merancang sebuah *project* menggunakan mikrokontroler yang dapat diaplikasikan dalam kehidupan secara nyata. Pembuatan *project* yang dimaksud tentu tidak terbatas pada percobaan-percobaan yang telah tersedia pada buku ini, namun lebih jauh lagi pengguna dapat melakukan banyak improvisasi dan pengembangan yang lebih *advance*.

4.1. PERCOBAAN 1 – LED

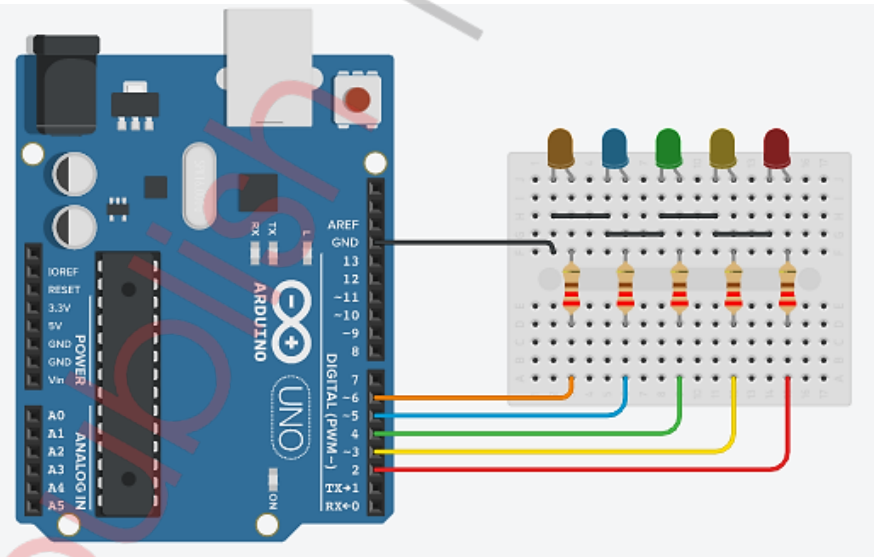
a. Tujuan Percobaan

1) Menggunakan output digital pada Arduino
2) Membuat program untuk mengendalikan LED
3) Mempersingkat penulisan *sketch* untuk menghemat memori

b. Alat dan Bahan

1) Arduino Uno
2) *Bread board*
3) 6 buah LED
4) 6 buah Resistor 220 Ω
5) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.1. Rangkaian Percobaan 1

d. Program/*Sketch*

```
int led1 = 6;//LED orange
int led2 = 5;//LED biru
int led3 = 4;//LED hijau
int led4 = 3;//LED kuning
int led5 = 2;//LED merah

void setup()
{
 pinMode (led1, OUTPUT);
 pinMode (led2, OUTPUT);
 pinMode (led3, OUTPUT);
 pinMode (led4, OUTPUT);
 pinMode (led5, OUTPUT);
}

void loop()
{

//led1 berkedip sebanyak 1 kali
 digitalWrite(led1, HIGH);
 delay(500);
 digitalWrite(led1, LOW);
 delay(500);

//led2 berkedip sebanyak 2 kali
 digitalWrite(led2, HIGH);
 delay(500);
 digitalWrite(led2, LOW);
 delay(500);
 digitalWrite(led2, HIGH);
 delay(500);
 digitalWrite(led2, LOW);
 delay(500);

//led3 berkedip sebanyak 3 kali
 digitalWrite(led3, HIGH);
 delay(500);
 digitalWrite(led3, LOW);
 delay(500);
 digitalWrite(led3, HIGH);
 delay(500);
 digitalWrite(led3, LOW);
 delay(500);
 digitalWrite(led3, HIGH);
 delay(500);
 digitalWrite(led3, LOW);
 delay(500);
```

```
//led4 berkedip sebanyak 4 kali
 digitalWrite(led4, HIGH);
 delay(500);
 digitalWrite(led4, LOW);
 delay(500);
 digitalWrite(led4, HIGH);
 delay(500);
 digitalWrite(led4, LOW);
 delay(500);
 digitalWrite(led4, HIGH);
 delay(500);
 digitalWrite(led4, LOW);
 delay(500);
 digitalWrite(led4, HIGH);
 delay(500);
 digitalWrite(led4, LOW);
 delay(500);

//led5 berkedip sebanyak 5 kali
 digitalWrite(led5, HIGH);
 delay(500);
 digitalWrite(led5, LOW);
 delay(500);
 digitalWrite(led5, HIGH);
 delay(500);
 digitalWrite(led5, LOW);
 delay(500);
 digitalWrite(led5, HIGH);
 delay(500);
 digitalWrite(led5, LOW);
 delay(500);
 digitalWrite(led5, HIGH);
 delay(500);
 digitalWrite(led5, LOW);
 delay(500);
 digitalWrite(led5, HIGH);
 delay(500);
 digitalWrite(led5, LOW);
 delay(500);
}
```

e. Instruksi dan Latihan

1) Instruksi

Buatlah rangkaian dan program sesuai dengan poin **c** dan **d**, kemudian jalankan program.

2) Latihan

Berdasarkan Gambar 4.1 buatlah program yang lebih ringkas namun dengan tujuan yang sama. Silakan terapkan penggunaan **fungsi**, **for**, **array**, dan lain-lain agar penggunaan memori menjadi lebih efisien.

4.2. PERCOBAAN 2 – PUSH BUTTON

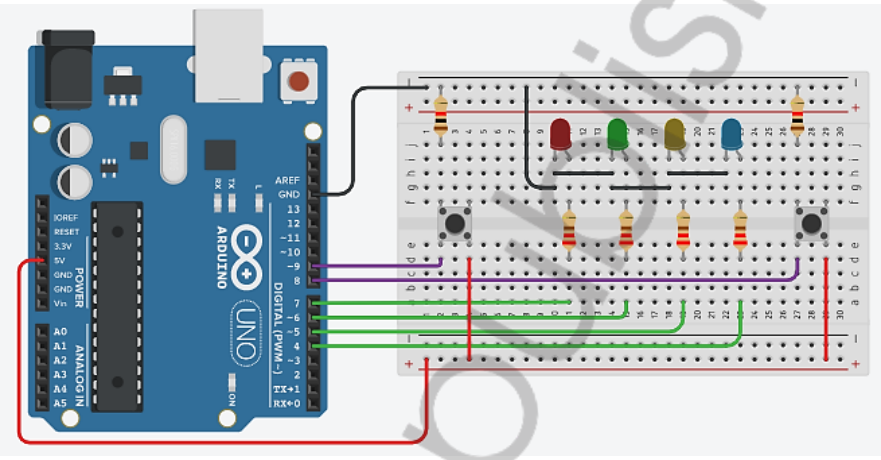
a. Tujuan Percobaan

1) Menggunakan input digital pada Arduino
2) Membuat program untuk membaca nilai digital dari *push button* melalui serial monitor
3) Menerapkan penggunaan **switch case** untuk mengendalikan LED

b. Alat dan Bahan

1) Arduino Uno
2) *Bread board*
3) 2 buah *push button*
4) 2 buah Resistor 1 kΩ
5) 4 buah Resistor 220 Ω
6) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.2. Rangkaian Percobaan 2

d. Program/*Sketch*

```
const int pb1 = 8;//Push button 1
const int pb2 = 9;//Push button 2
int j;

void setup(){
 pinMode(pb1, INPUT);
 pinMode(pb2, INPUT);

Serial.begin(9600);
}

void loop(){
 int nilaiPb1 = digitalRead(pb1);
 int nilaiPb2 = digitalRead(pb2);

 if(nilaiPb1==HIGH){
 j++;
 delay(20);
 }

 if(nilaiPb2==HIGH){
 j--;
 delay(20);
 }
```

```
Serial.println(j);
}
```

e. Instruksi dan Latihan

1) Instruksi
   - Buatlah rangkaian dan program sesuai dengan poin **c** dan **d**, kemudian jalankan program.
   - Buka serial monitor, kemudian tekan *push button* secara bergantian. Amati nilai yang terbaca pada serial monitor.
   - Ubah-ubah nilai **j** lalu amati nilai yang keluar di serial monitor.
2) Latihan
   - Masih dengan rangkaian yang sama sesuai Gambar 4.2, kendalikan nyala LED dengan ketentuan:
     - LED 1 (merah) akan menyala ketika **j = 1**
     - LED 2 (hijau) akan menyala ketika **j = 2**
     - LED 3 (kuning) akan menyala ketika **j = 3**
     - LED 4 (biru) akan menyala ketika **j = 4**
     - Semua LED akan mati ketika **j<1** atau **j>4**
     - Nilai **j** dikendalikan oleh kedua *push button*
   - Gunakan perintah **switch case** untuk mengendalikan *push button.*

4.3. PERCOBAAN 3 – ADC dan PWM

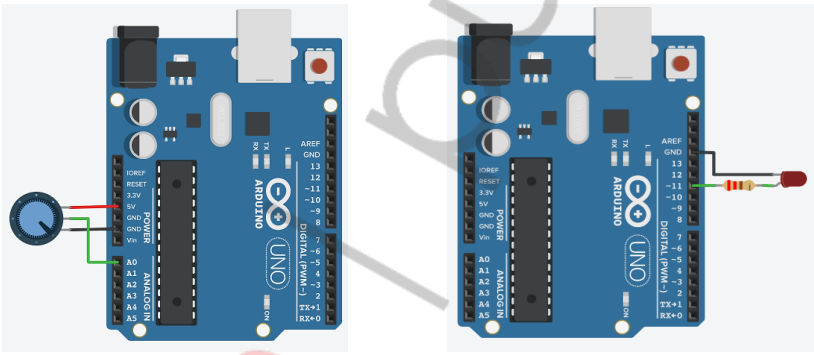
a. Tujuan Percobaan

1) Menerapkan penggunaan PWM dan ADC
2) Menggunakan *channel* analog dan PWM pada Arduino
3) Mengendalikan LED berdasarkan nilai voltase sebuah potensiometer

b. Alat dan Bahan

1) Arduino Uno
2) Potensiometer
3) Resistor 220 Ω
4) LED
5) Beberapa kabel penghubung

c. Rangkaian



(a) (b)

Gambar 4.3. Rangkaian Percobaan 3 (a) ADC dan (b) PWM

d. Program/*Sketch*

1) Program ADC

```
const int pot = A0;//Channel analog
int nilaiPot;

float nilaiSerial;//Agar nilaiSerial berbentuk
 //desimal

void setup() {
 Serial.begin(9600);//Memulai komunikasi serial
 pinMode (A0, INPUT);
}
void loop() {
 nilaiPot = analogRead(pot);//Membaca nilai potensio
```

```
 nilaiSerial = nilaiPot*(5.0/1023);//Konversi nilai
 //analog
 //ke digital (ADC)
 Serial.print("Nilai input = ");
 Serial.print(nilaiPot);
 Serial.print(" = ");
 Serial.print(nilaiSerial);
 Serial.println(" Volt");
 delay(200);
}
```

2) Program PWM

```
int led = 11; //pin PWM

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

void loop()
{
 for (int i = 0; i<255; i +=5)//nilai bertambah 5
 //tiap 30 ms
 {
 analogWrite (led, i);
 delay(30);
 }
 for (int i = 255; i>=0; i-=5)//nilai berkurang 5
 //tiap 30 ms
 {

analogWrite (led, i);
 delay(30);
 }

}
```

e. Instruksi dan Latihan

1) Instruksi

- Buatlah rangkaian serta program ADC dan PWM secara terpisah (dibuat masing-masing).
- Pada percobaan PWM, amati perubahan intensitas cahaya LED.

- Pada percobaan ADC, buka serial monitor dan putar potensiometer. Amati perubahan nilai yang terjadi.

2) Latihan
   - Buatlah sebuah rangkaian baru dengan menggabungkan kedua rangkaian pada Gambar 4.3 menggunakan 1 *board* Arduino.
   - Buatlah program untuk mengatur intensitas cahaya LED berdasarkan nilai potensiometer. Artinya, intensitas cahaya LED akan berubah ketika potensiometer diputar.

4.4. PERCOBAAN 4 – LCD

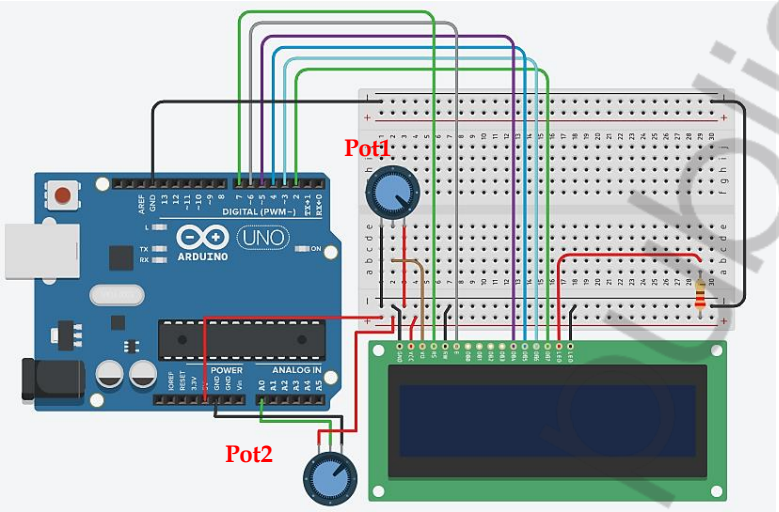
a. Tujuan Percobaan

1) Membuat rangkaian minimum untuk menggunakan LCD 16x2
2) Menerapkan penggunaan LCD sebagai *display* suatu input/output

b. Alat dan Bahan

1) Arduino Uno
2) LCD 16x2
3) 2 buah potensiometer
4) *Breadboard*
5) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.4. Rangkaian Percobaan 4

d. Program/*Sketch*

```
#include<LiquidCrystal.h> //Library LCD
Const int RS=7, EN=6, D4=5, D5=4, D6=3, D7=2;
LiquidCrystal lcd (RS,EN,D4,D5,D6,D7);//Inisialisasi
 //pin LCD
int pot = A0;

void setup(){
 lcd.clear();
 lcd.begin(16,2);//Memulai penggunaan LCD
 lcd.setCursor(0,0);
 lcd.print("Data");

 pinMode (pot, INPUT);
}

void loop(){
 int nilaiPot = analogRead(pot);//Membaca input
 lcd.setCursor(0,1);
 lcd.print(nilaiPot);//Menampilkan data dari
 //potensiometer
}
```

e. Instruksi dan Latihan

1) Instruksi
   - Buatlah rangkaian dan program seperti pada poin **c** dan **d**, lalu jalankan.
   - Atur kontras LCD menggunakan **Pot1**.
   - Putar **Pot2** untuk melihat perubahan data melalui LCD.
2) Latihan
   - Modifikasi rangkaian pada Gambar 4.4 dengan menambahkan 3 buah LED.
   - Konversi data input dari potensiometer menjadi satuan tegangan (0-5Volt)
   - Kendalikan nyala LED dengan ketentuan:
     - LED 1 menyala ketika **1 Volt ≤ tegangan < 2 Volt**
     - LED 2 menyala ketika **2 Volt ≤ tegangan < 3 Volt**.
     - LED 3 menyala ketika **3 Volt ≤ tegangan < 4 Volt**.
     - Semua LED mati jika **tegangan < 1 Volt** atau **tegangan ≥ 4 Volt**.

4.5. PERCOBAAN 5 – KEYPAD

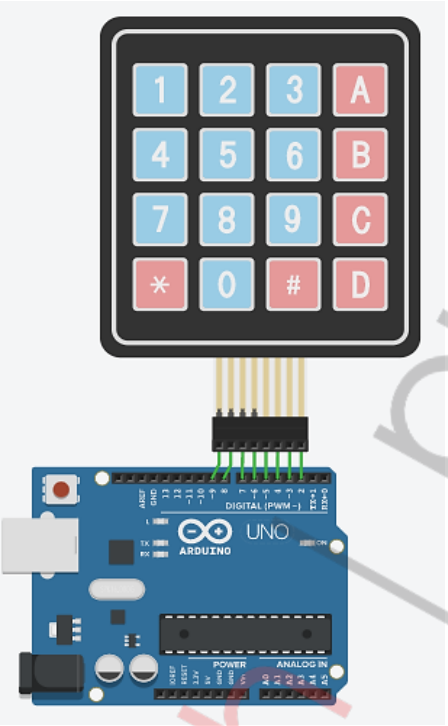
a. Tujuan Percobaan

Menerapkan penggunaan *keypad* untuk mengendalikan suatu output

b. Alat dan Bahan

1) Arduino Uno
2) *Keypad* 4x4
3) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.5. Rangkaian Percobaan 5

d. Program/*Sketch*

```
#include <Keypad.h> //Memanggil library Kepad
const byte Baris = 4;//Jumlah baris
const byte Kolom = 4;//Jumlah kolom

//Memetakan tombol keypad
char petaTombol[Baris] [Kolom] =
{
 {'1','2','3','A'},//Karakter-karakter yang
 {'4','5','6','B'},//tersedia di keypad
 {'7','8','9','C'},
 {'*','0','#','D'},
};
```

```
//Koneksi Arduino dengan keypad
byte pinBaris[Baris] = {9, 8, 7, 6};
byte pinKolom[Kolom] = {5, 4, 3, 2};

//Definisi keypad
Keypad tombol =
Keypad(makeKeymap(petaTombol),pinBaris,pinKolom,
Baris,Kolom);

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

void loop() {
 //Variabel untuk menyimpan data saat tombol
 //ditekan
 char tombolDitekan = tombol.getKey();

 //Kondisi jika tombol ditekan
 if(tombolDitekan != NO_KEY)//Kondisi jika tombol
 //ditekan

 Serial.print(tombolDitekan);
}
```

e. Instruksi dan Latihan

1) Instruksi

   Buatlah rangkaian dan program sesuai poin **c** dan **d**, jalankan program kemudian amati serial monitor.

2) Latihan

   Modifikasi rangkaian dan program yang ada, kemudian gunakan tombol-tombol pada *keypad* untuk mengendalikan output tertentu seperti LED, *buzzer*, tampilan LCD, servo, dan lain-lain (bebas).

4.6. PERCOBAAN 6 – SENSOR TEMPERATUR

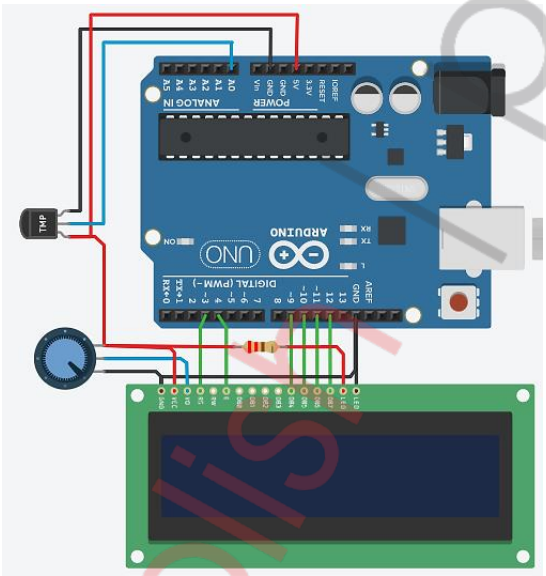
a. Tujuan Percobaan

1) Membaca nilai sensor dan menampilkannya pada LCD
2) Mengendalikan output tertentu berdasarkan nilai temperatur
3) Menggunakan berbagai jenis sensor

b. Alat dan Bahan

1) Arduino Uno
2) Sensor temperatur TMP36
3) LCD 16x2
4) Potensiometer
5) Resistor 220 Ω
6) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.6. Rangkaian Percobaan 6

d. Program/*Sketch*

```
#include <LiquidCrystal.h>
const int RS=3, EN=4, D4=9, D5=10, D6=11, D7=12;
LiquidCrystal lcd (RS, EN, D4, D5, D6, D7);

int TMP36 = A0;//Inisiasi input sensor
```

```
void setup(){
 lcd.clear();
 lcd.begin (16,2);
 lcd.setCursor (0,0);
 lcd.print ("Nilai Suhu");
}

void loop(){
 int baca = analogRead (TMP36);//Membaca nilai sensor
 float tegangan = baca * (5.0/1023);//Konversi nilai
 float suhu = (tegangan-0.5)*100;

 lcd.setCursor(0,1);
 lcd.print(suhu);
 lcd.print(" Celcius");
}
```

e. Instruksi dan Latihan

1) Instruksi
   - Buatlah rangkaian dan program sesuai poin **c** dan **d**, jalankan program.
   - Klik sensor TMP36, gerakkan slider untuk melihat kesesuaian hasil baca sensor dengan di LCD dengan keterangan di sensor.
   - **Untuk rangkaian berbasis hardware:** Gunakan es dan air hangat untuk melihat perubahan suhu secara signifikan.
2) Latihan
   - Modifikasi rangkaian dan program yang ada, tambahkan output berupa LED RGB atau 3 buah LED secara terpisah.
   - Buatlah rangkaian beserta program dengan memanfaatkan sensor selain TMP36. Silakan berkreativitas menggunakan berbagai sensor dan output yang tersedia.

4.7. PERCOBAAN 7 – SENSOR ULTRASONIK

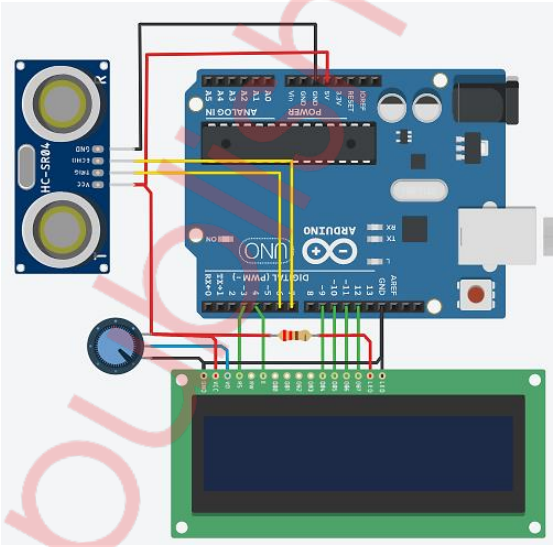
a. Tujuan Percobaan

1) Mengetahui prinsip kerja sensor ultrasonik
2) Membaca nilai sensor dan menampilkannya pada LCD
3) Memprogram Arduino yang terhubung dengan sensor dengan dan tanpa *library*

b. Alat dan Bahan

1) Arduino Uno
2) Sensor ultrasonik HC-SR04
3) LCD 16x2
4) Potensiometer
5) Resistor 220 Ω
6) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.7. Rangkaian Percobaan 7

d. Program/*Sketch*

```
#include <LiquidCrystal.h>
const int RS=3, EN=4, D4=9, D5=10, D6=11, D7=12;
LiquidCrystal lcd (RS, EN, D4, D5, D6, D7);

//Inisiasi sensor ultrasonik
const int trig = 6;//Transmitter
const int echo = 7;//Receiver
float durasi, jarak;

void setup(){
 pinMode (trig, OUTPUT);
 pinMode (echo, INPUT);

 lcd.clear();
 lcd.begin (16,2);
 lcd.setCursor (0,0);
 lcd.print ("Jarak Objek");
}

void loop(){
 digitalWrite (trig, LOW);
 delay (2);
 digitalWrite (trig, HIGH);
 delayMicroseconds (10);
 digitalWrite(trig, LOW);

 durasi = pulseIn (echo, HIGH);
 jarak = durasi*17/1000;

 Serial.print("Jarak Objek: ");
 Serial.println(jarak);

 lcd.setCursor (0,1);
 lcd.print(jarak);
 lcd.print( "cm");
}
```

e. Instruksi dan Latihan

1) Instruksi

- Buatlah rangkaian dan program sesuai poin **c** dan **d**, jalankan program.

- Tekan sensor ultrasonik, geser objek, dan bandingkan nilai yang terbaca pada sensor dan LCD.
- Program pada poin **d** dibuat tanpa menggunakan *library* karena di Tinkercad belum mendukung penggunaan *library* HC-SR04. Oleh karena itu, jelaskan proses perhitungan jarak pada sensor ultrasonik berdasarkan program.
- Berikan penjelasan mengenai selisih jarak yang terbaca pada sensor dan LCD.
- **Untuk rangkaian berbasis hardware:** Gunakan sebuah benda/objek, dekatkan dan jauhkan dari sensor untuk melihat hasil baca sensor. Selanjutnya jelaskan proses perhitungan jarak pada sensor berdasarkan program.

2) Latihan

Buatlah sebuah rangkaian lain dengan sensor yang berbeda, kemudian ketikkan program dengan menggunakan *library* maupun tanpa *library*.

4.8. PERCOBAAN 8 – CONTOH PROJECT SEDERHANA

a. Tujuan Percobaan

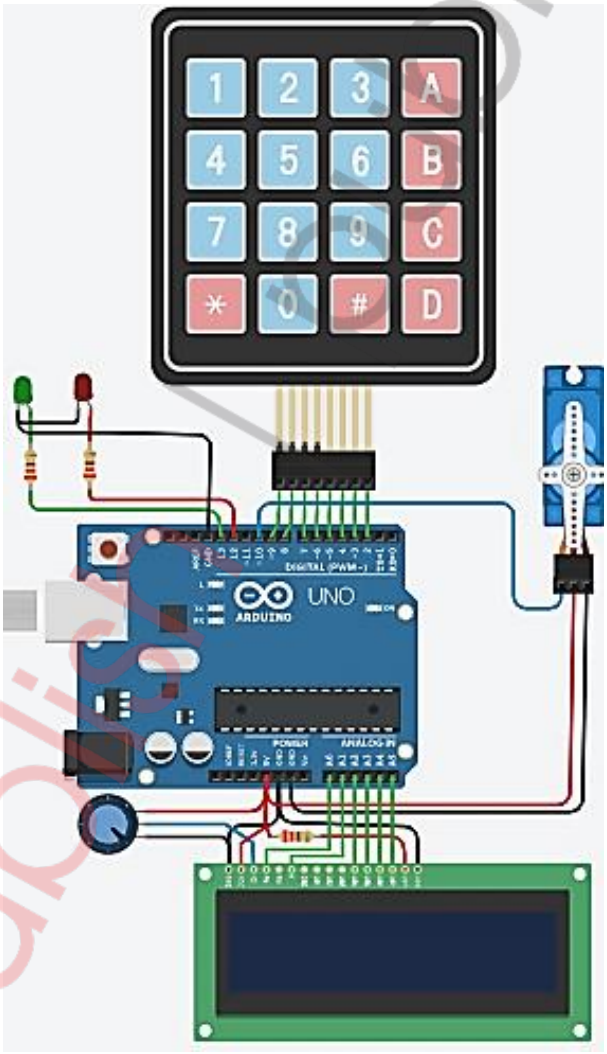
1) Menerapkan penggunaan *keypad*, LCD, LED, dan motor servo
2) Membuat *project* sederhana untuk membuka pintu menggunakan *password*

b. Alat dan Bahan

1) Arduino Uno
2) *Keypad*
3) LCD 16x2
4) 2 buah LED
5) Potensiometer
6) 3 buah Resistor 220 Ω

7) Servo
8) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.8. Rangkaian Percobaan 8

d. Program/Sketch

```
#include <LiquidCrystal.h>//Memasukkan library LCD
const int RS=A0, EN=A1, D4=A2, D5=A3, D6=A4, D7=A5;
LiquidCrystal lcd (RS, EN, D4, D5, D6, D7);

#include <Servo.h> //Memasukkan library Servo
Servo servo;

#include <Keypad.h>//Memasukkan library Keypad
const byte BARIS = 4;//empat baris
const byte KOLOM = 4;//empat kolom

//definisi simbol pada tombol yang ada di keypad
char petaTombol[BARIS][KOLOM] = {
 {'1','2','3','A'},
 {'4','5','6','B'},
 {'7','8','9','C'},
 {'*','0','#','D'}
};
byte pinBaris[BARIS] = {9, 8, 7, 6};//Terhubung
 //dengan pin
                                    //baris pada
 //keypad
byte pinKolom[KOLOM] = {5, 4, 3, 2};//Terhubung
 //dengan pin
                                    //kolom pada
 //keypad
Keypad tombol = Keypad(makeKeymap(petaTombol), pinBaris,
pinKolom, BARIS, KOLOM);
int ledM = 12, ledH = 13;
int posisi = 0;
char* password = "1234";

void setup(){
 lcd.begin(16,2);
 servo.attach(10);
 pinMode(ledH, OUTPUT);
 pinMode(ledM, OUTPUT);
 statusLock (true);//Status password
}

void loop(){

//Mendefinisikan tombol yang ditekan
 char tombolDitekan = tombol.getKey();
 lcd.setCursor (0,0);
 lcd.print("SELAMAT DATANG");
 lcd.setCursor(0,1);
```

```
 lcd.print("Masukkan Psw");
 if(tombolDitekan == '*' || tombolDitekan == '#' ||
 tombolDitekan == 'A' || tombolDitekan == 'B' ||
 tombolDitekan == 'C' || tombolDitekan == 'D')
 {
 posisi = 0;
 statusLock (true);
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print(" Psw Salah! ");
 delay(100);
 lcd.clear();
 statusLock (true);//Untuk mengunci kembali
 //setelah terbuka, tekan
 //salah satu di antara
                  //*, #, A, B, C, D
 }
 if (tombolDitekan == password [posisi])
 {
 posisi ++;
 }
 if(posisi == 4)
 {
 statusLock (false);
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("*Terverifikasi*");
 delay(3000);
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Selamat, Anda");
 lcd.setCursor(0, 1);
 lcd.print("Berhasil Masuk");
 delay(7000);
 lcd.clear();
 }
 delay(100);
}

//Fungsi untuk mendefinisikan statusLock
void statusLock(int terkunci){
 if(terkunci){//Jika terkunci LED Merah menyala
 digitalWrite(ledM, HIGH);
 digitalWrite(ledH, LOW);
 servo.write(90);
 }
 else{
 digitalWrite(ledM, LOW); //Jika tidak terkunci
 //LED Hijau menyala
 digitalWrite(ledH, HIGH);
```

```
servo.write(0);
}
}
```

e. Instruksi dan Latihan

1) Instruksi
   - Buatlah rangkaian dan program sesuai poin **c** dan **d**, jalankan program.
   - Tekan tombol sesuai *password* yang tersedia di program, pastikan rangkaian dan program telah berjalan dengan baik.

2) Latihan

   Buatlah sebuah *project* sederhana menggunakan Mikrokontroler Arduino yang disertai dengan berbagai jenis sensor, *display*, dan komponen lainnya. Silakan berkreasi.

REFERENSI

Smith, G. A. (2011): Introduction to Arduino-A piece of cake. ISBN: 1463698348.

Evans, B. W. (2008): Arduino Programming Notebook. ISBN: 978-1-4302-3778-5.

Purdum, J. (2011): Beginning C for Arduino. ISBN: 978-1-4302-4777-7.

Crisp J. (2004): Introduction Microprocessors and Microcontrollers (2nd Edition)-an imprint of Elsevier. ISBN:0-7506-5989-0.

https://e-dokumen.id/dokumen/

https://id.wikipedia.org

https://www.arduino.cc

https://www.labelektronika.com

https://www.myusro.id

https://www.thinkercad.com

https://www.tptumetro.com

httpd://www.tutorialspoint.com

PROFIL PENULIS

Penulis bernama lengkap Ahmad Zarkasi dilahirkan di Selanglet pada tanggal 23 April 1991. Penulis menempuh pendidikan S1 Fisika di Universitas Mataram dan pendidikan S2 Fisika di Universitas Brawijaya dengan bidang minat Elektronika dan Instrumentasi. Saat ini, penulis merupakan staf pengajar di Program Studi Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Mulawarman. Beberapa mata kuliah yang penulis ampu adalah Pengantar Mikrokontroler, Sistem Sensor, Fisika Instrumentasi, Instrumentasi Geofisika, Listrik Magnet, dan Fisika Eksperimen.