Technotra

Course on Microcontroller, Arduino & Robotics

Course outline

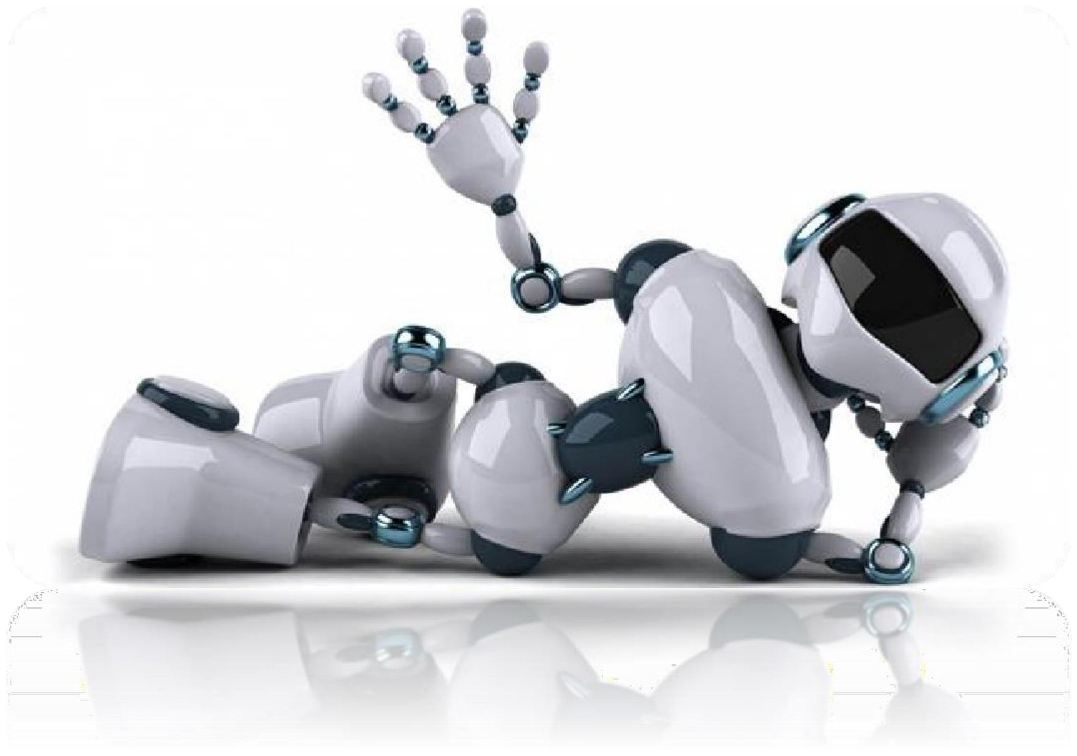
- What is Robot? Introducing different types of Robot
- What is Microcontroller? Microcontroller & Microprocessor
- AVR ATMEL Microcontroller
- Programming of ATMEL Microcontroller using Programmer's notepad
- Simulation using Proteus ISIS Software
- Introducing Basic Components: Resistor, Capacitor, Rectifier Diode, Zener Diode, LED, Push button, Breadboard etc and their use
- Introducing Arduino(Types of Arduino)
- What is sensor? Classification of sensors, Different types of sensors(Temperature Sensor, Ultrasonic Sensor, Humidity Sensor, PIR motion sensor, Gas Sensor, LDR), How to make light sensor circuit using voltage divider rule and operational amplifier.

- What is Motor?, How to choose motor for a project, Different types of motors(DC, Stepper, Servo), DC motor driving, driving the motors clockwise, anticlockwise, slow, fast etc, how to choose a motor controller, using high current motor drivers, making a motor driver circuit using H-bridge
- Motor Driver Module (L298N, L293D, BTS)
- How to make 5V power Supply, Different types of Battery, How to choose battery and its rating.
- AC to DC switching circuit using relay
- Interfacing LCD display. Program the displays and experience the outcome practically.
- Dot Matrix, 7 Segment Display, IR Receiver
- Wireless communication using Bluetooth, Pairing two Bluetooth (Master & Slave)
- Making an app controlled robot using Bluetooth module HC-05
- Home automation using Bluetooth

- Collecting Data from IR Sensor
- PCB Designing Using Proteus, KiCad
- Making a Line Follower Robot & Human Following Robot
- Using Sonar Sensor Making Obstacle Avoiding Robot
- Voice Controlled Robot
- Use of ESP8266 NodeMCU
- How to make Humanoid Robot, Degree of Freedom, Actuator etc.

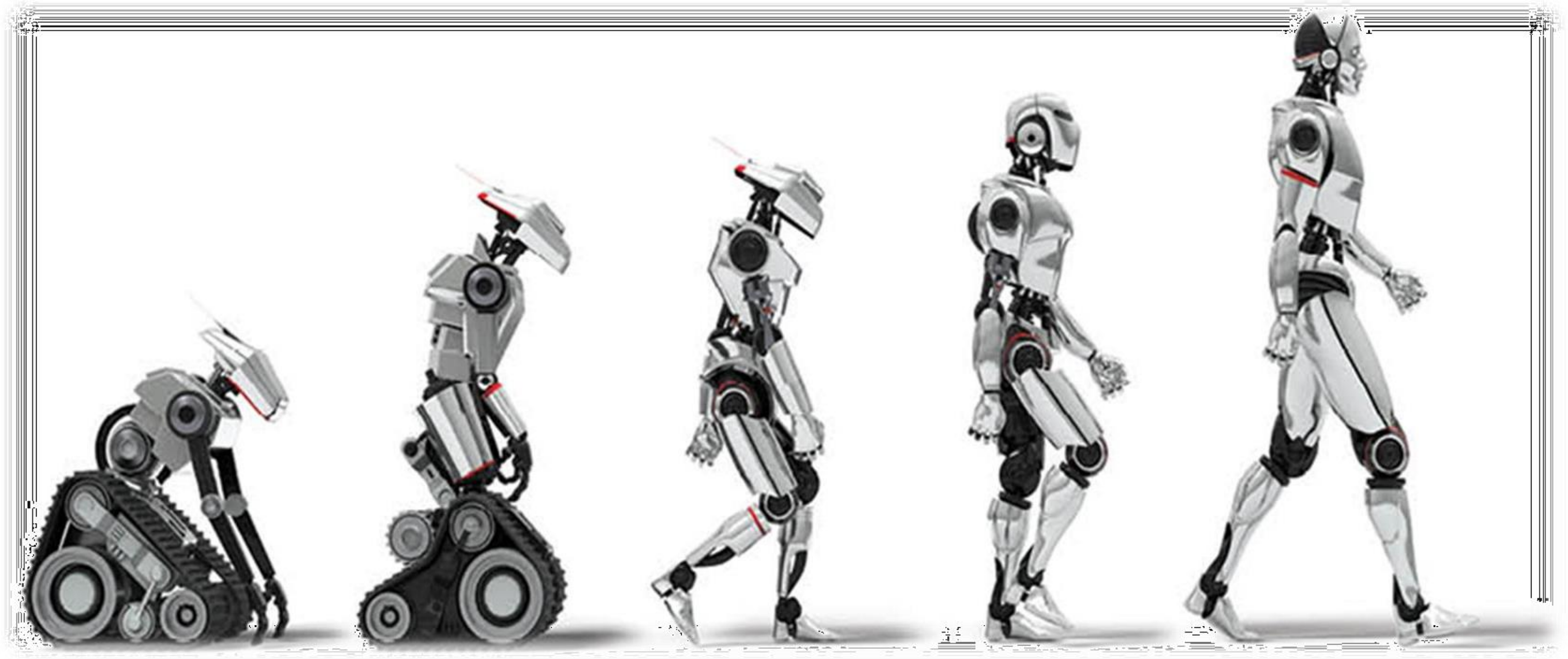
What is Robot?

From ROBOT INSTITUTE OF AMERICA , “A Robot is a reprogrammable, multifunctional manipulator designed to moved materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks”



What is Robotics?

Robotics is a branch of engineering that involves the conception, design, manufacture, and operation of robots. This field overlaps with electronics, computer science, artificial intelligence, mechatronics, nanotechnology and bioengineering.



Three Laws of Robotics

1. Robots must never harm human beings.

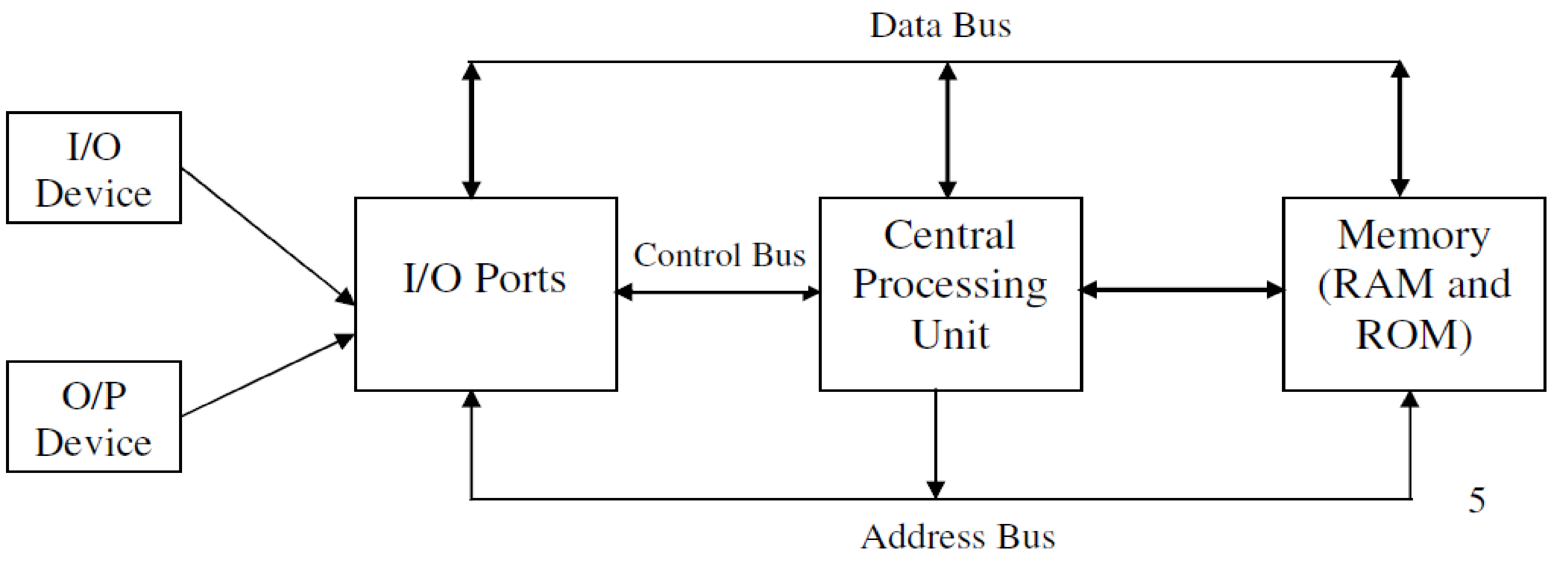
2. Robots must follow instructions from humans without violating rule 1.

3. Robots must protect themselves without violating the other rules.

What is Micro-Computer (Computer)?

Micro-computer is a system which consist at least the following components

- **Microprocessor**
- **Memory**
- **Input Port**
- **Output Port**

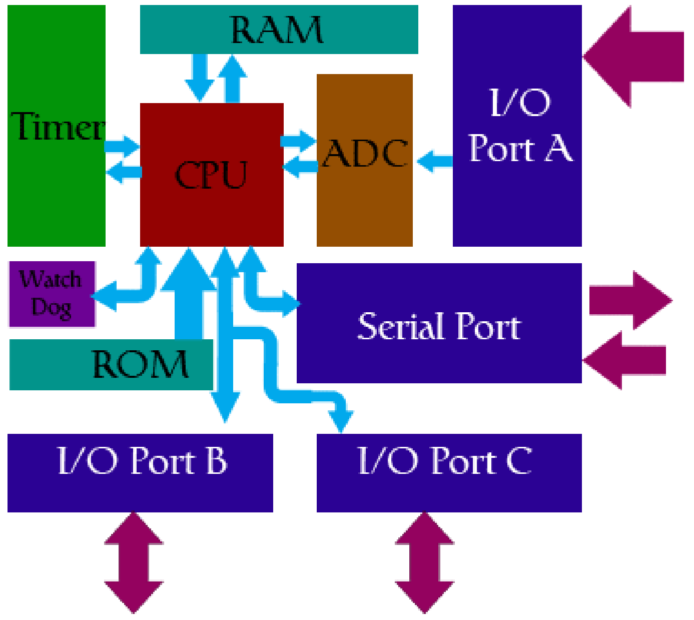


What is Microcontroller?

Micro-computer in a single chip.

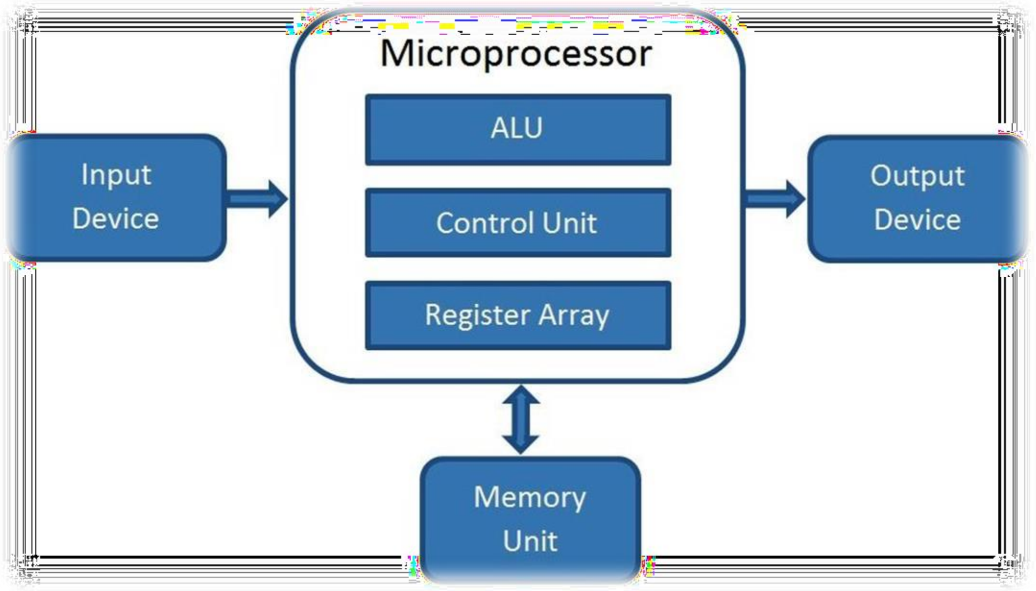
A microcontroller may consist of following functional units:

- Central Processing Unit
- Memory Unit
- System Bus
- Input/Output Unit
- Serial Communication
- Timer Unit.
- Watchdog
- Analog to Digital Converter
- Oscillator



Microprocessor

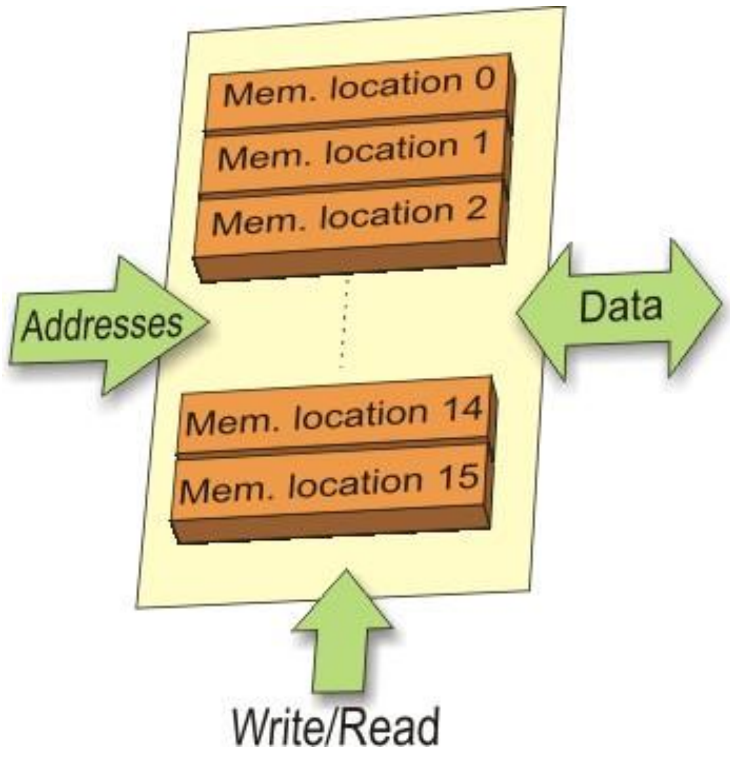
It takes binary data from memory or input device and provides outputprocessing the data as per user instruction.



Microprocessor is a multipurpose, programmable register based electronic device which read binary instructions from memory, process the input data as per instructions and provides output.

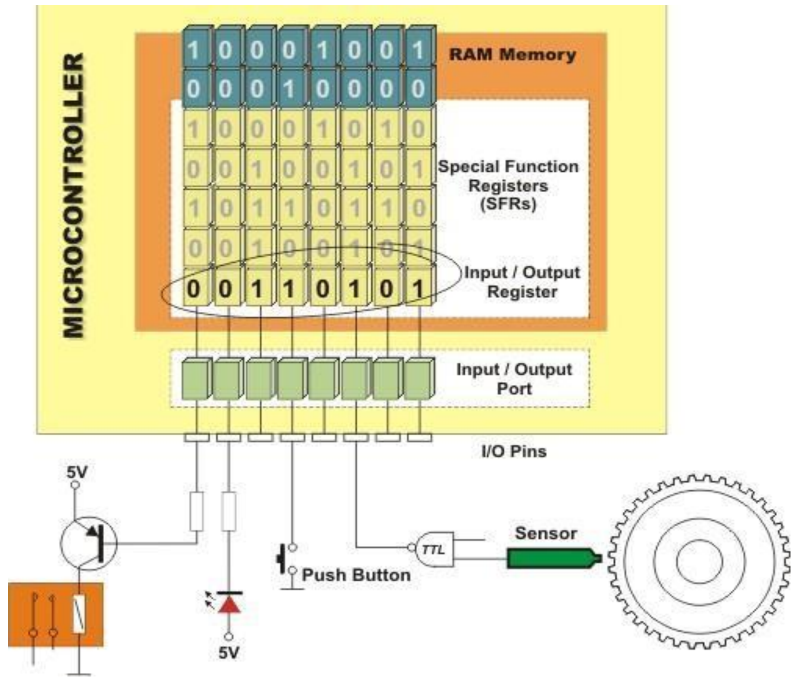
Memory Unit

- **Read Only Memory (ROM):** Program Memory
- **Random Access Memory (RAM):** To store data (variables)
- **Electrically Erasable Programmable ROM (EEPROM):** Store data (variable) even power source is shutdown.



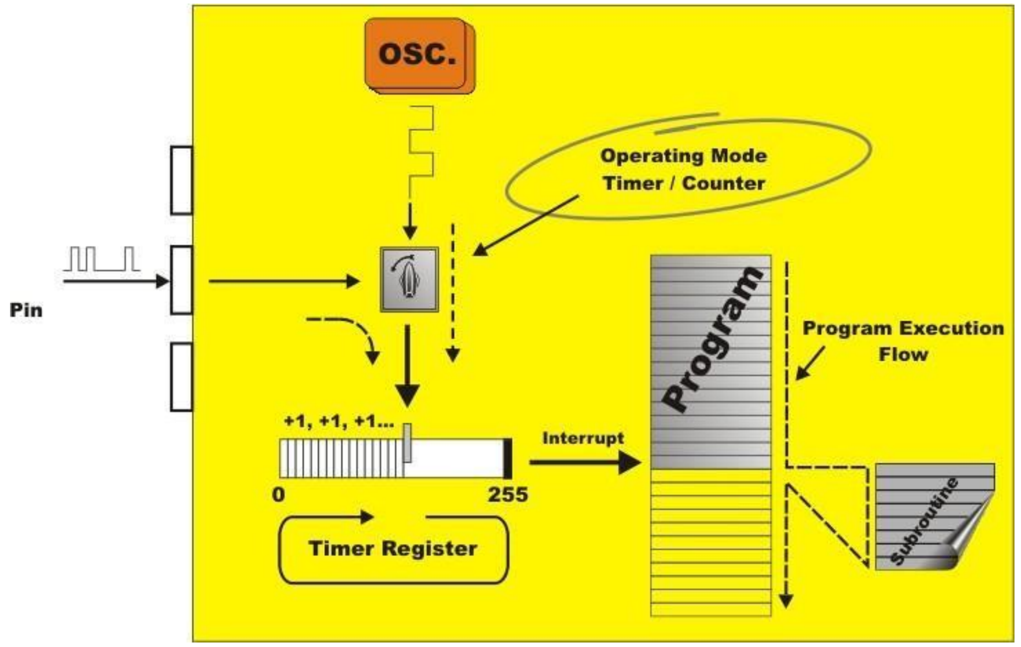
Input/output ports (I/O Ports)

In order to make the microcontroller useful, it is necessary to connect it to peripheral devices. Each microcontroller has one or more registers (called a port) connected to the microcontroller pins. Why do we call them input/output ports? Because it is possible to change a pin function according to the user's needs.



Timer/Counters

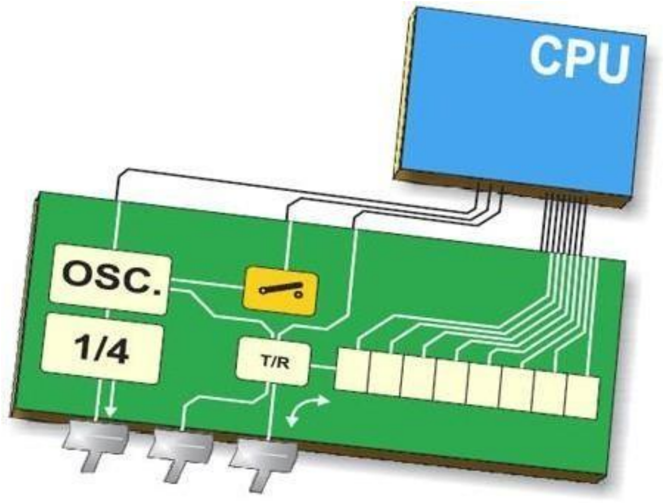
These are act as "**stopwatches**" or,"**external event counter**". These are commonly 8- or 16-bit SFRs the contents of which is automatically incremented by each coming pulse.



Serial communication

The most commonly used serial communication systems are:

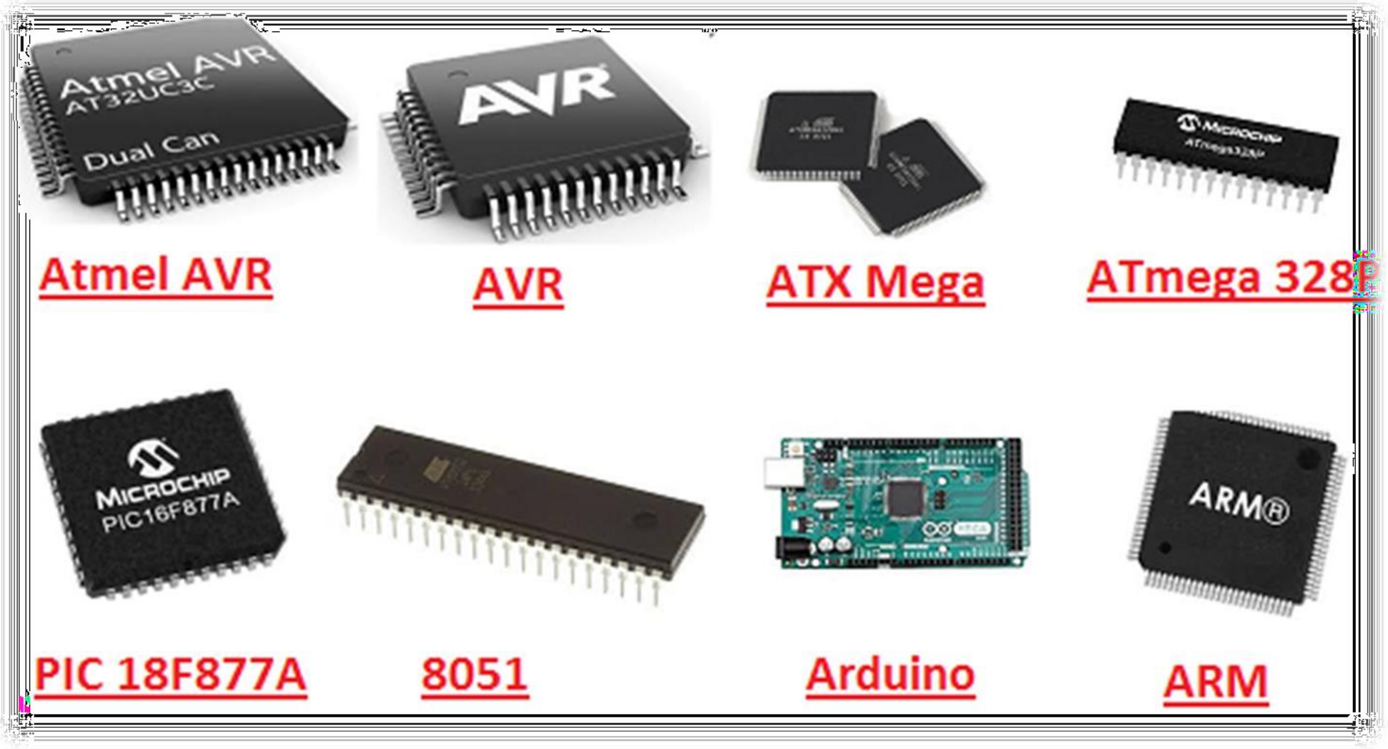
- I²C (Inter Integrated Circuit)
- SPI (Serial Peripheral Interface Bus)
- UART (Universal Asynchronous Receiver/Transmitter)



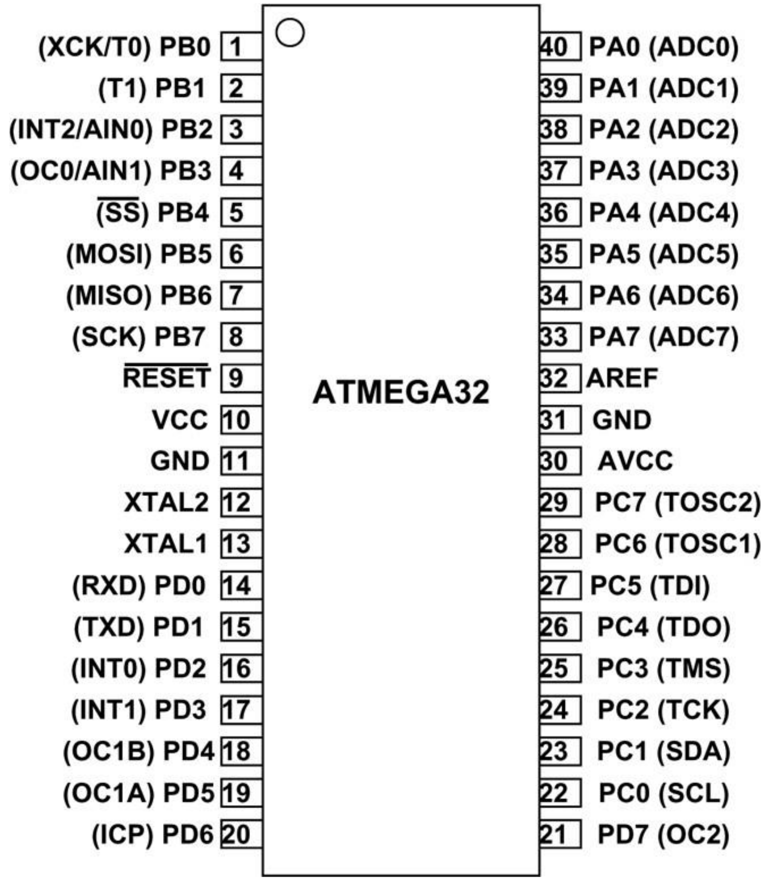
Name of Some Microcontroller Manufacturer

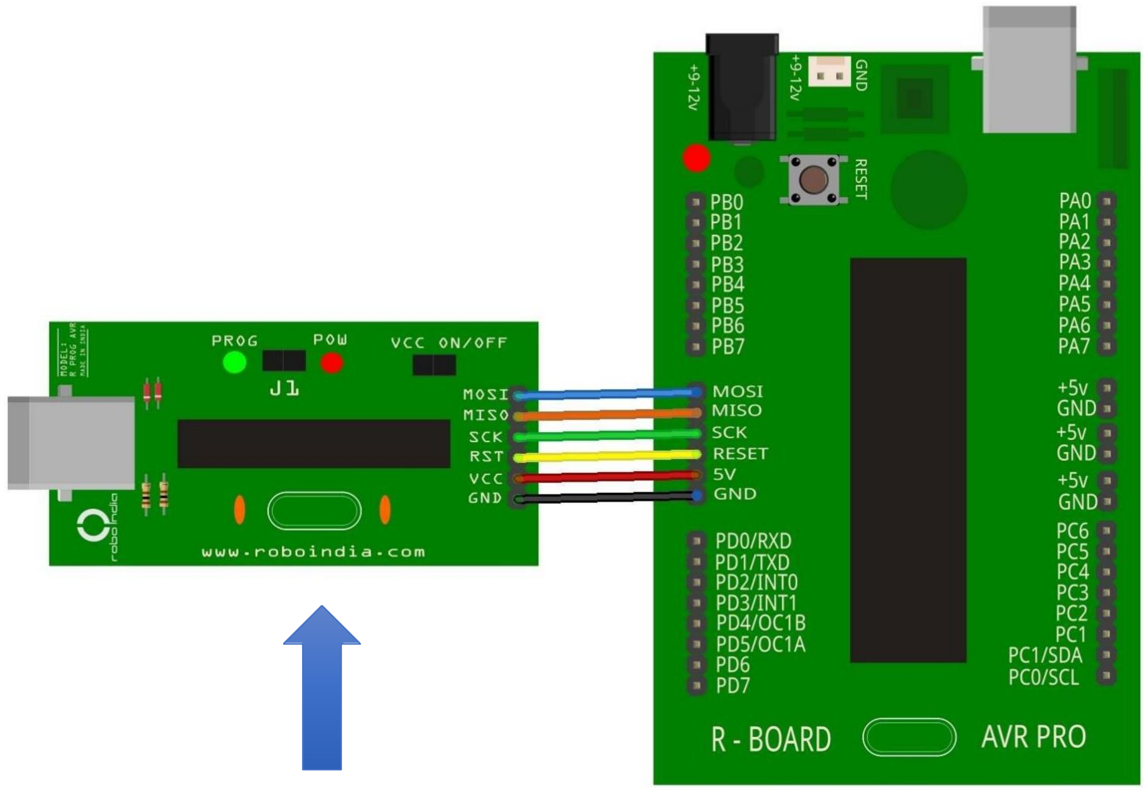
ATMEL (AVR microcontroller)
Microchip (PIC microcontroller)
Texas Instruments (TI)
Freescale
Philips
Motorola

AVR was developed by Alf-Egil Bogen and Vegard Wollan. AVR derives its name from its developers and stands for Alf-Egil Bogen Vegard Wollan RISC microcontroller, also known as Advanced Virtual RISC.



ATMEL Microcontroller





Loader

Programming

Data Type

Char

int

float

double

Size and Range of Data Types on 16 bit machine

Type	Size(Bits)	Range
Char or Signed Char	8	-128 to 127
Unsigned	8	0 to 255
Int or signed int	16	-32768 to 32767
Unsigned int	16	0 to 655535
Float	32	3.4 e-38 to 3.4 e+38
Double	64	1.7 e-308 to 1.7e+308

Programming

Operators

Bitwise left shift	<<
Bitwise right shift	>>
Bitwise AND	&
Bitwise OR	|
Bitwise exclusive OR	^
Bitwise exclusive NOR	~

Programming

Syntax Understanding

PORTA = 0000 1111

DDRA = 0000 0000

DDR= Data Direction Register

Pa0=1

Pa2=1

Pa3=1

Pa4=1

Pa4=0

Pa5=0

Pa6=0

Pa7=0

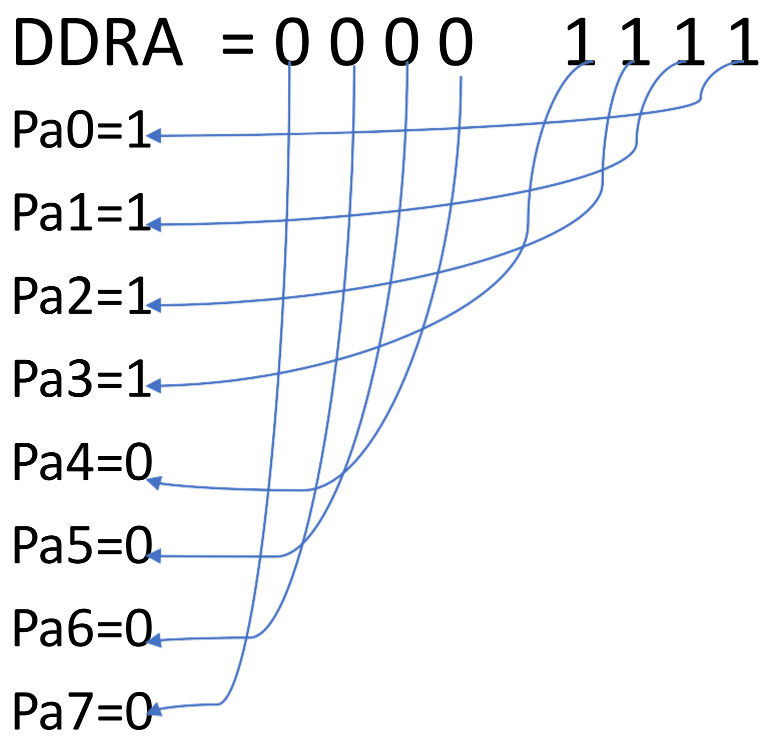
Determine the behavior of Pin

If
DDRA = 0000 0000
Then pin of port(A/B/C/D) set in input mode

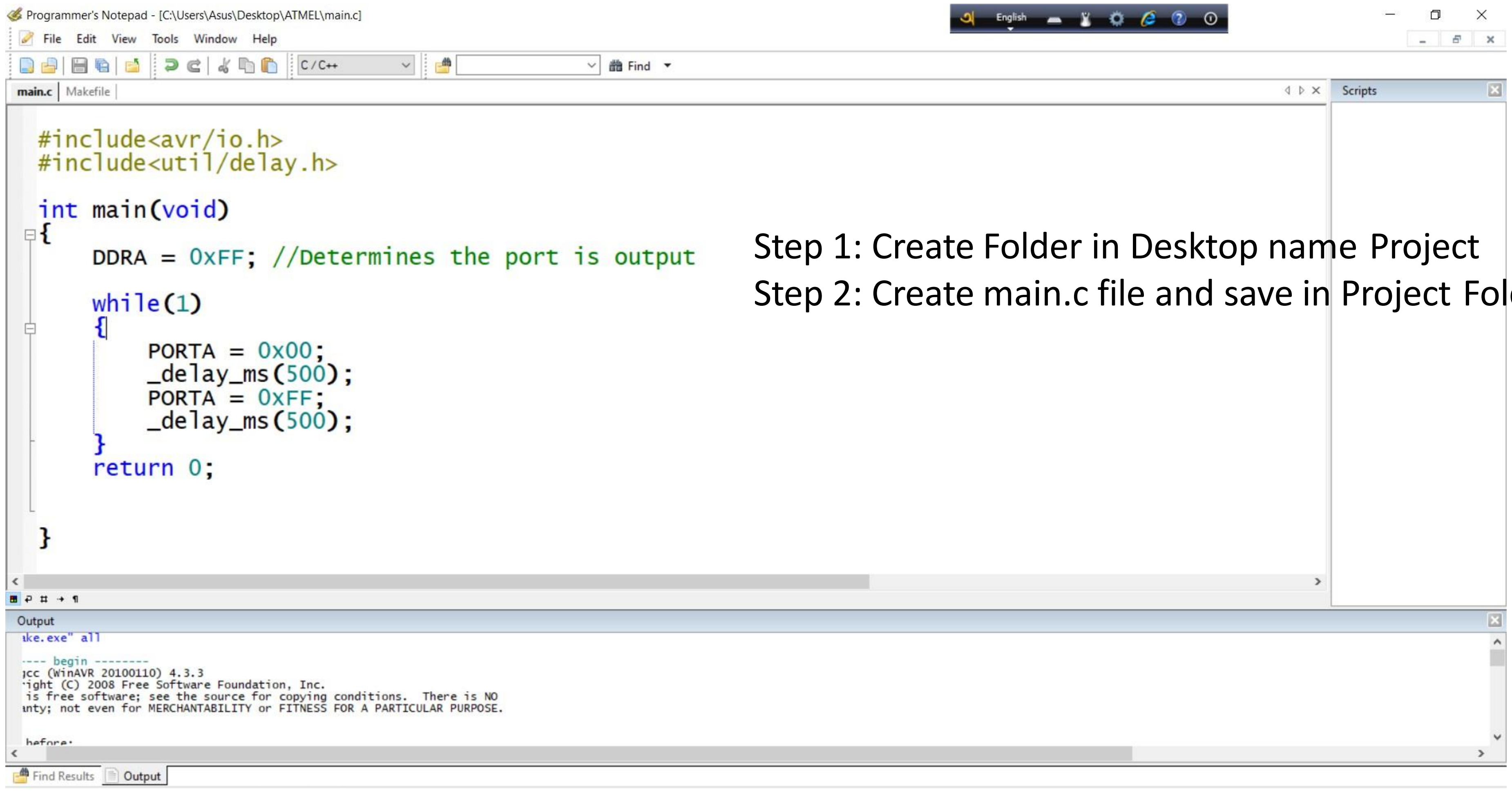
If
DDRA = 1111 1111
Then pin of port(A/B/C/D) set in output mode

Programming

<u>Syntax Understanding</u>



Programmers Notepad


Programmer's Notepad - [C:\Users\Asus\Desktop\ATMEL\main.c]
File Edit View Tools Window Help
C / C++
Find
main.c
Makefile
Scripts
Output
Find Results
Output

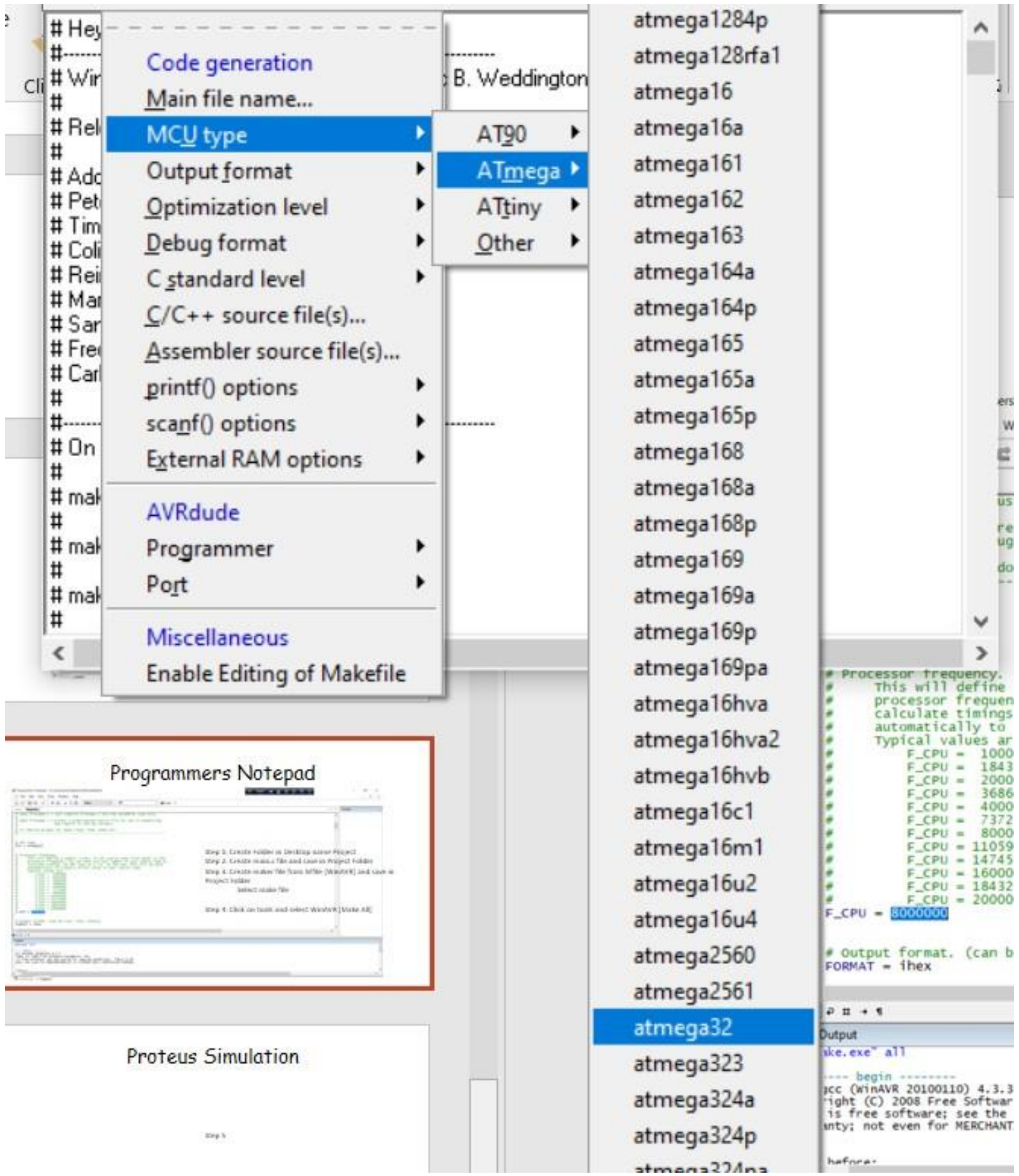
```
#include<avr/io.h>
#include<util/delay.h>

int main(void)
{
    DDRA = 0xFF; //Determines the port is output

    while(1)
    {
        PORTA = 0x00;
        _delay_ms(500);
        PORTA = 0xFF;
        _delay_ms(500);
    }
    return 0;

}
```

```
ıke.exe" all

---- begin --------
ɟcc (WinAVR 20100110) 4.3.3
'ight (C) 2008 Free Software Foundation, Inc.
 is free software; see the source for copying conditions.  There is NO
ınty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

 before:
```

Step 1: Create Folder in Desktop name Project

Step 2: Create main.c file and save in Project Folder

Mfile[WinAVR]



```
# Processor frequency.
#     This will define
#     processor frequen
#     calculate timings
#     automatically to
#     Typical values ar
#         F_CPU =   1000
#         F_CPU =   1843
#         F_CPU =   2000
#         F_CPU =   3686
#         F_CPU =   4000
#         F_CPU =   7372
#         F_CPU =   8000
#         F_CPU =  11059
#         F_CPU =  14745
#         F_CPU =  16000
#         F_CPU =  18432
#         F_CPU =  20000
F_CPU = 8000000

# Output format. (can b
FORMAT = ihex
```

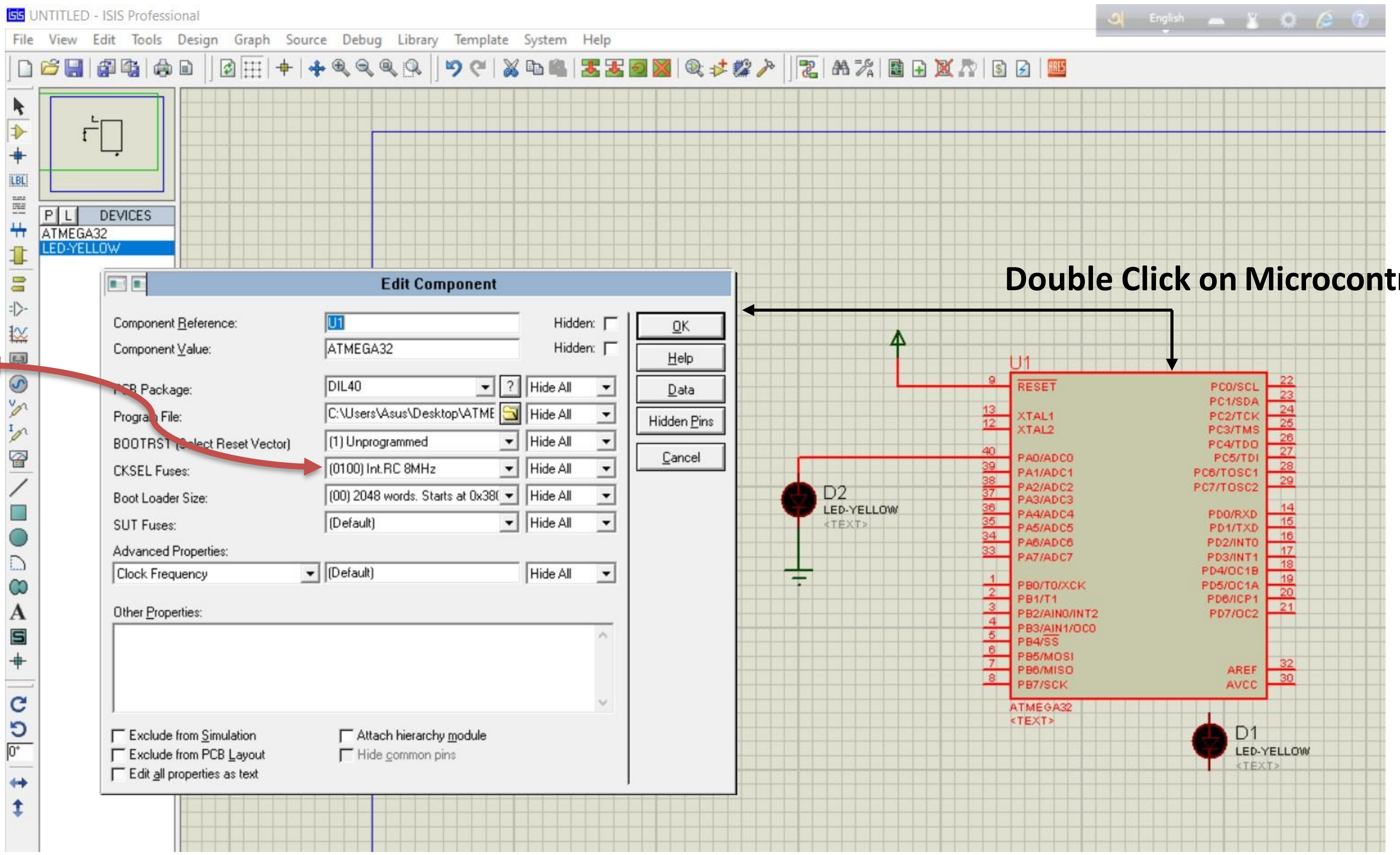
```
ake.exe" all

-------- begin --------
gcc (WinAVR 20100110) 4.3.3
right (C) 2008 Free Softwar
 is free software; see the
anty; not even for MERCHANT
```

Step 3: Create maker file and save in Project Folder
Step 4: Click on tools and select WinAVR [Make All]

Proteus Simulation

Change the clock frequency to 8MHZ

Double Click on Microcontroller



Edit Component

Component Reference:	U1	Hidden: ☐	
Component Value:	ATMEGA32	Hidden: ☐	
PCB Package:	DIL40		Hide All
Program File:	C:\Users\Asus\Desktop\ATME		Hide All
BOOTRST (Select Reset Vector)	(1) Unprogrammed		Hide All
CKSEL Fuses:	(0100) Int.RC 8MHz		Hide All
Boot Loader Size:	(00) 2048 words. Starts at 0x38(		Hide All
SUT Fuses:	(Default)		Hide All

Advanced Properties:

Clock Frequency	(Default)	Hide All

Other Properties:

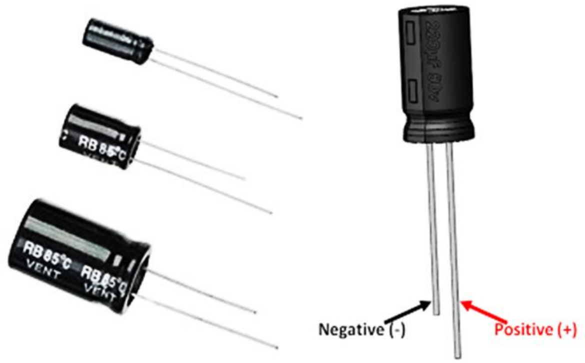
☐ Exclude from Simulation
☐ Exclude from PCB Layout
☐ Edit all properties as text
☐ Attach hierarchy module
☐ Hide common pins

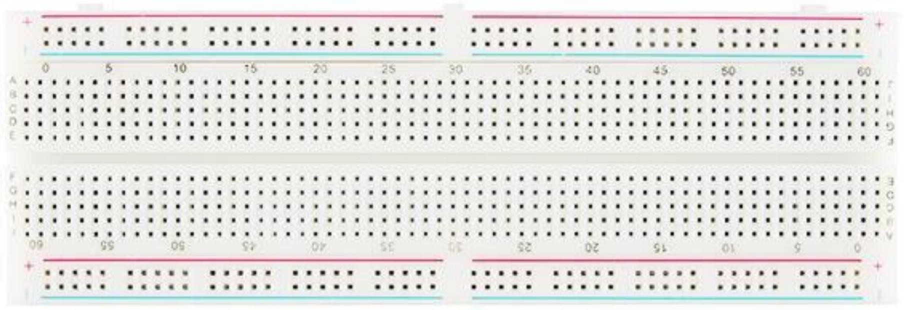
Technotra

Course on Microcontroller, Arduino & Robotics

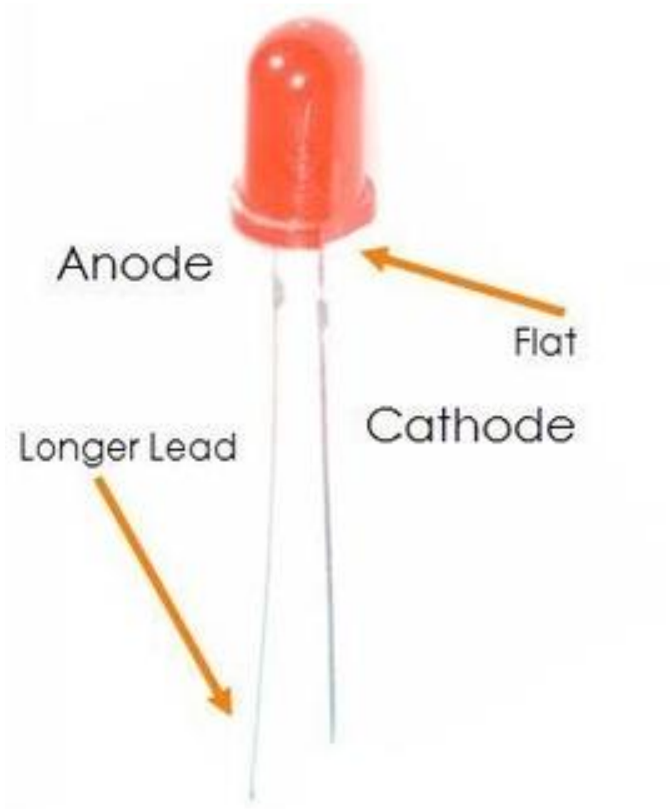
Introduction of Basic Components

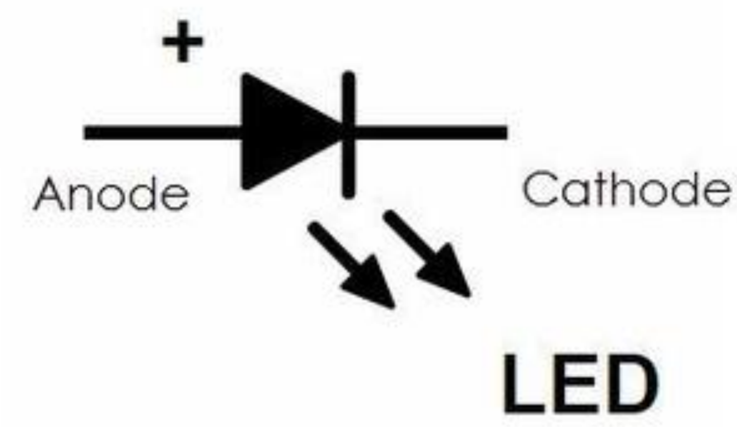
- Resistor
- Capacitor
- Amplifier Diode
- Zener Diode
- LED
- Push Button
- Breadboard





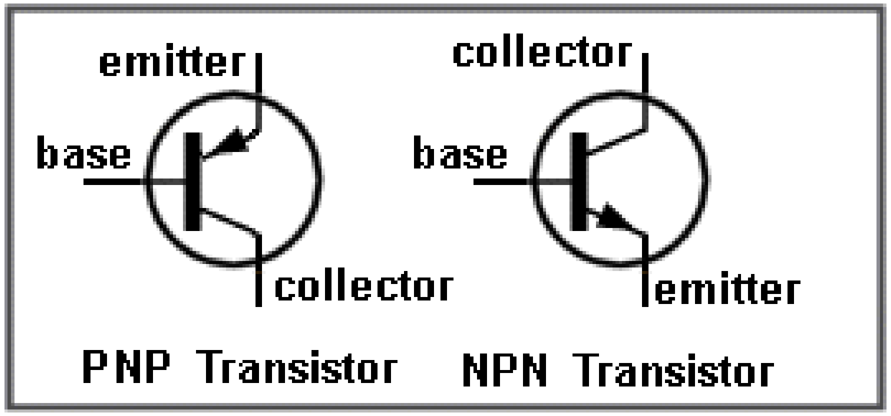
LED : Light Emitting Diode

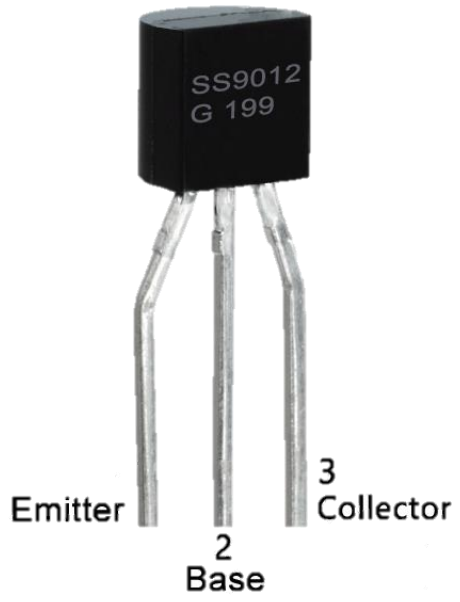




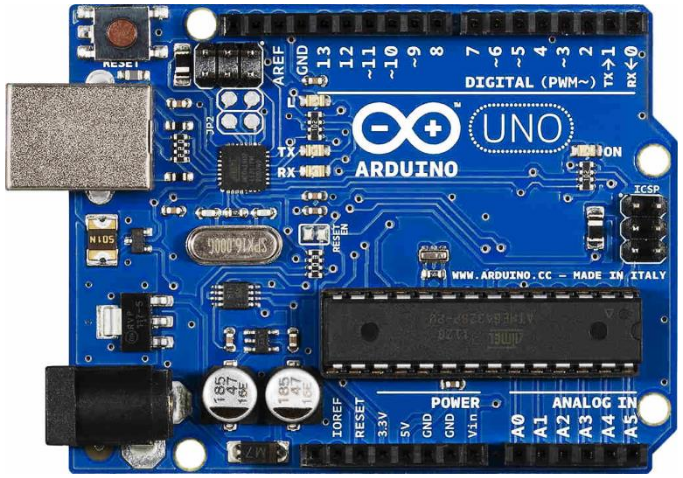
Transistor :

- Transistor is a semi - conductor device used to amplify or switch electronic signals and electrical power.
- **Collector :** Similar to the positive leg on an LED, this is where power flows in.
- **Base :** Represents “Trigger" pin coming from the controller, sensor, or others.
- **Emitter :** Like the negative leg on an LED, this is the ground side of the transistor.

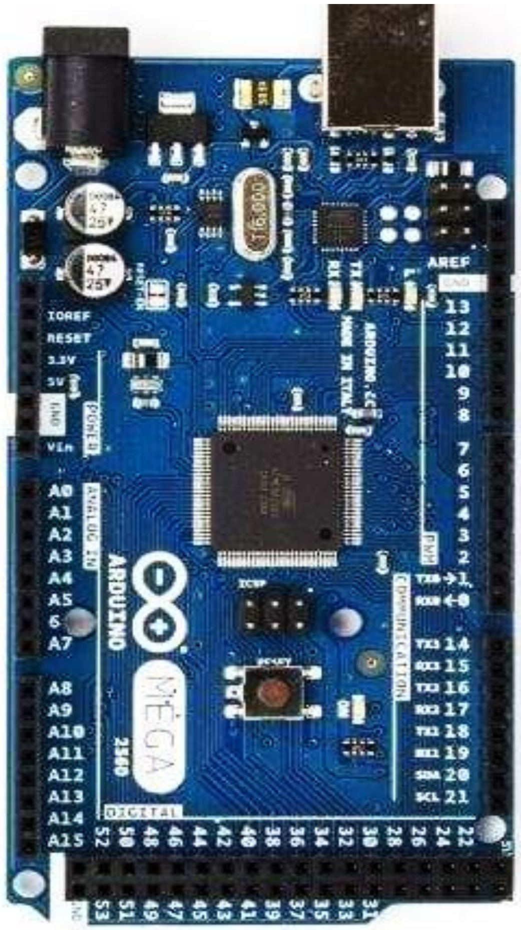




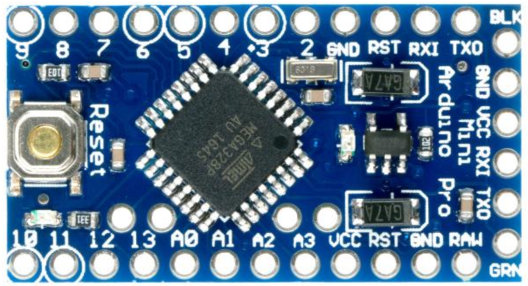
Arduino UNO :



Arduino MEGA :

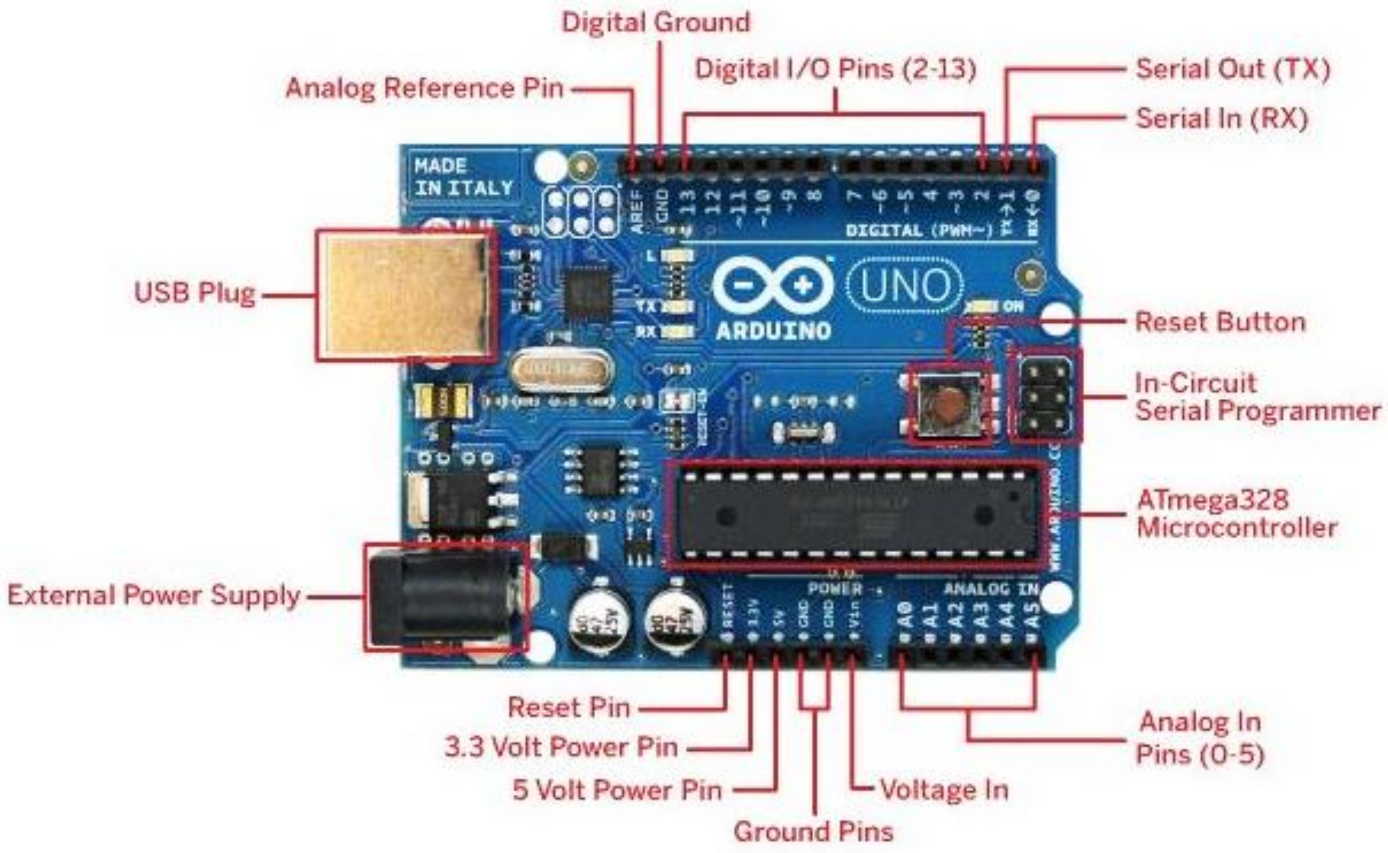


Arduino MINI :

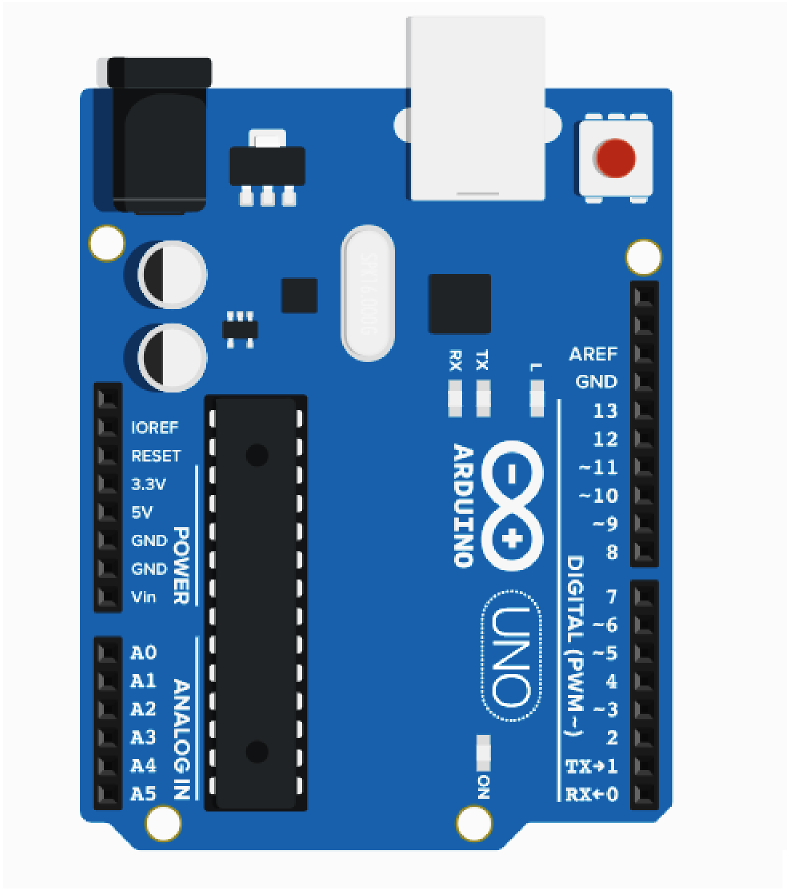


Technotra

Pin Diagram of Arduino UNO



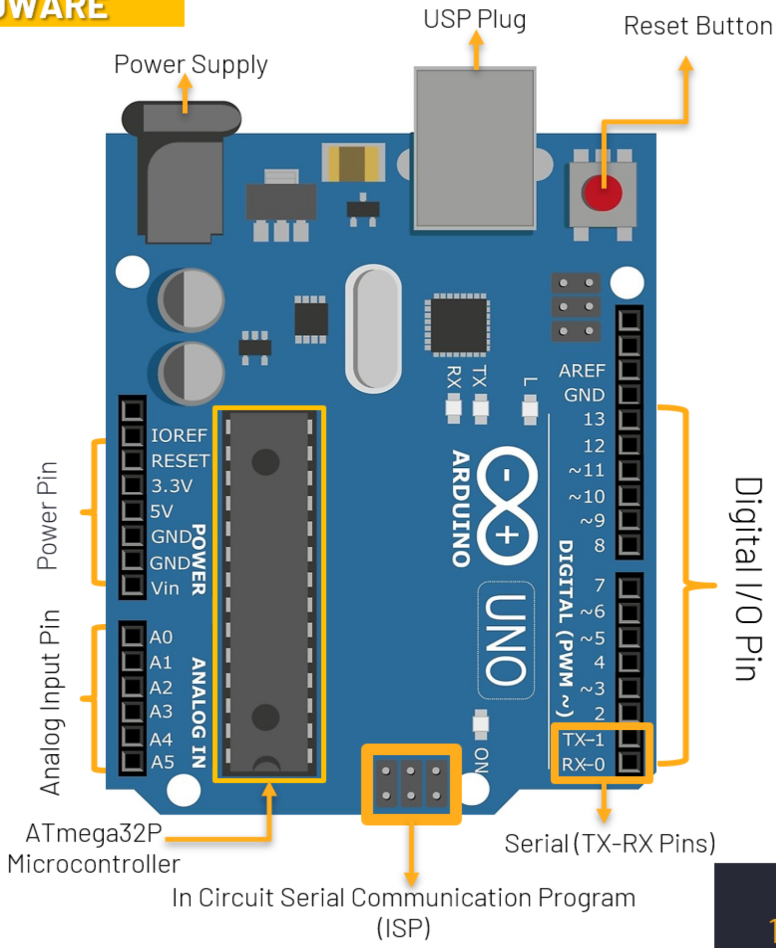
Arduino UNO Key Points :



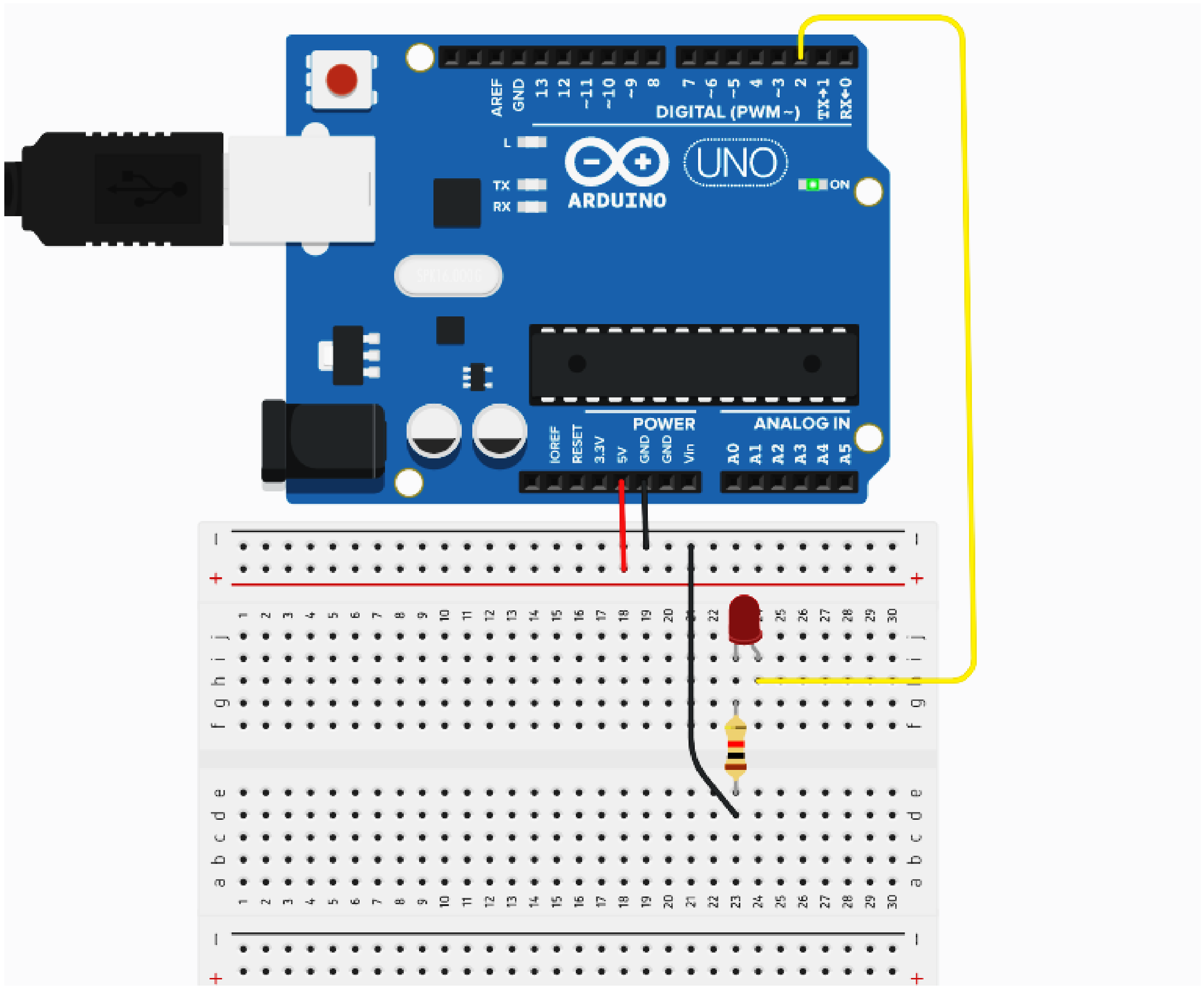
- Digital Pins : 0 to 13
- Analog Pins : A0 to A5
- Serial Communication Pins : TX & RX
- Voltage : 0 – 3.3V/5V

ARDUINO UNO HARDWARE

▪ Microcontroller	ATmegs328P
▪ Operating Voltage	5V
▪ Input Voltage (Recommended)	7-12V
▪ Input Voltage (Limit)	6-20V
▪ Digital I/O Pins	14
▪ PWM Digital I/O Pins	6
▪ Analog Input Pins	6
▪ DC Current per I/O Pin	20mA
▪ DC Current for 3.3V Pin	50mA
▪ Flash Memory	32KB
▪ SRAM	2KB
▪ EEPROM	1KB
▪ Clock Speed	16MHz
▪ Serial Communication Pin	0-RX Pin, 1-TX Pin



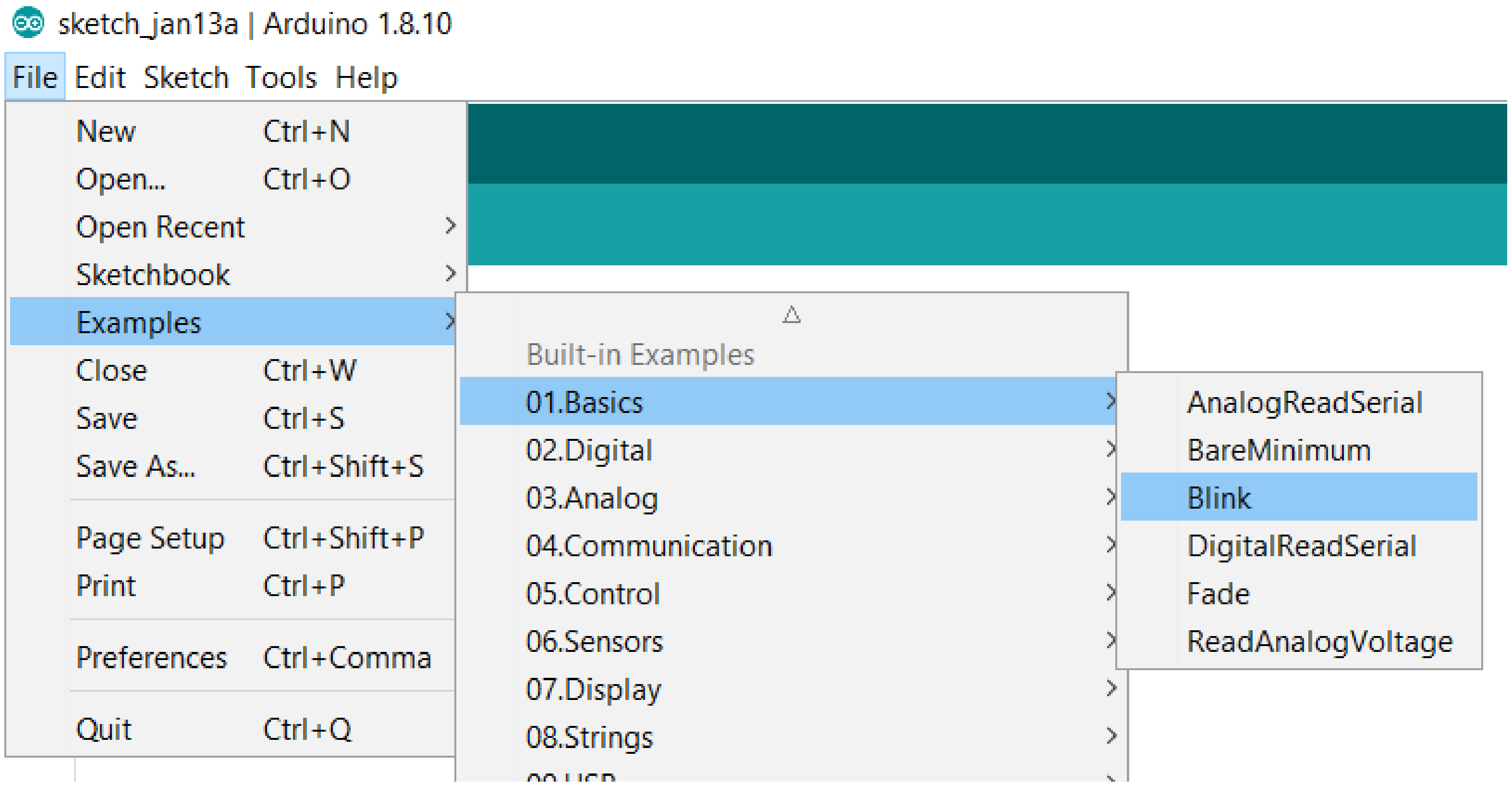
LED Blink :


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

LED Blink :

Step 1 : Open Arduino Software

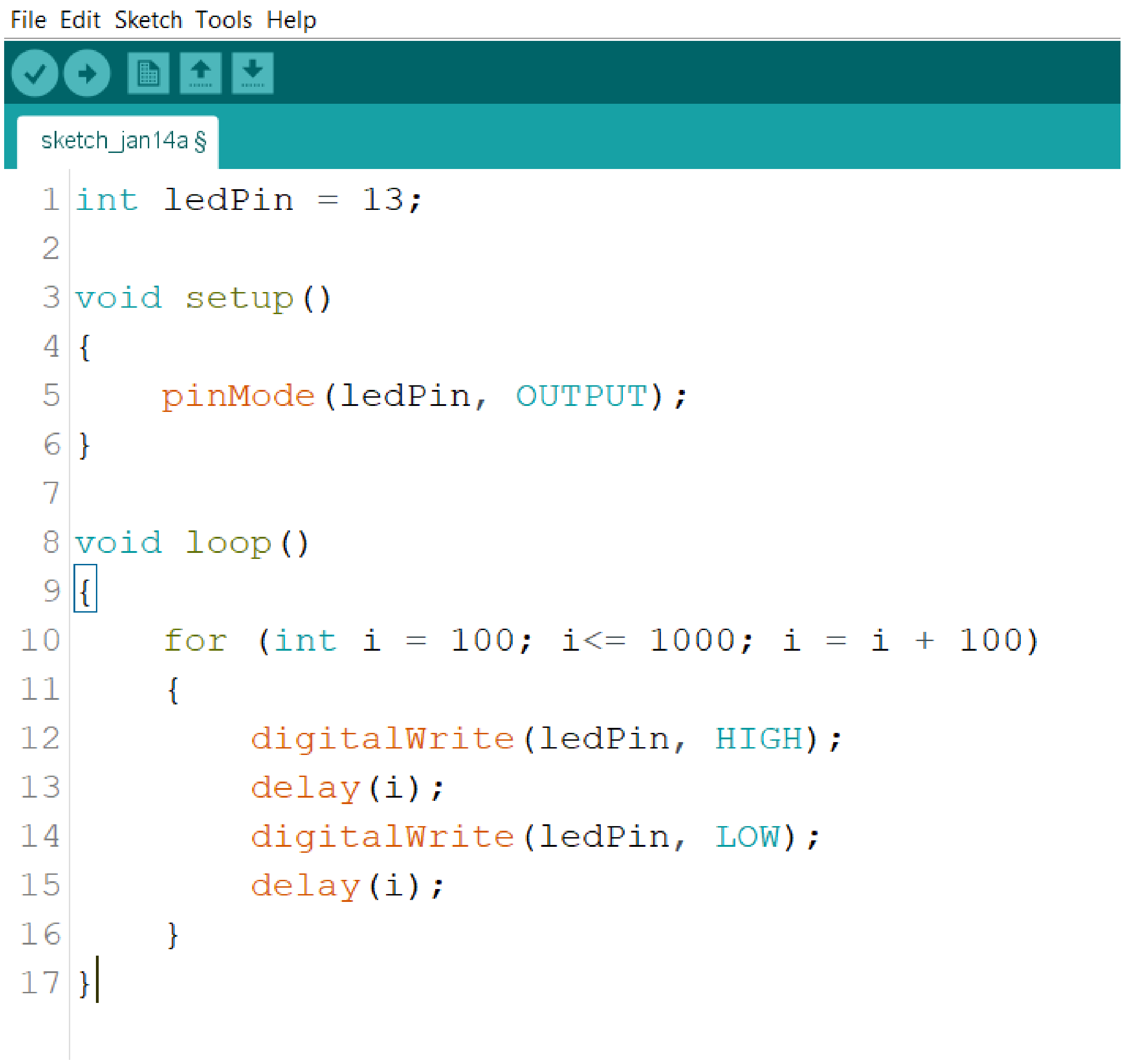
Step 2 : **File > Examples > Basics > Blink**



Code : 1

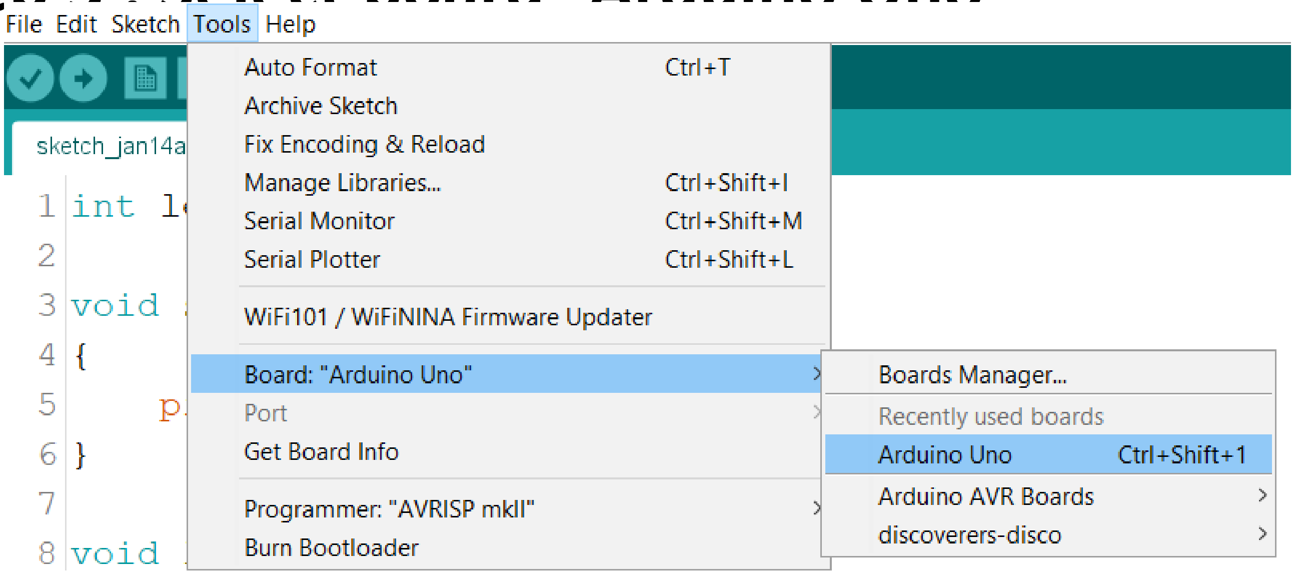
```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Code : 2



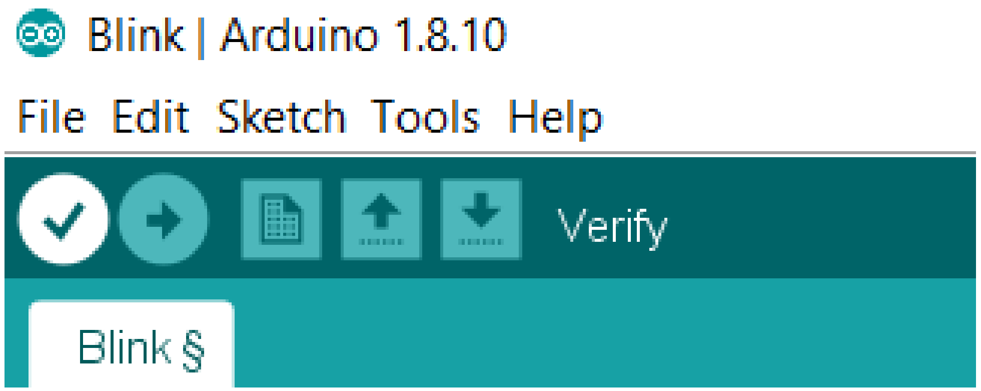
```
int ledPin = 13;

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

void loop()
{
     for (int i = 100; i<= 1000; i = i + 100)
     {
          digitalWrite(ledPin, HIGH);
          delay(i);
          digitalWrite(ledPin, LOW);
          delay(i);
     }
}
```

Step 3 : Select board “Arduino Uno”

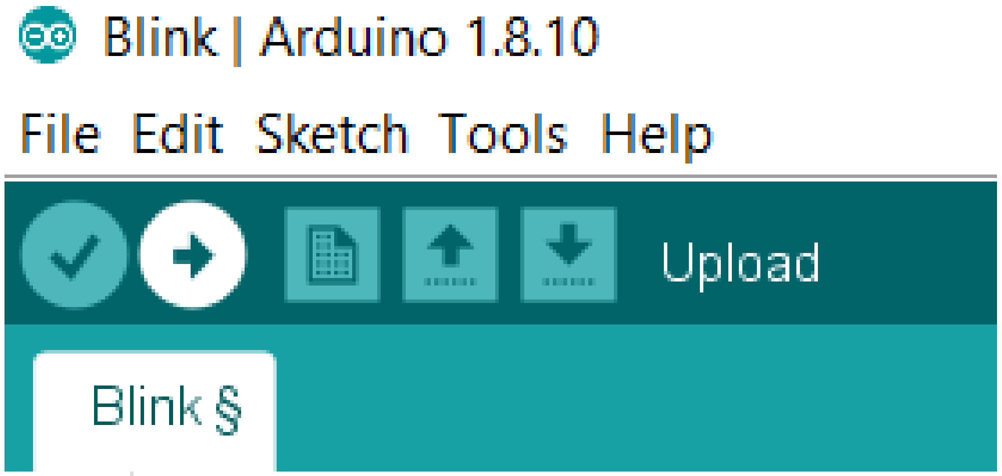


Step 4 : Select Port

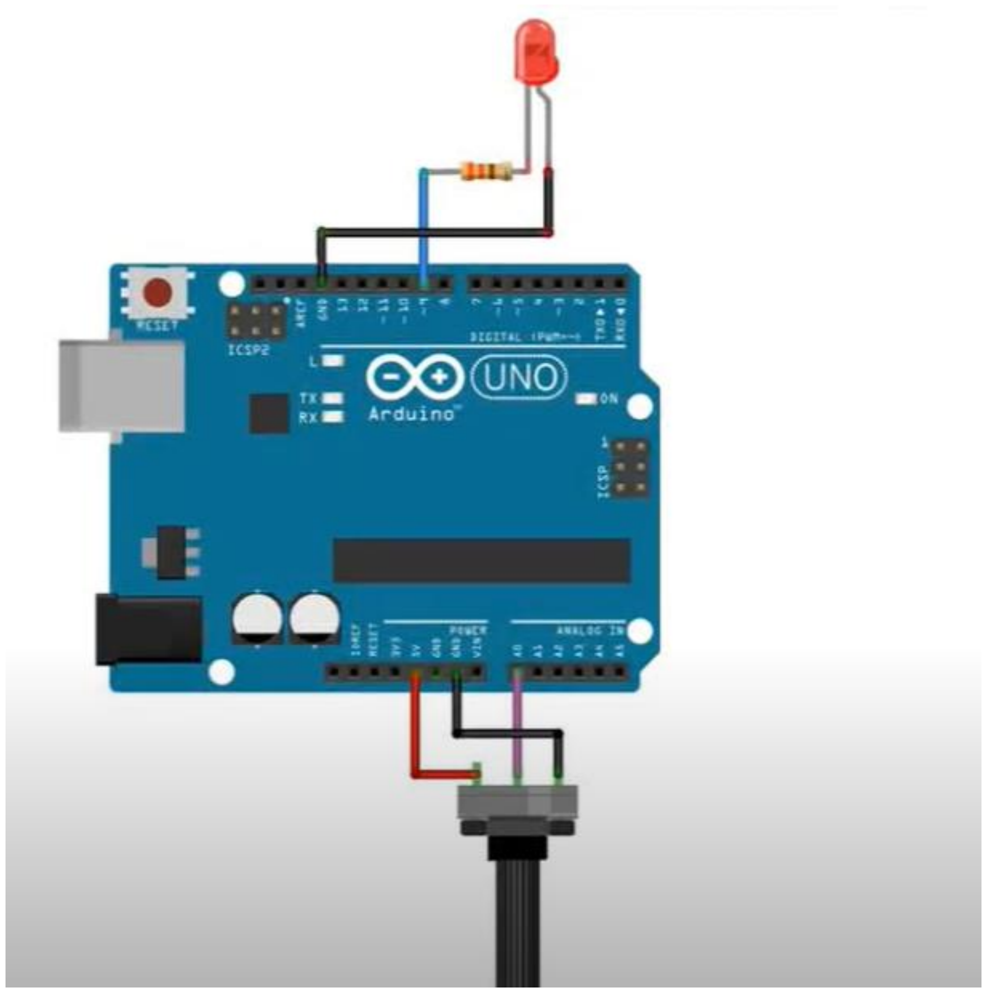
Step 5 : Compile/Verify the code to check for any errors.



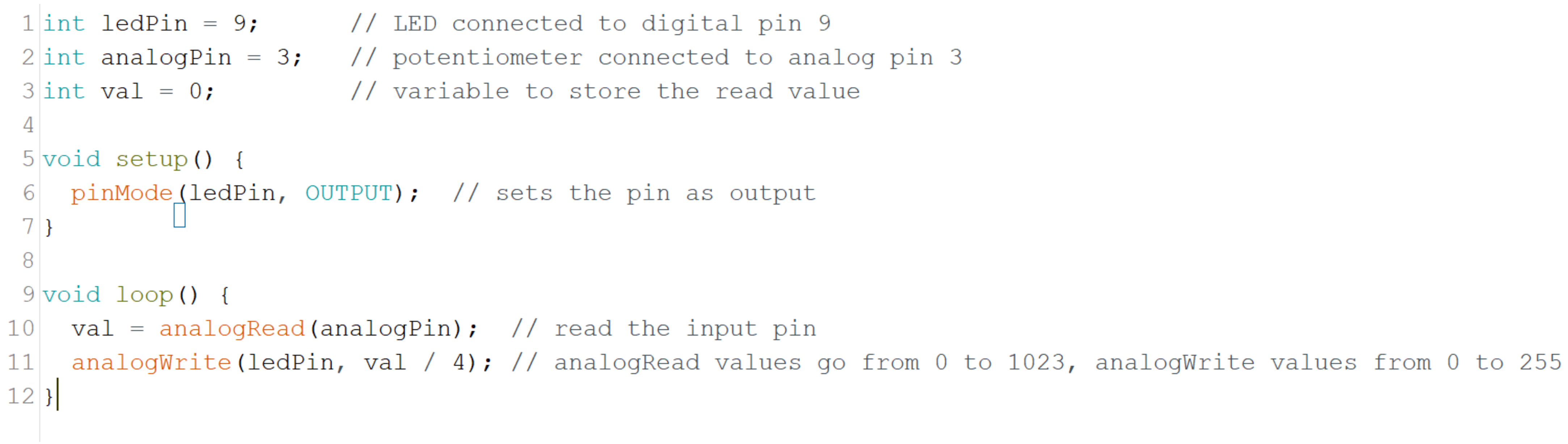
Step 6 : Upload your code to your Arduino Board.



Circuit

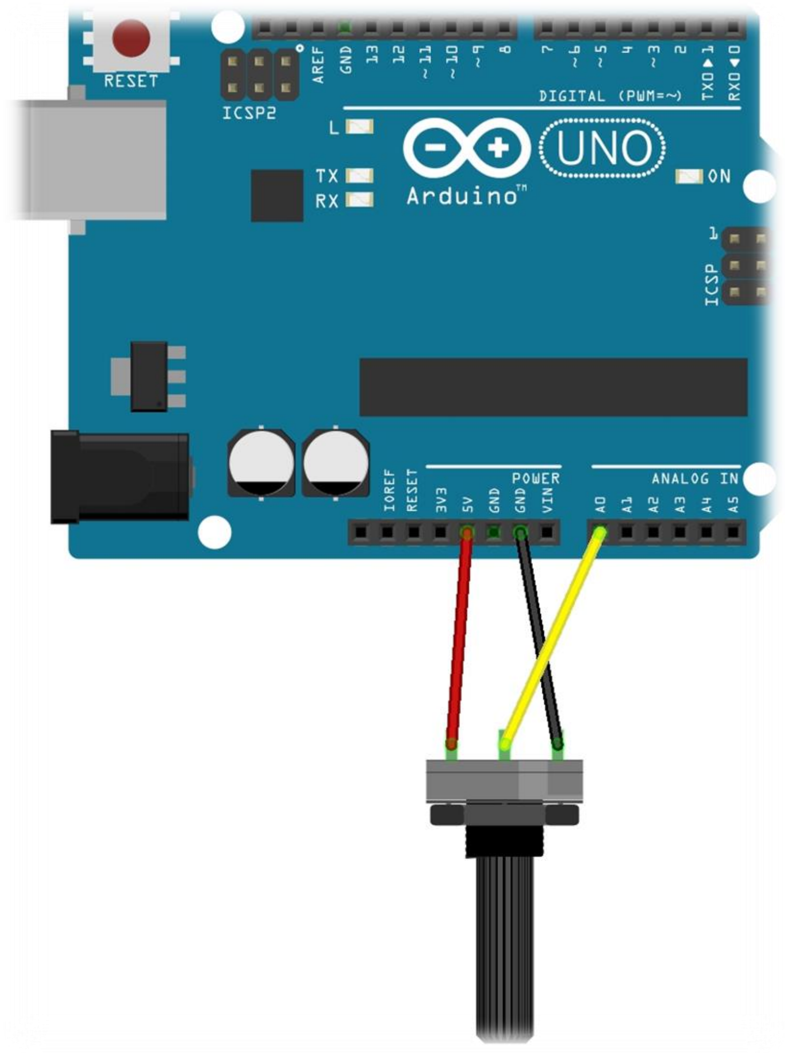


Code

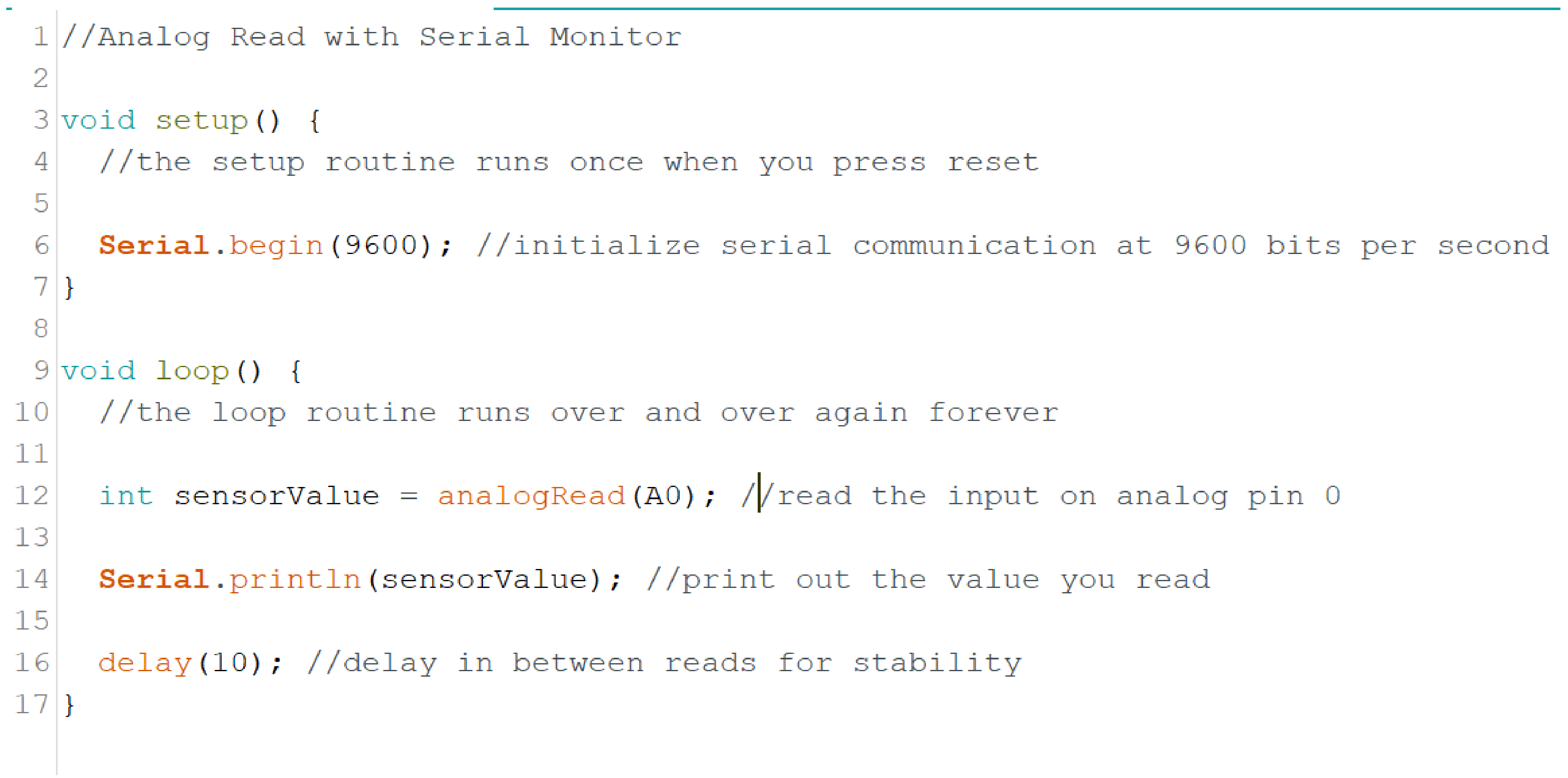


```
int ledPin = 9;      // LED connected to digital pin 9
int analogPin = 3;   // potentiometer connected to analog pin 3
int val = 0;         // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT);  // sets the pin as output
}

void loop() {
  val = analogRead(analogPin);  // read the input pin
  analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWrite values from 0 to 255
}
```

Analog Read with serial monitor



Code

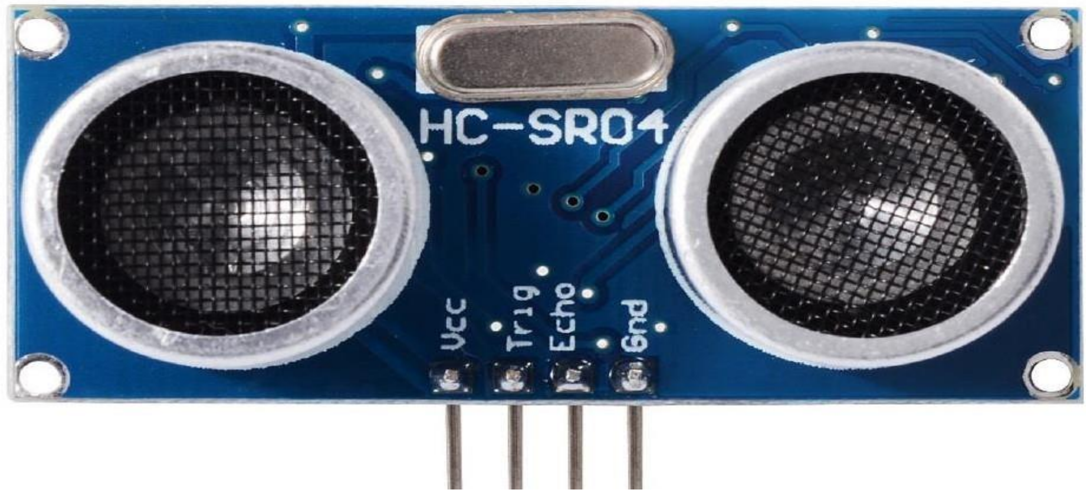


```
//Analog Read with Serial Monitor

void setup() {
  //the setup routine runs once when you press reset

  Serial.begin(9600); //initialize serial communication at 9600 bits per second
}

void loop() {
  //the loop routine runs over and over again forever

  int sensorValue = analogRead(A0); //read the input on analog pin 0

  Serial.println(sensorValue); //print out the value you read

  delay(10); //delay in between reads for stability
}
```

Fritzing Circuit Design Software

- Download Link: https://drive.google.com/drive/folders/1aP2bftMaBvLTqNrrNHctaV06Pl2BVocM?usp=sharing

Ultrasonic Sensor



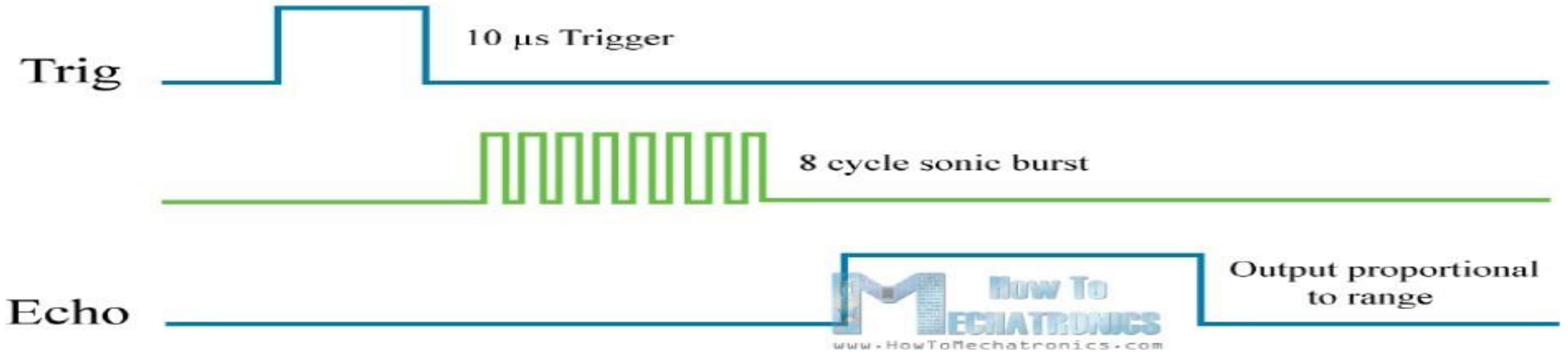
Specification:
- Power Supply :+5V DC
- Working Current: 15mA
- Effectual Angle: < 15 degree
- Ranging Distance : 2cm - 400 cm
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS

$$distance = \frac{speed\ of\ sound\ \times time\ taken}{2}$$

$$s = \frac{vt}{2}$$

এই সূত্র ব্যবহার করে খুব সহজেই আমরা বস্তুর দূরত্ব বের করতে পারব !

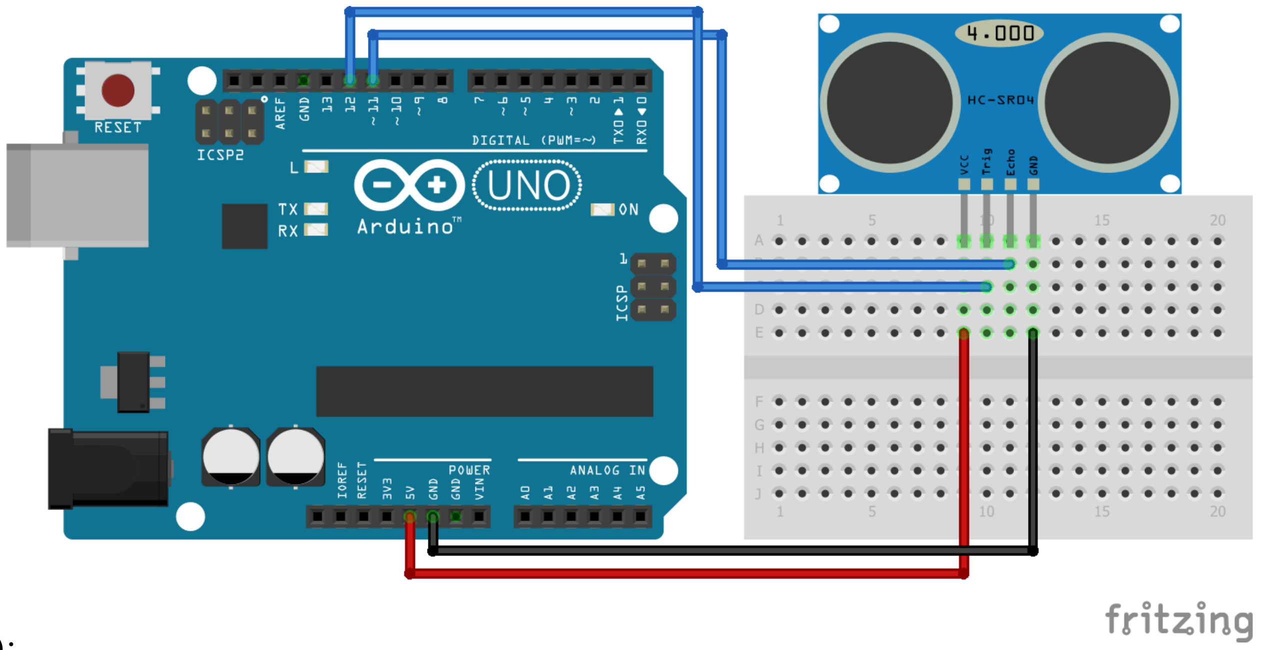
ট্রিগার ও ইকো পিন একসাথে সচল থাকতে পারবে না। প্রথমে ১০ মাইক্রোসেকেন্ড সময় জুড়ে ট্রিগার সচল থাকবে। এসময়ে ৮ টি সাইকেল পরিমাণ শব্দ ছুঁড়ে মারবে ট্রিগার এরপরে ইকো সচল হবে ও মেপে দেখবে কত সময় পরে শব্দ আবার ফিরে এসেছে।



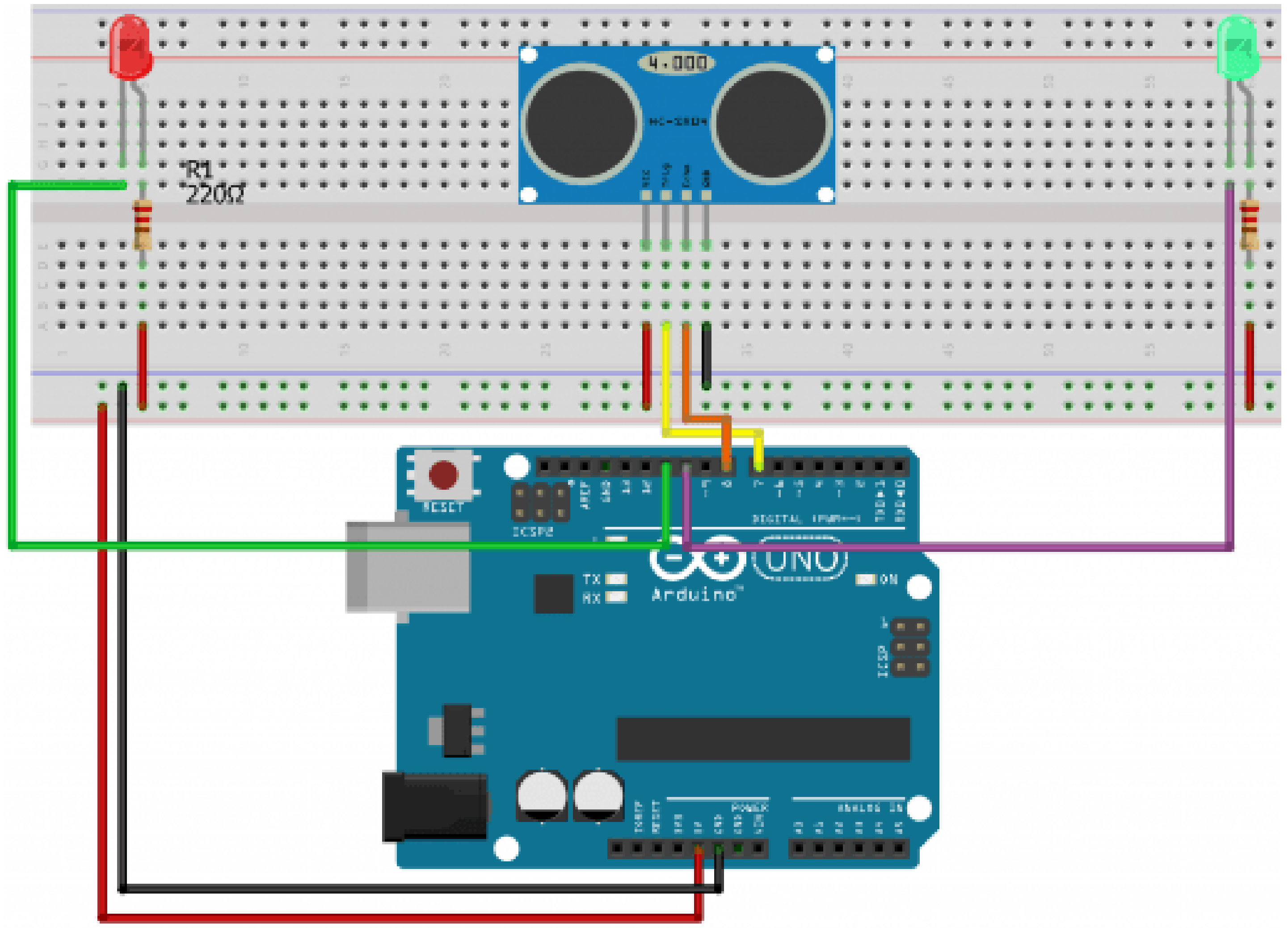
Distance Measurement using Ultrasonic Sensor

```
const int trigPin = 12;
const int echoPin = 11;
long duration;
int distance;

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

void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration*0.034/2;
  Serial.print("Distance: ");
  Serial.println(distance);
}
```

Ultrasonic Sensor



Task: Build and code a circuit to control the color of an LED using a distance sensor. If an object is less than 20 cm close to the distance sensor, the LED will be red, but if the object is over 20 cm far from the distance sensor then the LED will be green.



Code

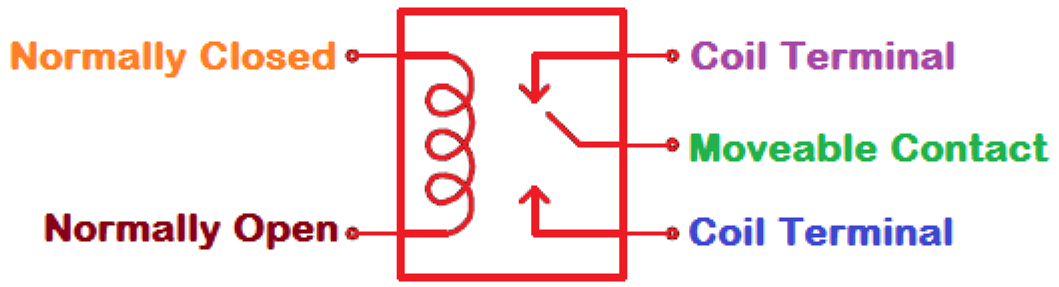
```
int trigPin = 7; //Define the pins that you will work with
int echoPin = 8;
int LEDR = 10;
int LEDV = 11;
float Speed = 0.0343; // Sound speed at cm/us
long duration, distance;
void setup() { pinMode(trigPin, OUTPUT); //Define digital pin 7 as an output
pinMode(echoPin, INPUT); //Define digital pin 8 as an input
pinMode(LEDR, OUTPUT); //Define digital pin 10 as an output
pinMode(LEDV, OUTPUT); //Define digital pin 11 as an output
digitalWrite (LEDR , LOW); // Define digital pin 10 in a low status
digitalWrite (LEDV , LOW); /Define digital pin 11 in a low status }
void loop() { digitalWrite(trigPin, LOW); // Make sure that the TRIG is deactivated
delayMicroseconds(2); // Make sure that the TRIG is in LOW
digitalWrite(trigPin, HIGH); // Activate the output pulse
delayMicroseconds(10); // Wait for 10µs, the pulse remains active during this time
digitalWrite(trigPin, LOW); //Stop the pulse and wait for ECHO
duration = pulseIn(echoPin, HIGH) ;
distance = Speed* duration / 2;
if ( distance < 20)
{
digitalWrite (LEDR , HIGH); //If the sensor detects a distances less than 20 cm the red LED turns on
digitalWrite (LEDV , LOW); //and turns off the green LED }
else{ // otherwise
digitalWrite (LEDR , LOW); // turn off the red LED
digitalWrite (LEDV , HIGH); //turn on the green LED }
}
```

Relay

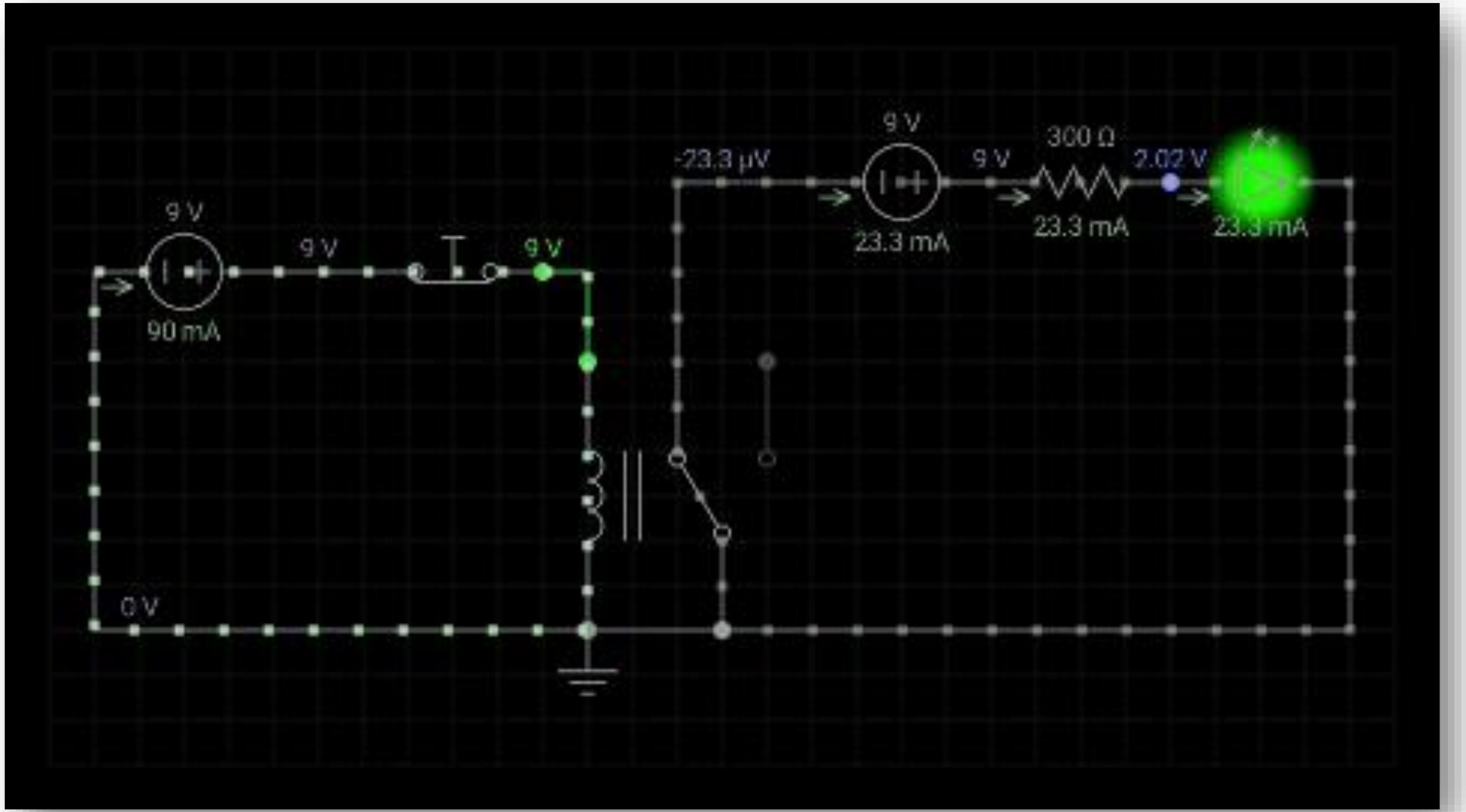
A Relay is an electromechanical device that can be used to make or break an electrical connection

"A relay is an electrically operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals."

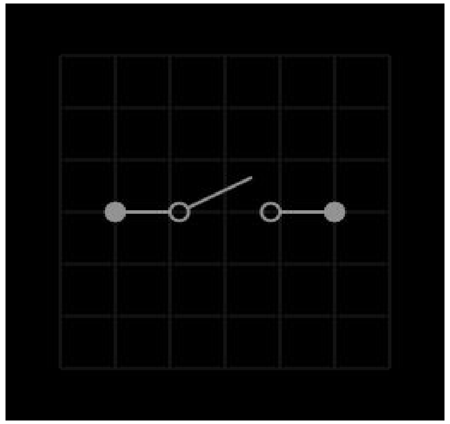
Relay Internal Structure

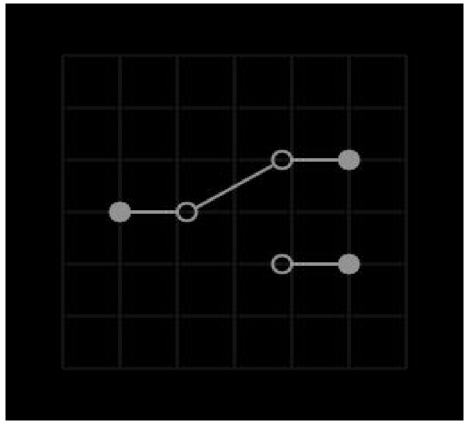


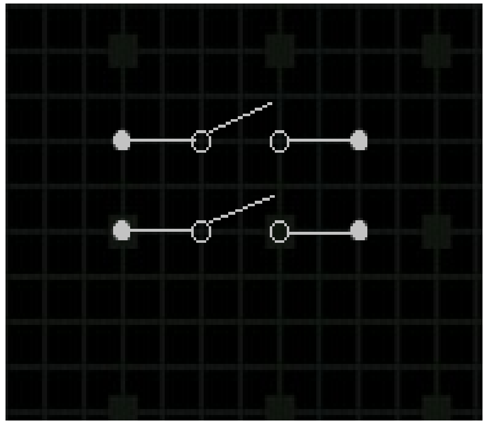
Working Principle of Relay

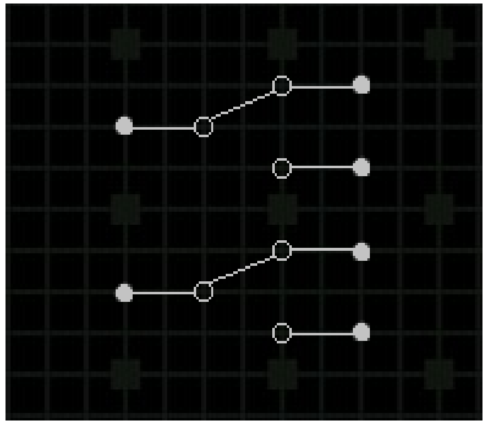


Types of Relay


SPDT

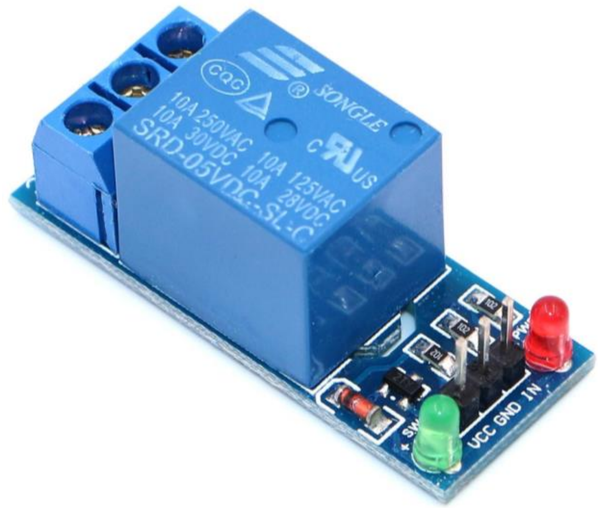

DPST


DPDT

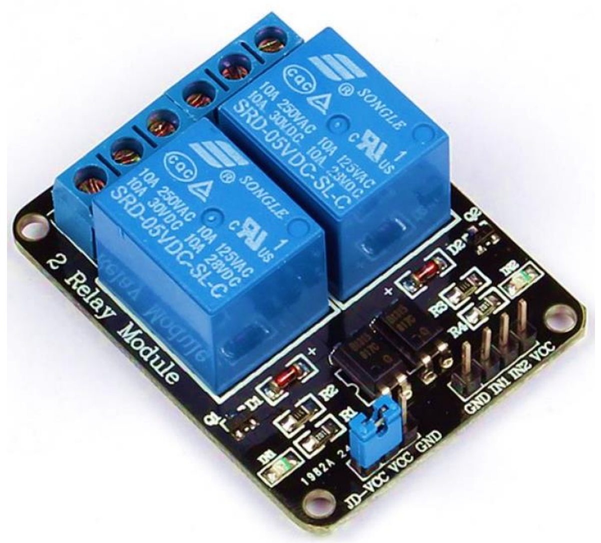

DPDT

1. **Single pole, double throw Relay (SPDT)**
2. **Double pole, single throw Relay (DPST)**
3. **Double pole, double throw Relay (DPDT)**
4. **Double pole, double throw Relay (DPDT)**

Relay Module



5V Single Channel

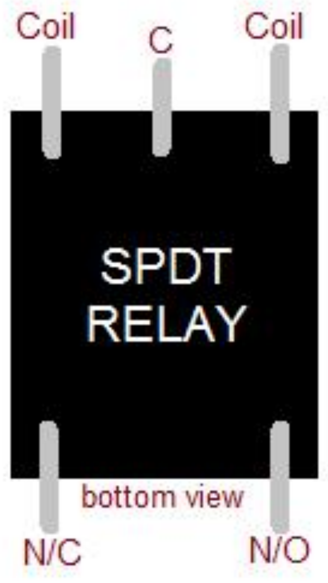


5V Double Channel

5V 4 Channel

5V 8 Channel

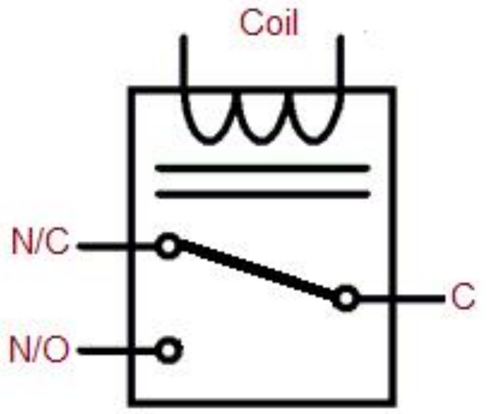
SPDT RELAY



Pin Details

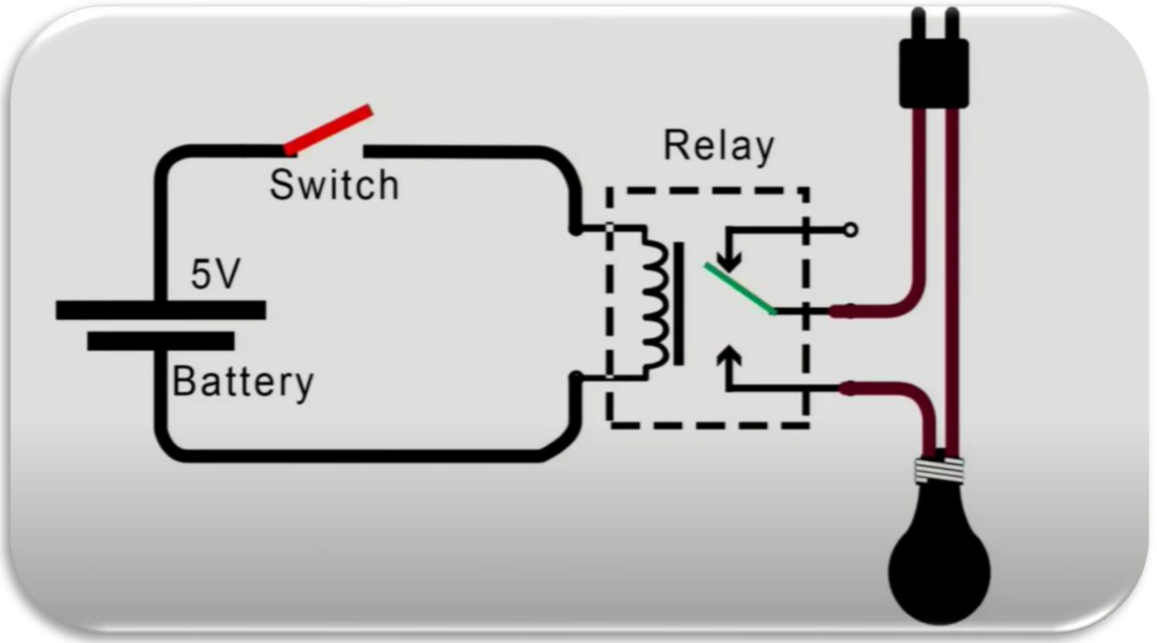
Physical View
*may vary

Coil - Electromagnetic Coil Terminal
C - Common Terminal
N/C - Normally Close Terminal
N/O - Normally Open Terminal

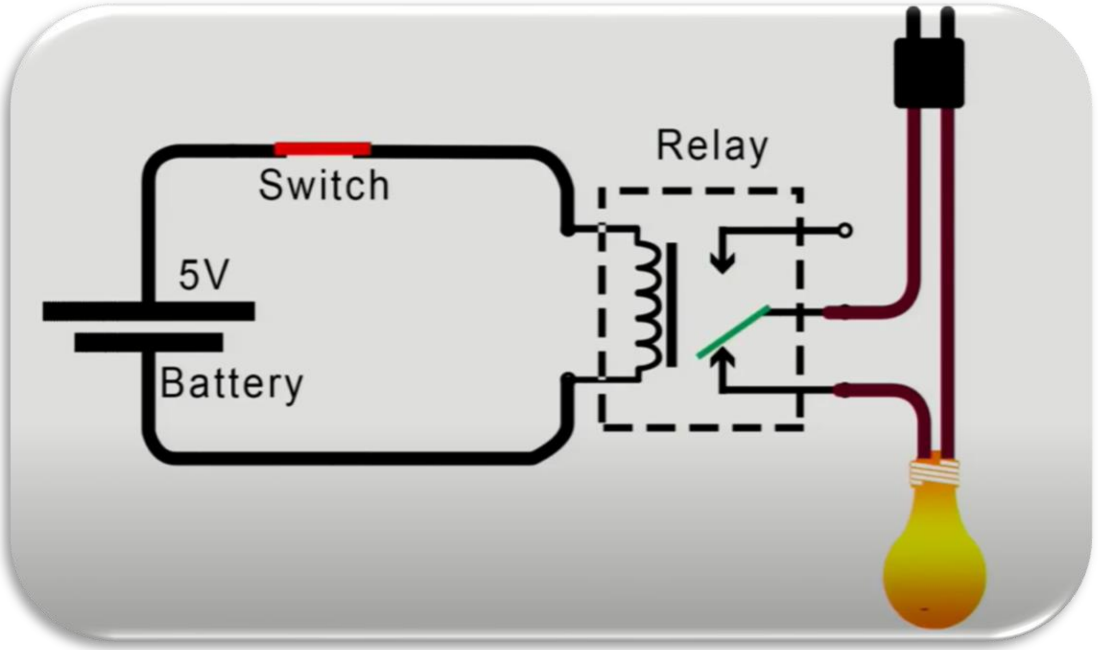


SPDT Symbol

Light Control Using Relay



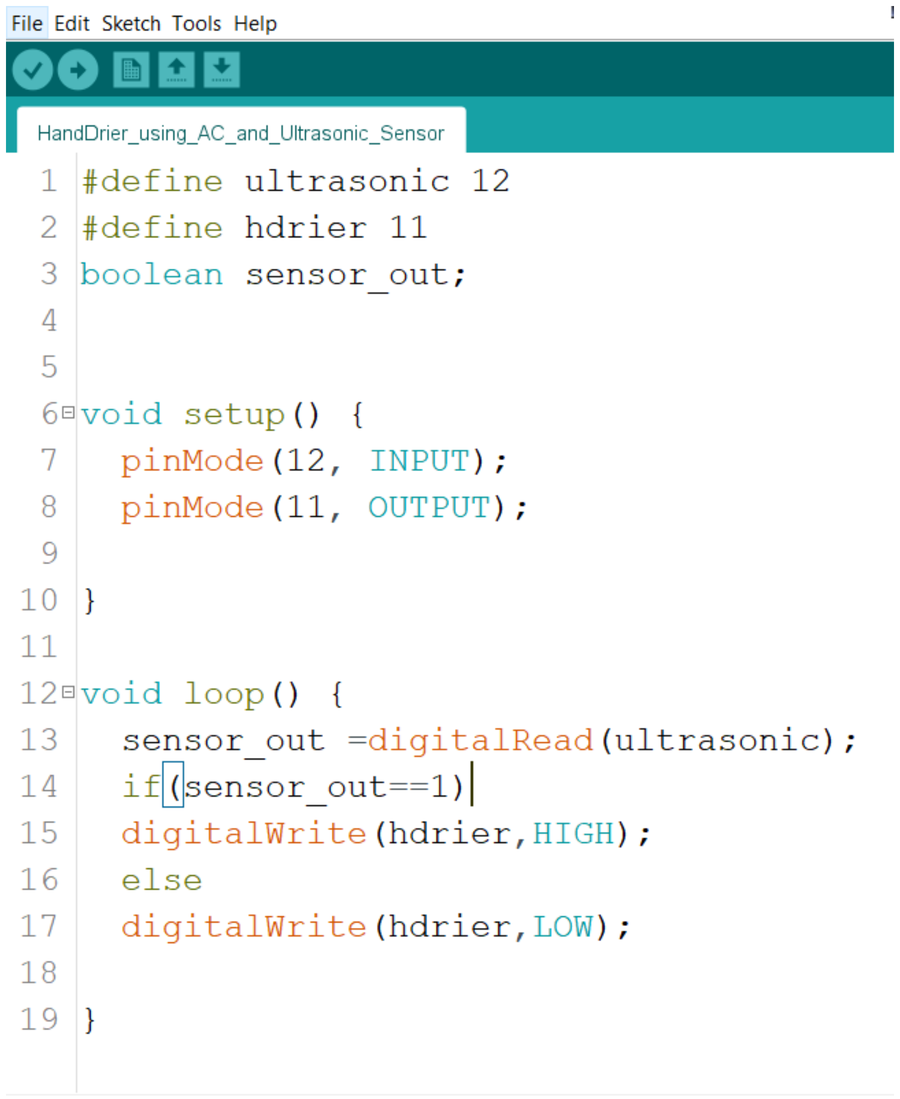
OPEN



CLOSE

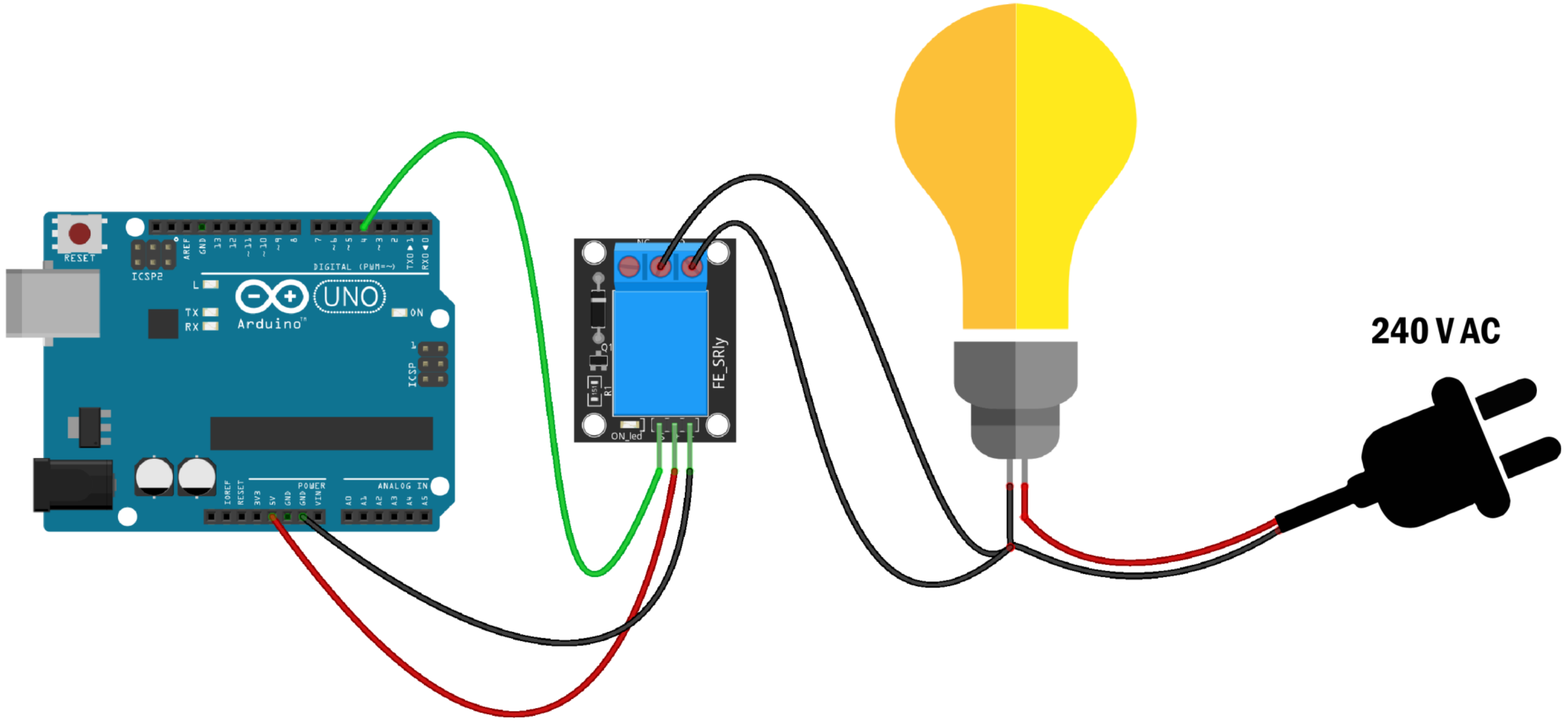
Code

```
#define ultrasonic 12
#define hdrier 11
boolean sensor_out;


void setup() {
  pinMode(12, INPUT);
  pinMode(11, OUTPUT);

}

void loop() {
  sensor_out =digitalRead(ultrasonic);
  if(sensor_out==1)
  digitalWrite(hdrier,HIGH);
  else
  digitalWrite(hdrier,LOW);

}
```


File Edit Sketch Tools Help
HandDrier_using_AC_and_Ultrasonic_Sensor

```
#define ultrasonic 12
#define hdrier 11
boolean sensor_out;


void setup()  {
  pinMode(12, INPUT);
  pinMode(11, OUTPUT);

}

void loop()  {
  sensor_out =digitalRead(ultrasonic);
  if(sensor_out==1)
  digitalWrite(hdrier,HIGH);
  else
  digitalWrite(hdrier,LOW);

}
```

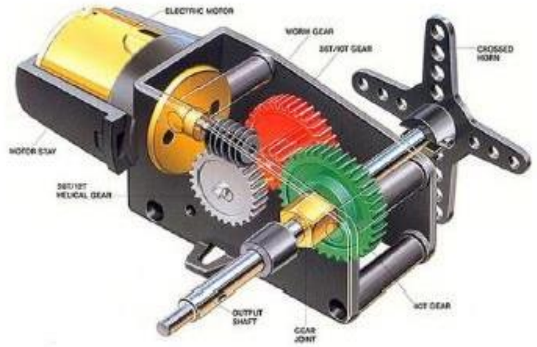
Circuit Diagram

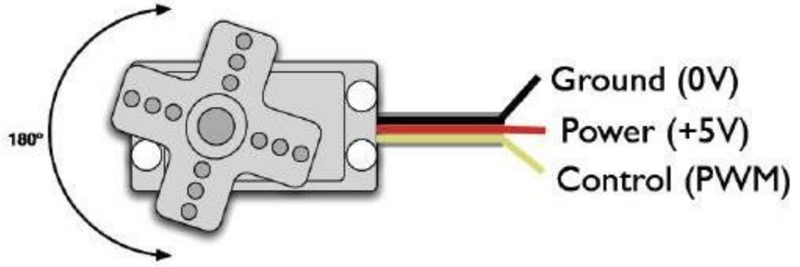


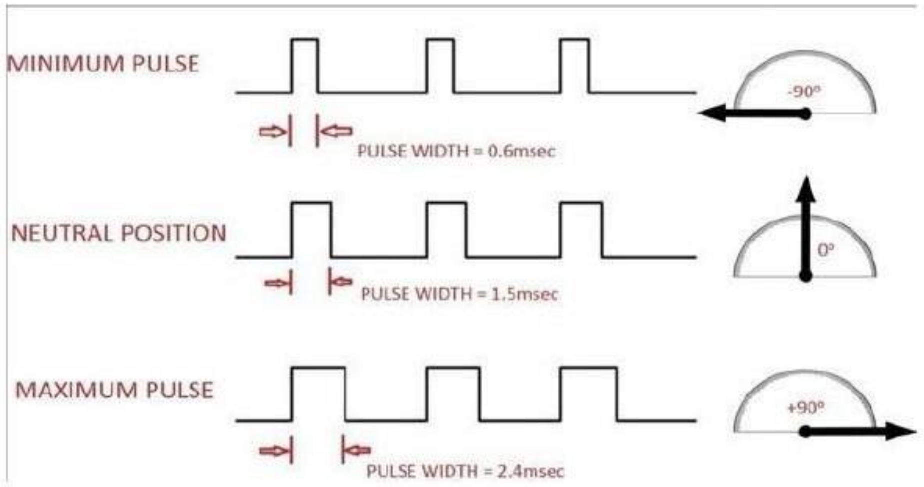
- সারভো মোটর

TOWER PRO
MICRO SERVO 9G
SG90

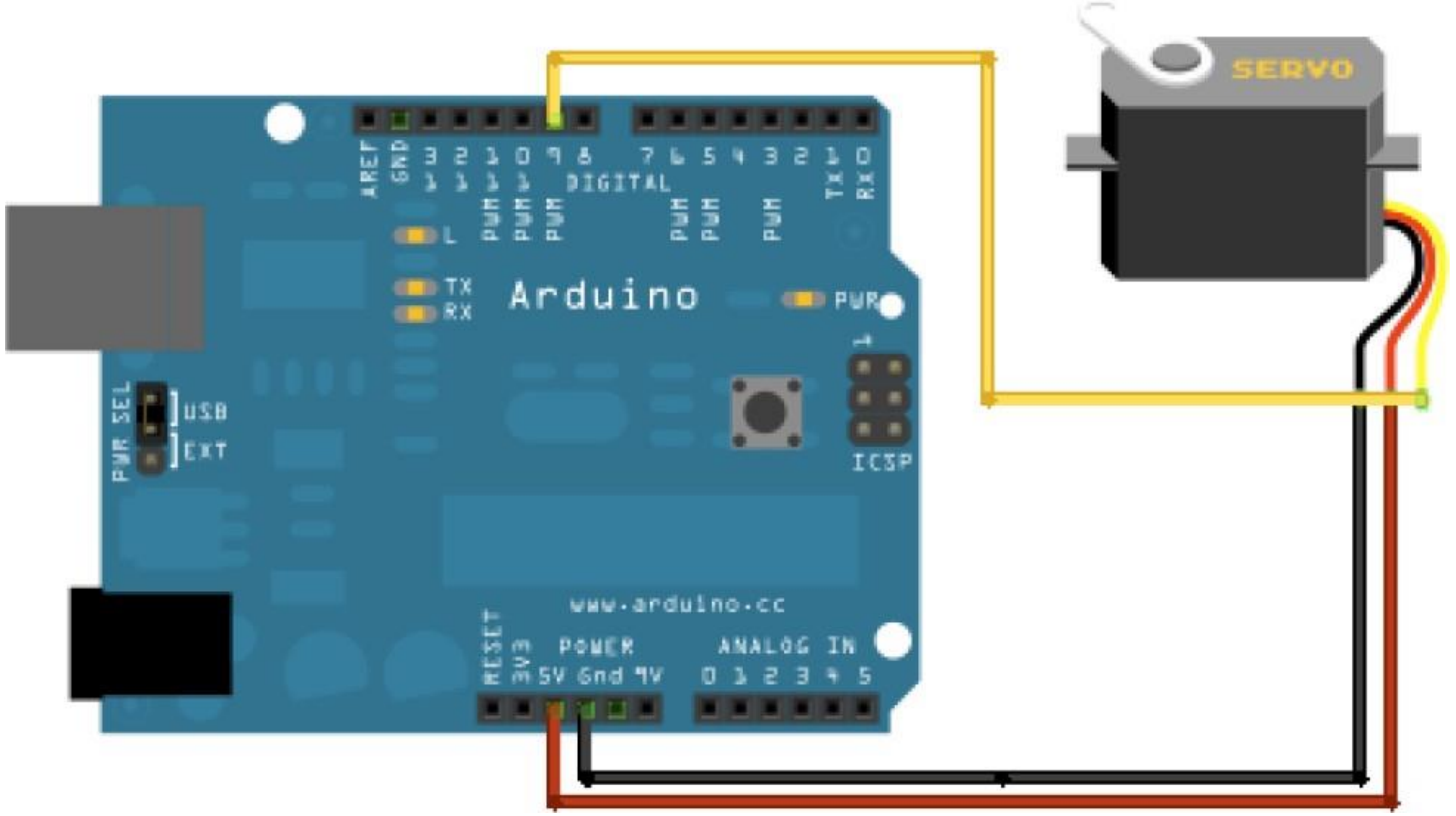
HITEC
HS-422

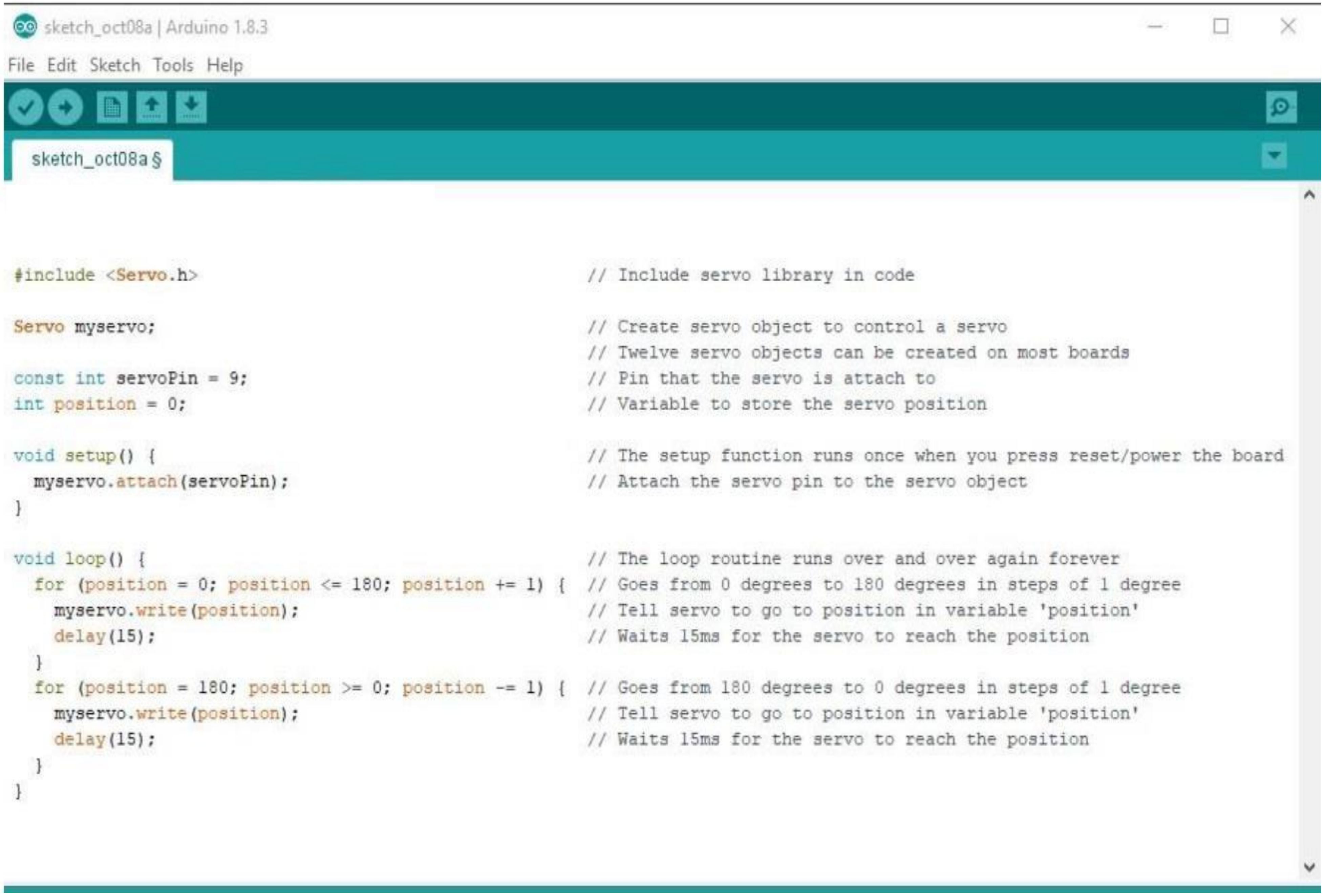

ELECTRIC MOTOR
WORM GEAR
36T/10T GEAR
CROSSED HORN
MOTOR STAY
36T/10T HELICAL GEAR
OUTPUT SHAFT
GEAR JOINT
10T GEAR


180°
Ground (0V)
Power (+5V)
Control (PWM)


MINIMUM PULSE
PULSE WIDTH = 0.6msec
-90°
NEUTRAL POSITION
PULSE WIDTH = 1.5msec
0°
MAXIMUM PULSE
PULSE WIDTH = 2.4msec
+90°

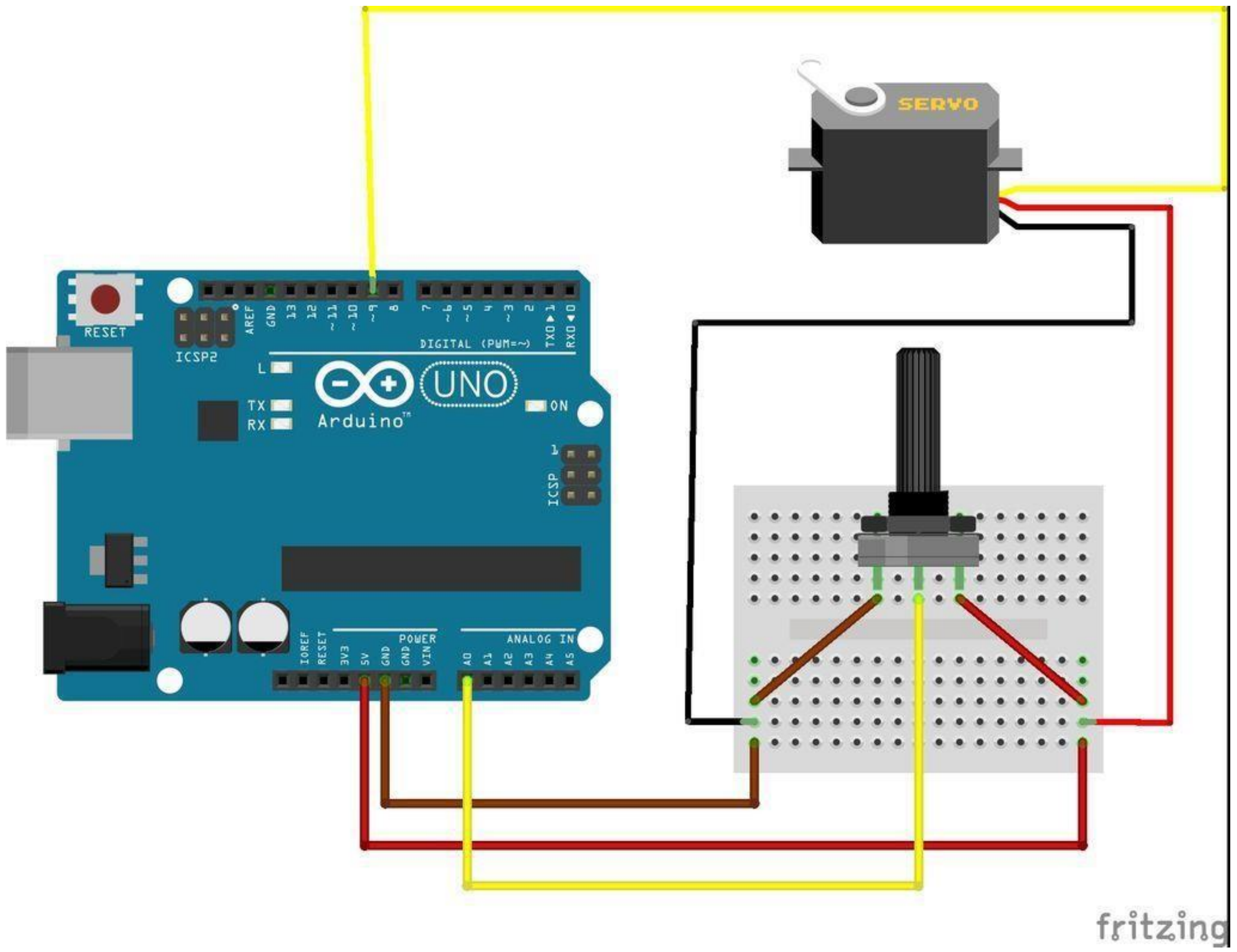
- সারভো মোটর স্বয়ংক্রিয় নিয়ন্ত্রণ

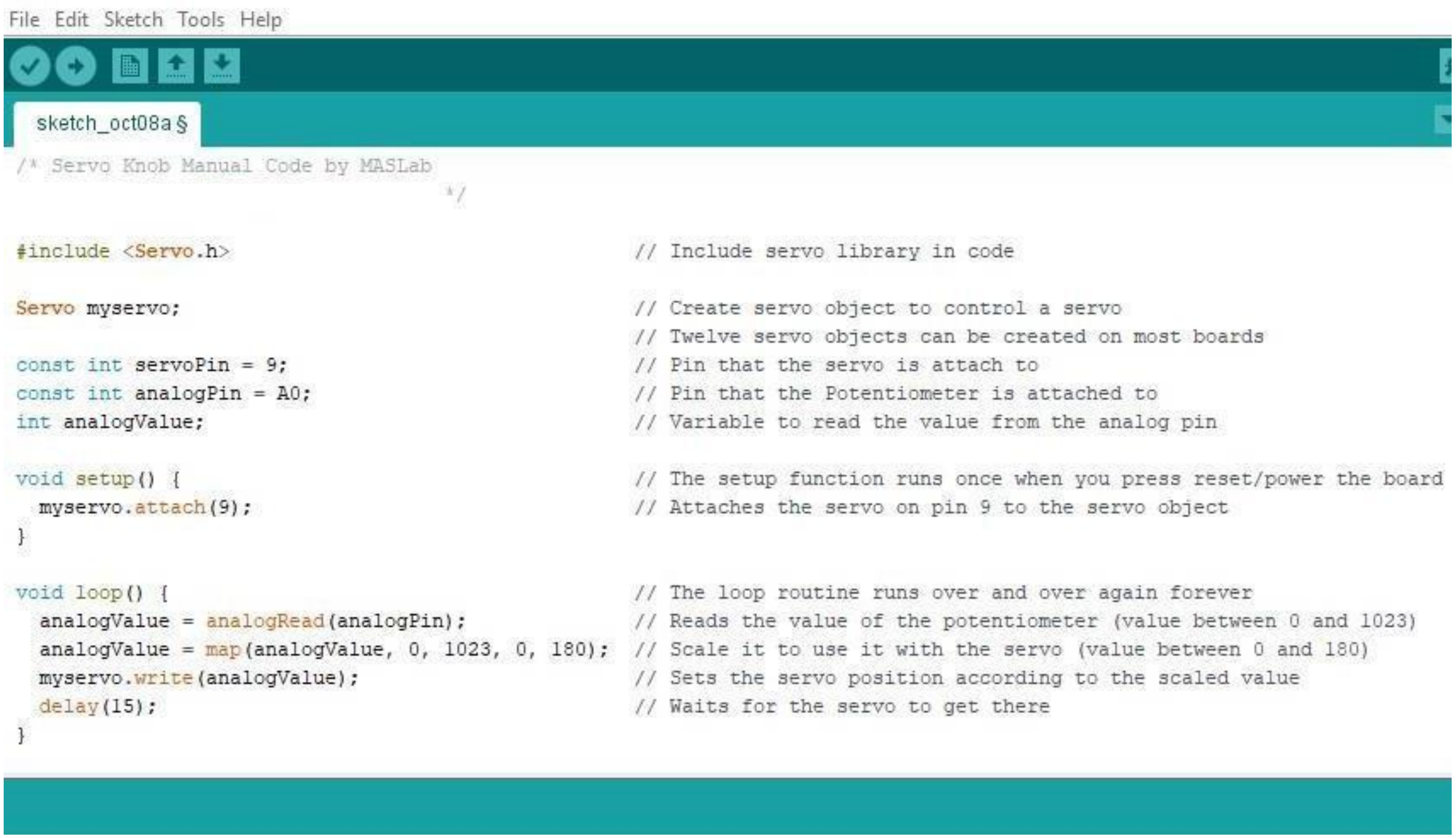




```
#include <Servo.h>                                     // Include servo library in code

Servo myservo;                                         // Create servo object to control a servo
                                                       // Twelve servo objects can be created on most boards
const int servoPin = 9;                                // Pin that the servo is attach to
int position = 0;                                      // Variable to store the servo position

void setup() {                                         // The setup function runs once when you press reset/power the board
  myservo.attach(servoPin);                            // Attach the servo pin to the servo object
}

void loop() {                                          // The loop routine runs over and over again forever
  for (position = 0; position <= 180; position += 1) { // Goes from 0 degrees to 180 degrees in steps of 1 degree
    myservo.write(position);                           // Tell servo to go to position in variable 'position'
    delay(15);                                         // Waits 15ms for the servo to reach the position
  }
  for (position = 180; position >= 0; position -= 1) { // Goes from 180 degrees to 0 degrees in steps of 1 degree
    myservo.write(position);                           // Tell servo to go to position in variable 'position'
    delay(15);                                         // Waits 15ms for the servo to reach the position
  }
}
```

পটেনশিওমিটার দিয়ে সারভো মোটর নিয়ন্ত্রণ





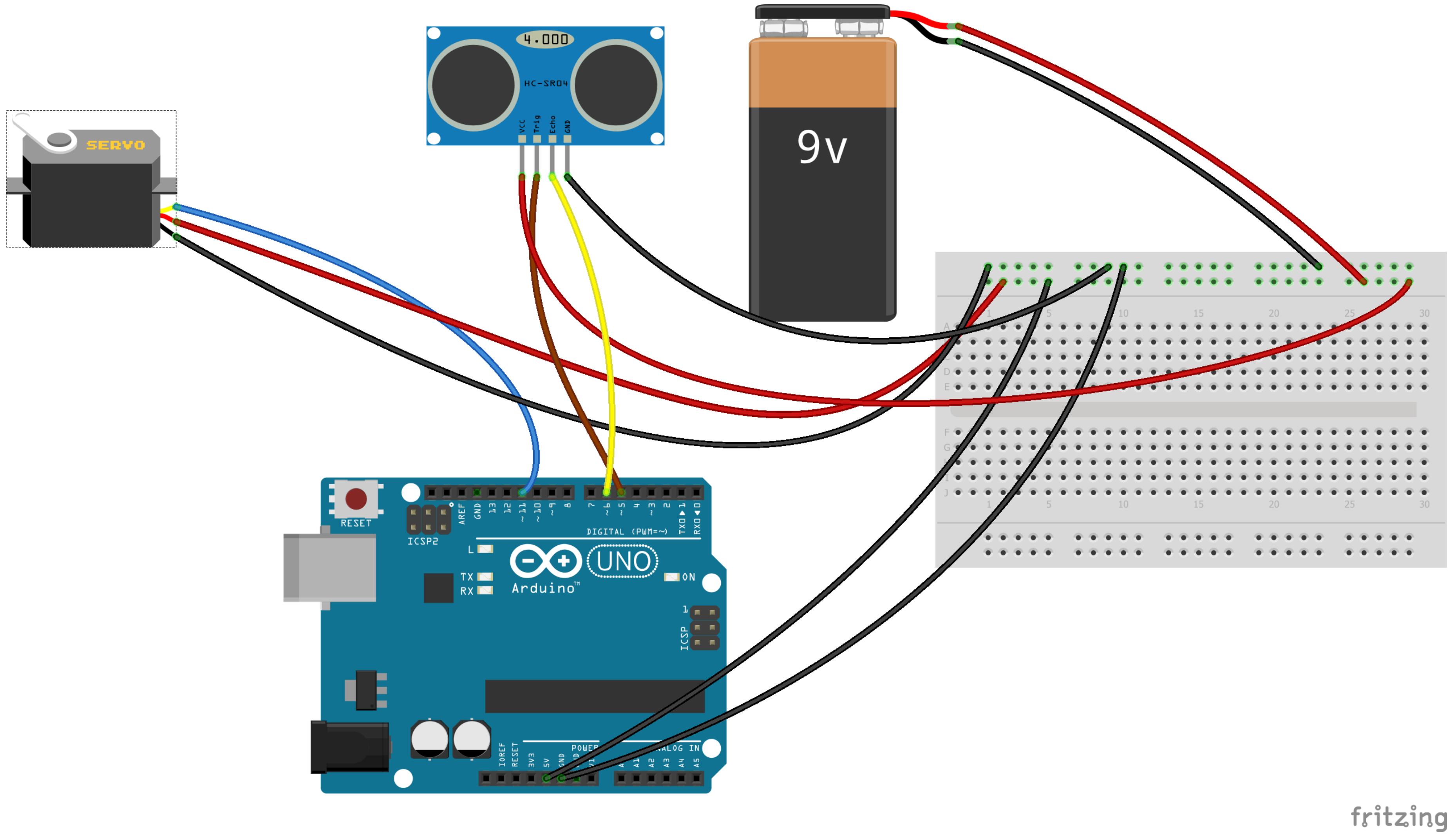
```
/* Servo Knob Manual Code by MASLab
                                                  */

#include <Servo.h>                                          // Include servo library in code

Servo myservo;                                              // Create servo object to control a servo
                                                            // Twelve servo objects can be created on most boards
const int servoPin = 9;                                     // Pin that the servo is attach to
const int analogPin = A0;                                   // Pin that the Potentiometer is attached to
int analogValue;                                            // Variable to read the value from the analog pin

void setup() {                                              // The setup function runs once when you press reset/power the board
  myservo.attach(9);                                        // Attaches the servo on pin 9 to the servo object
}

void loop() {                                               // The loop routine runs over and over again forever
  analogValue = analogRead(analogPin);                      // Reads the value of the potentiometer (value between 0 and 1023)
  analogValue = map(analogValue, 0, 1023, 0, 180);          // Scale it to use it with the servo (value between 0 and 180)
  myservo.write(analogValue);                               // Sets the servo position according to the scaled value
  delay(15);                                                // Waits for the servo to get there
}
```

Smart Dustbin

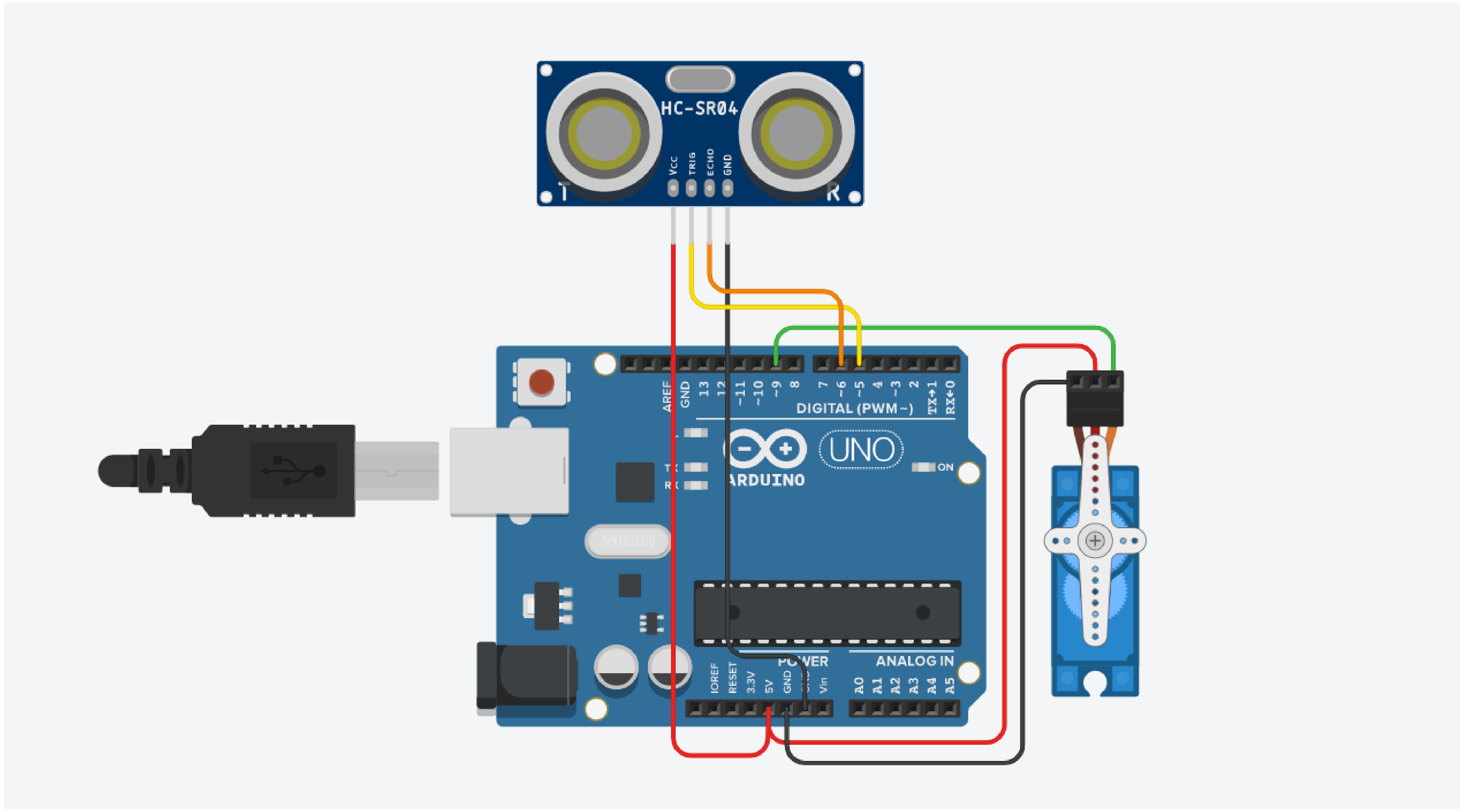
Components

- Dustbin Basket
- Arduino(Microcontroller any)
- Servo Motor (Mini-SG90)
- Ultrasonic Sensor
- Connecting wire(Male to Male, Male to Female)
- Battery (9V)

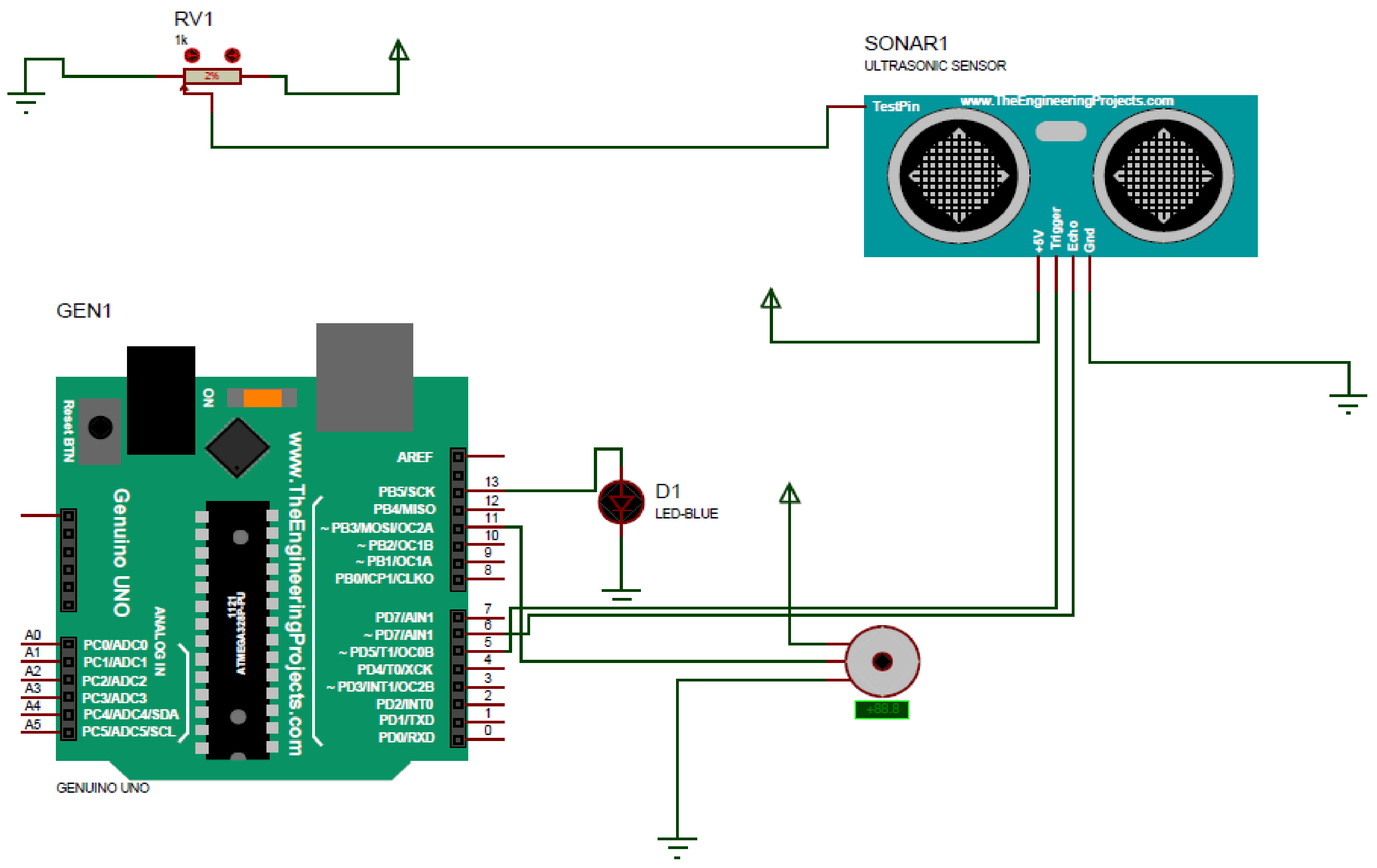
Circuit Diagram



Circuit Diagram in Tinkercad



Circuit Diagram in Proteus



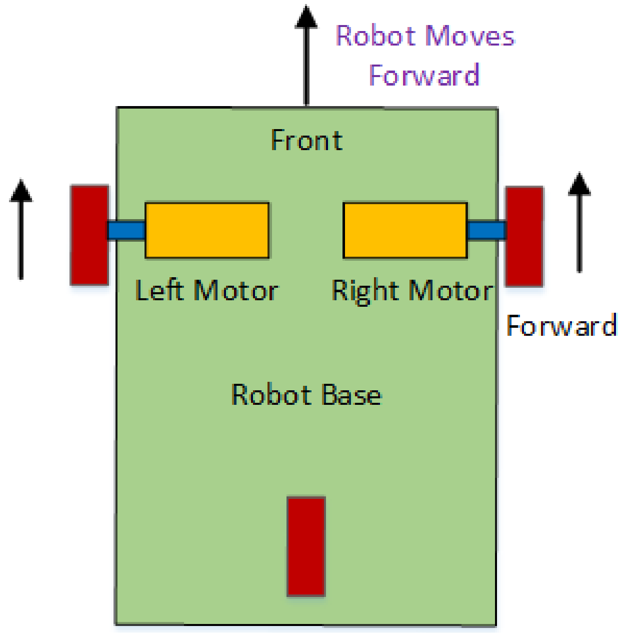
CODE

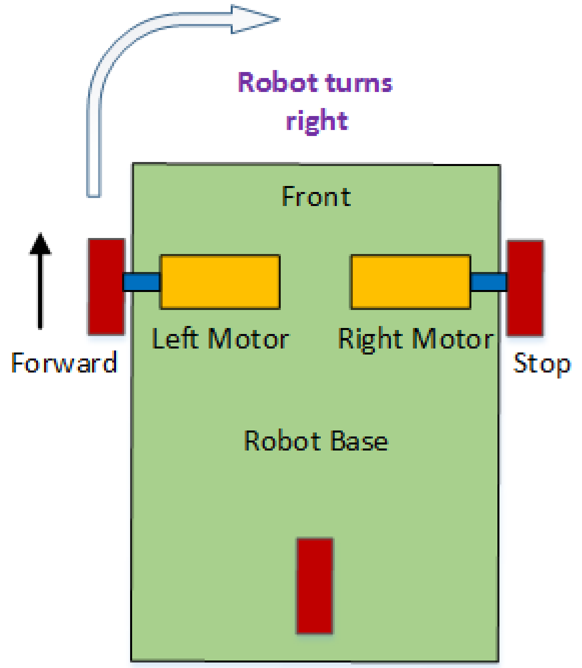
```
#include<Servo.h>
Servo myservo;

int pos = 20;
const int trigPin = 5;
const int echoPin = 6;
const int led = 13;

long duration;
float distance;

  void setup() {
 myservo.attach(11);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(led, OUTPUT);
 myservo.write(pos);
}

  void loop() {
 //Serial.begin(9600);
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 duration =pulseIn(echoPin, HIGH);
 distance = 0.034*(duration/2);
 Serial.println(distance);
 if(distance<27)
 {
  digitalWrite(led, HIGH);
  myservo.write(pos+160);
  delay(1000);
 }
 else
 {
  digitalWrite(led, LOW);
  myservo.write(pos);
 }
 delay(300);
}
```

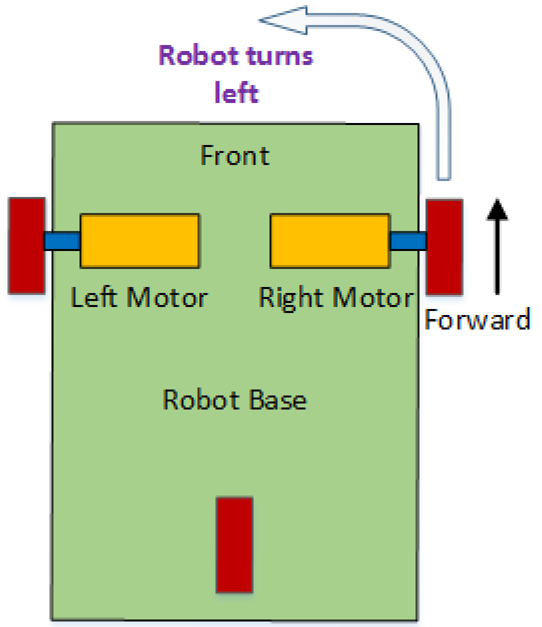
Day 9

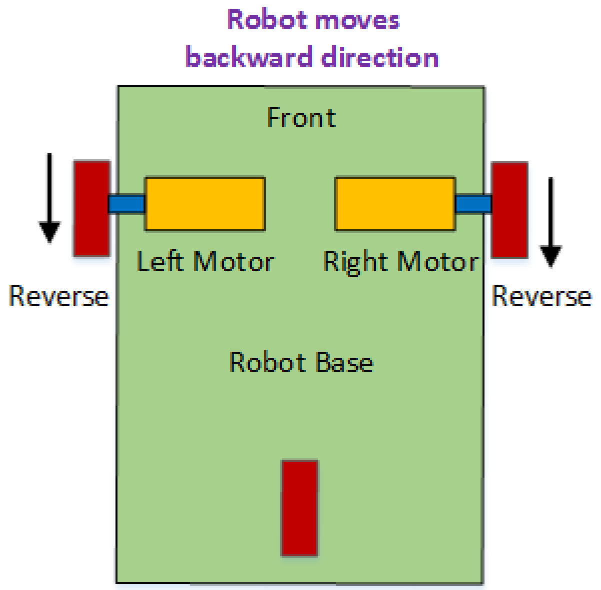
Motor Driver Module and DC Motor

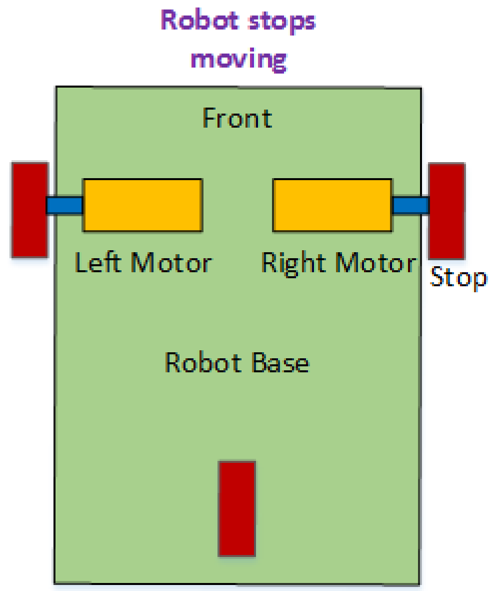
Moving a Robocar

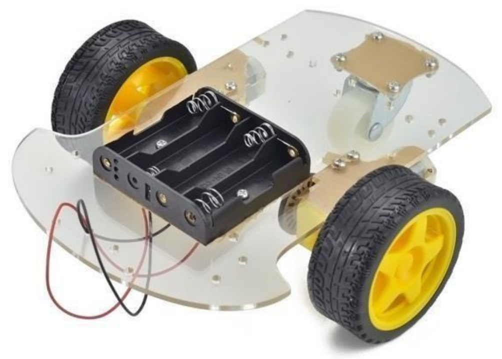




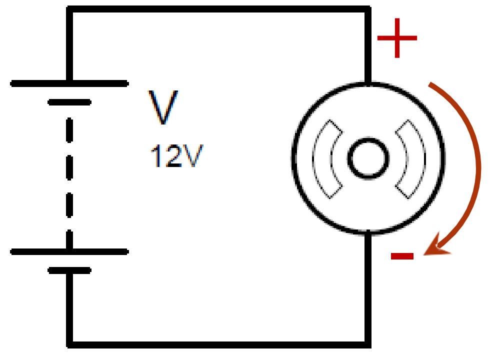


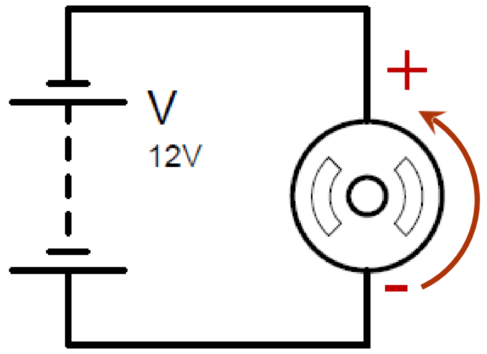


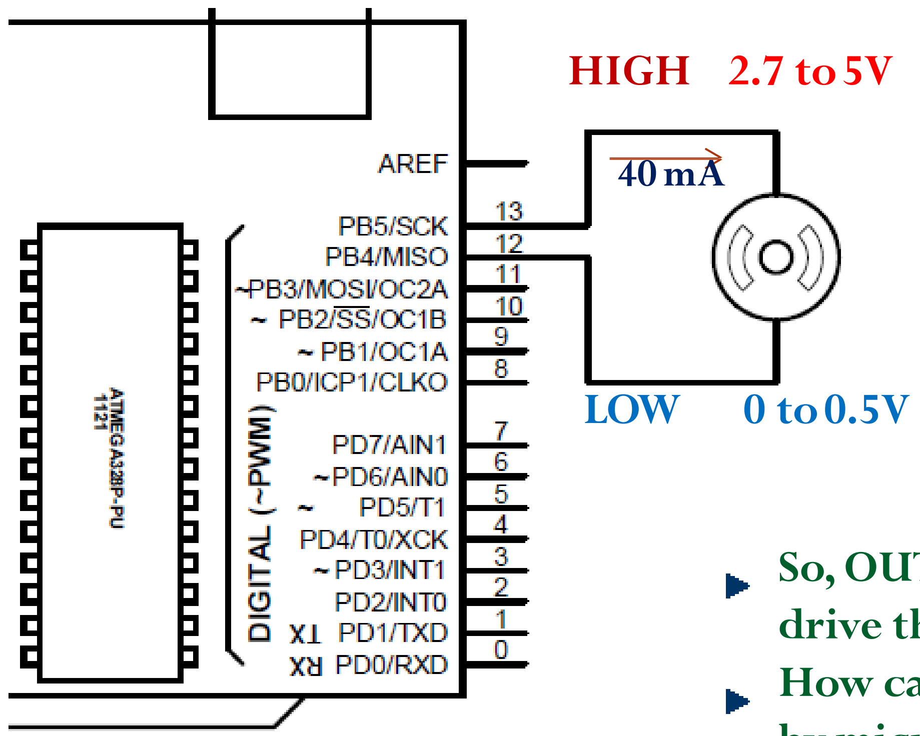




Direction Control of DC Motor



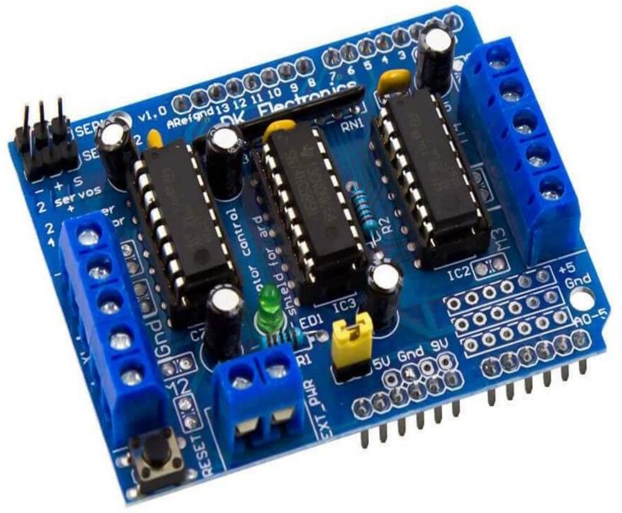




P=VI = 5 X 40 mA = 200 mW= 0.2W

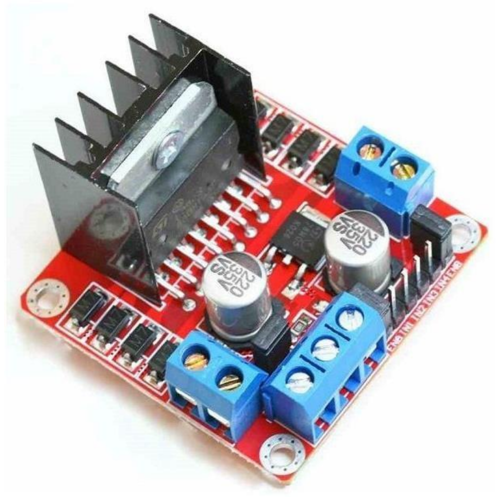
- Not enough power to drive a DC motor.
- The DC motor used in our project needs voltage of 6 to 12 V and 300-1000 mA current. Minimum of (6X300) =1800 mW or, 1.8W power
- So, OUTPUT pins of microcontroller can not drive the DC motor required in our project.
- How can we control the direction of DC motor by microcontroller?

Different Types of Motor Driver



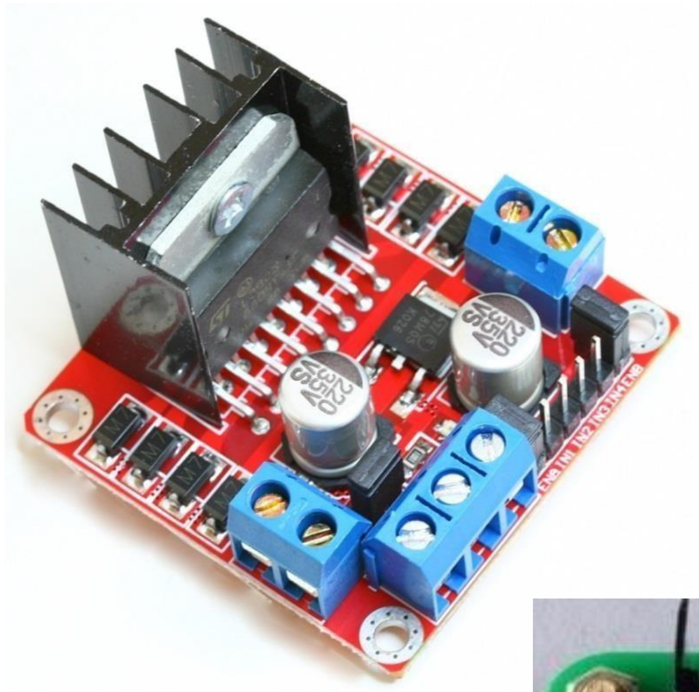
Model: L293D
Current rating: 0.6A
(1.2A peak current)
Voltage: 4.5V to 16V DC

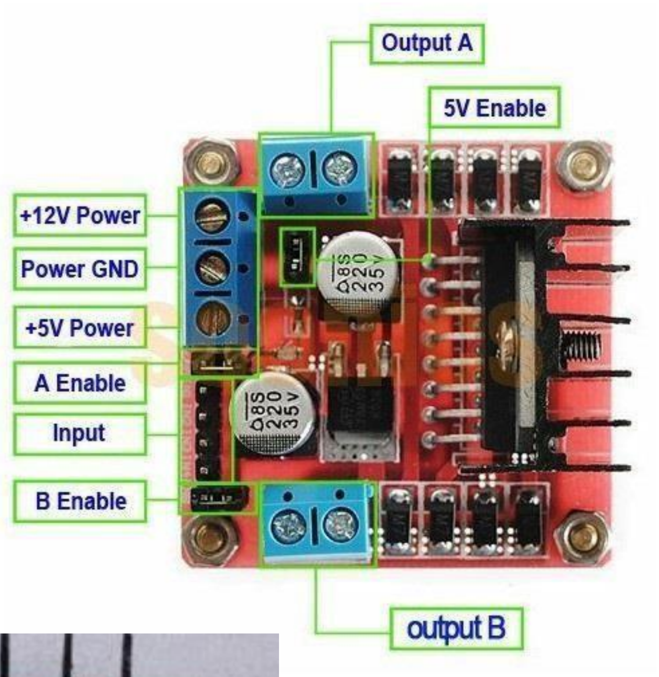
Model: BTS7960
Current rating: Max 43 Amp
Voltage: 24 V



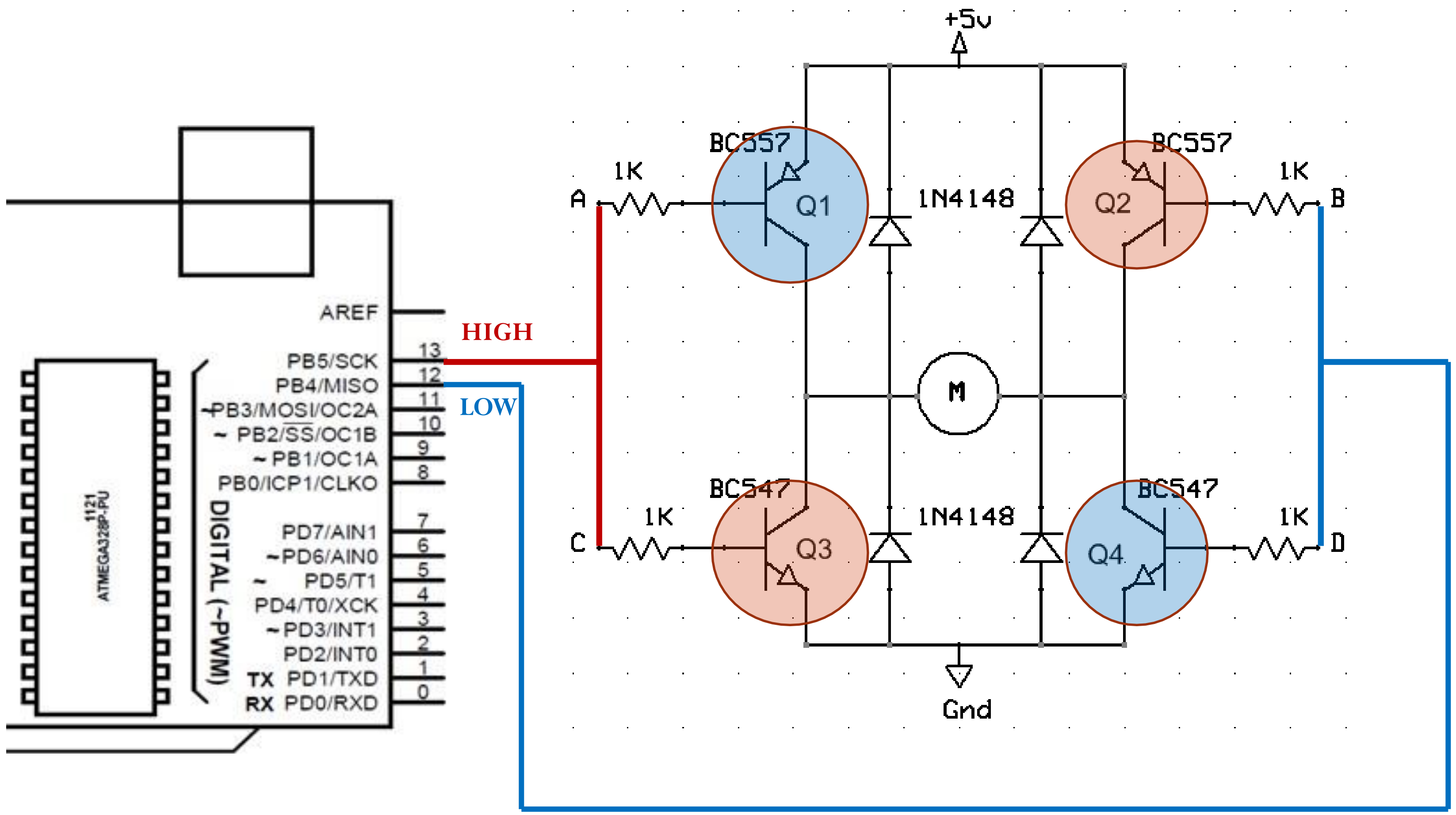
Model: L298N
Current Rating: 2 A
Voltage: 5 to 12 V

L298N Motor Driver Module

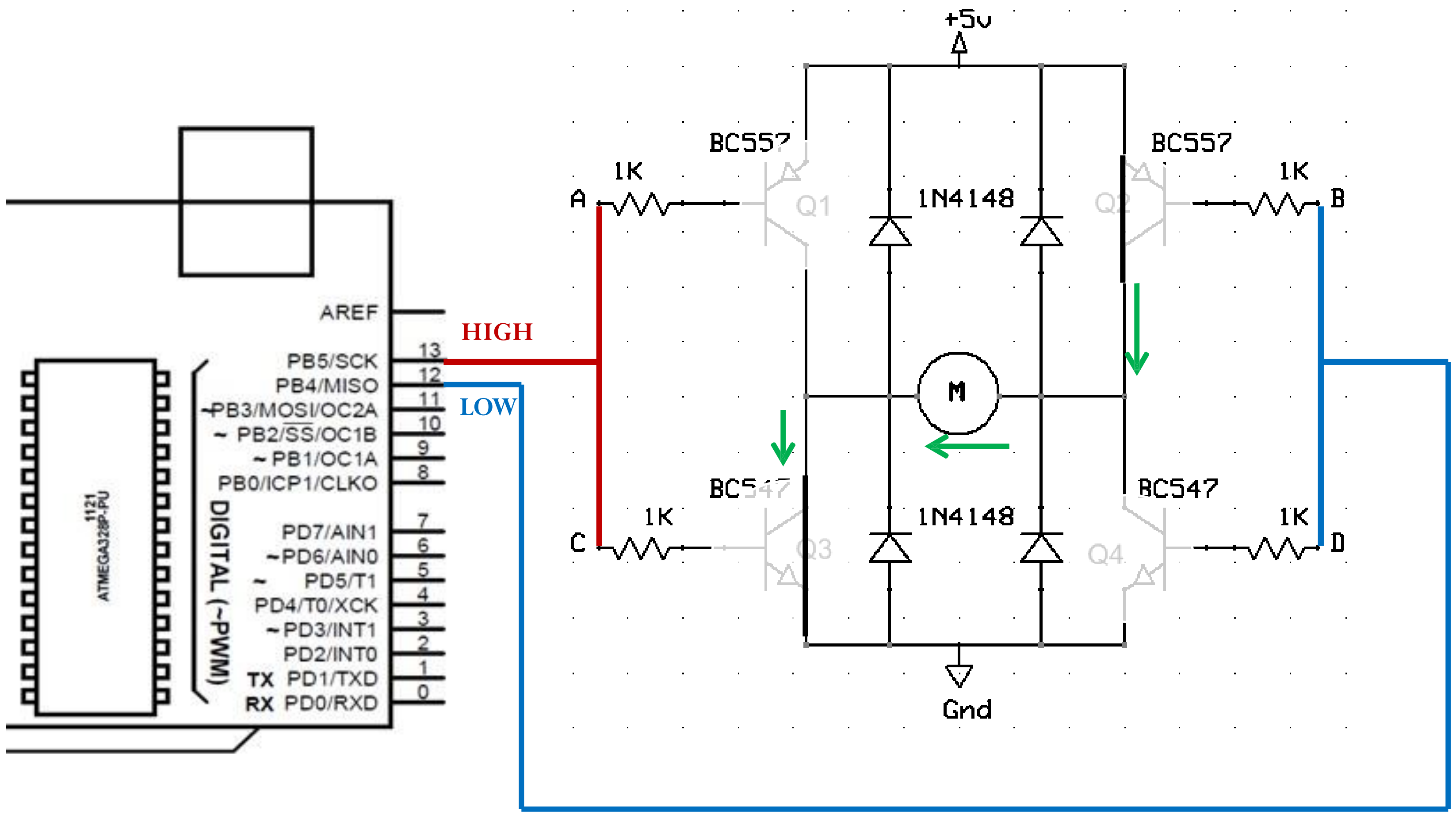




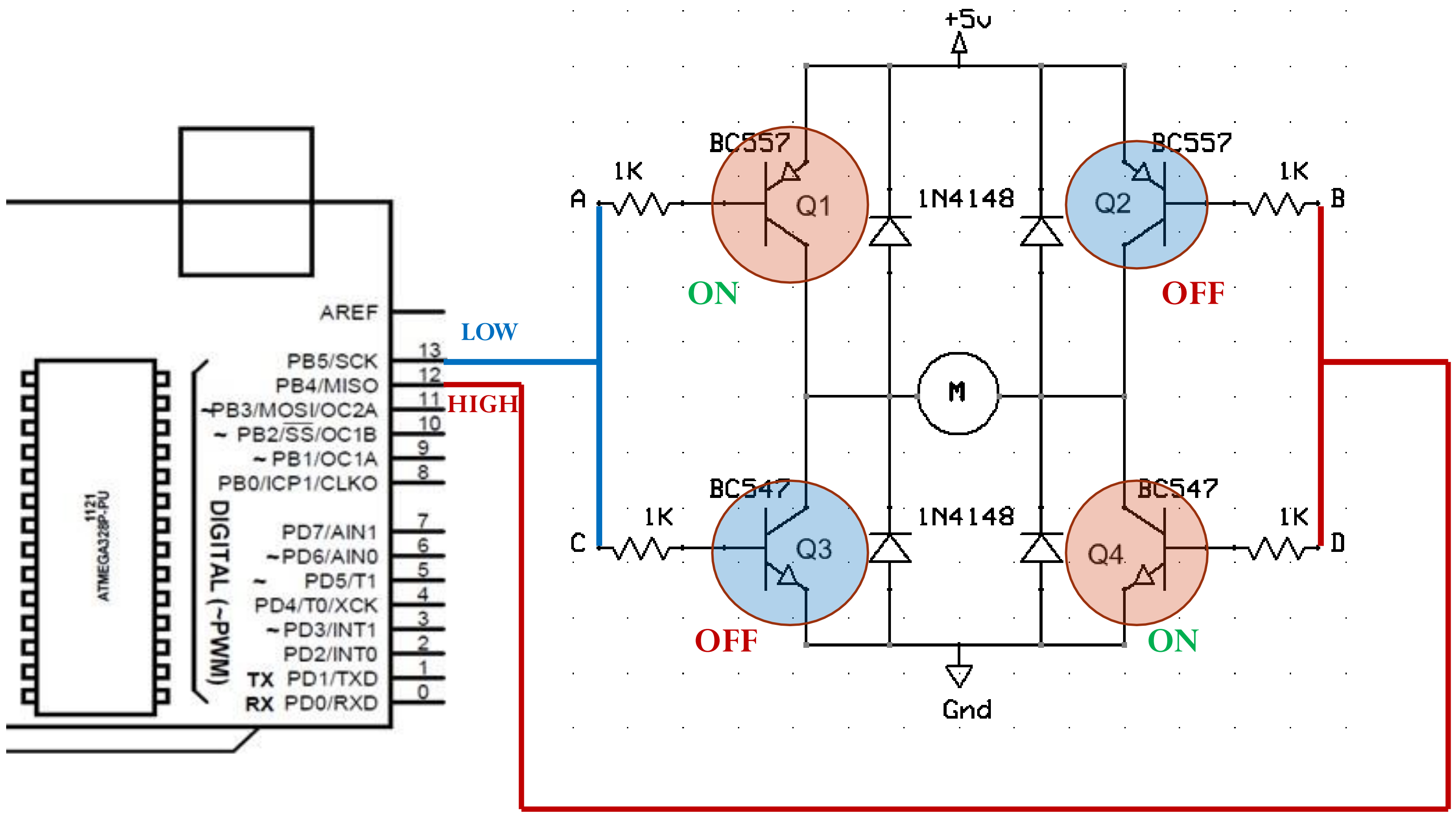
H-Bridge Motor driver



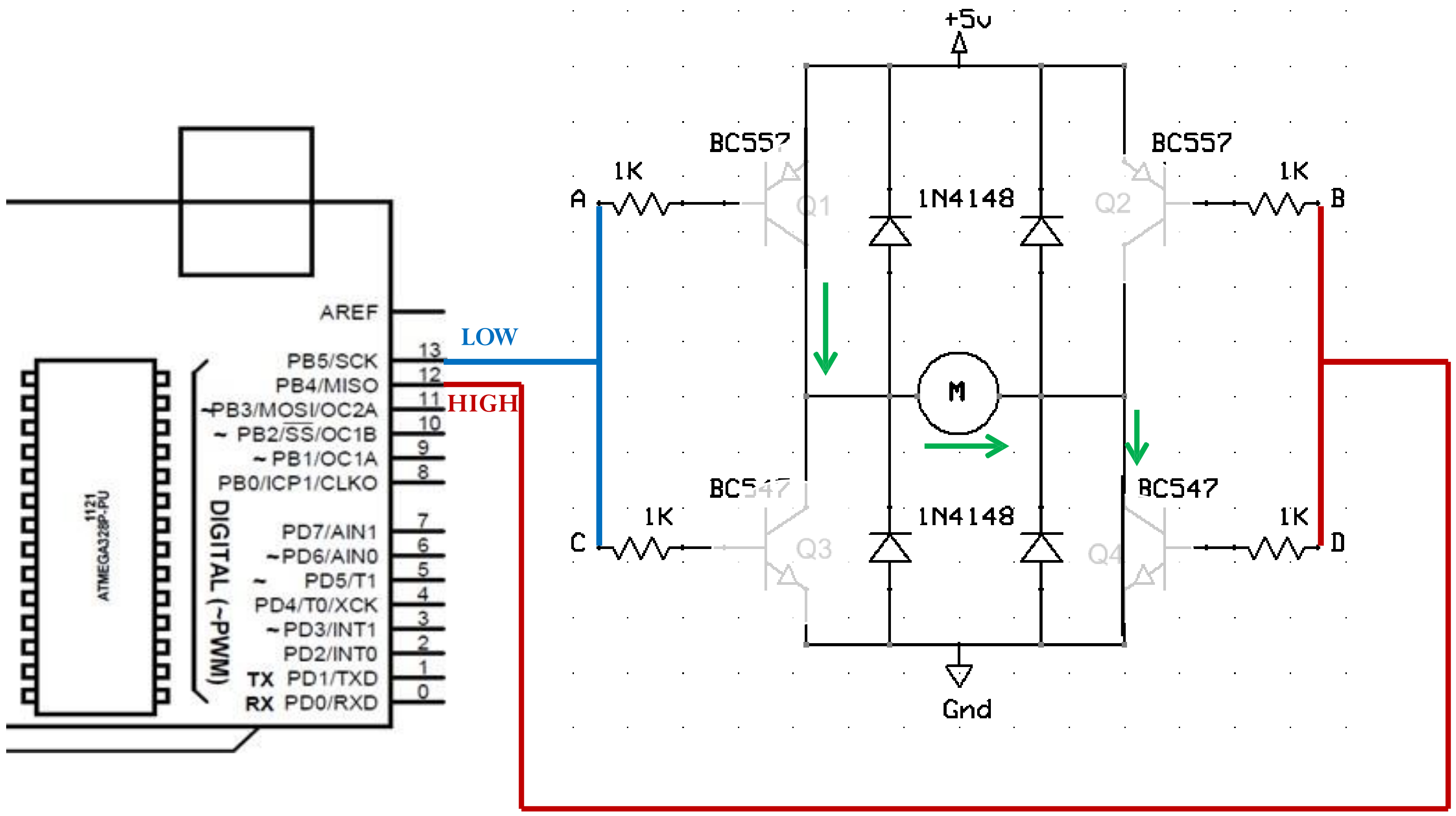
H-Bridge Motor driver



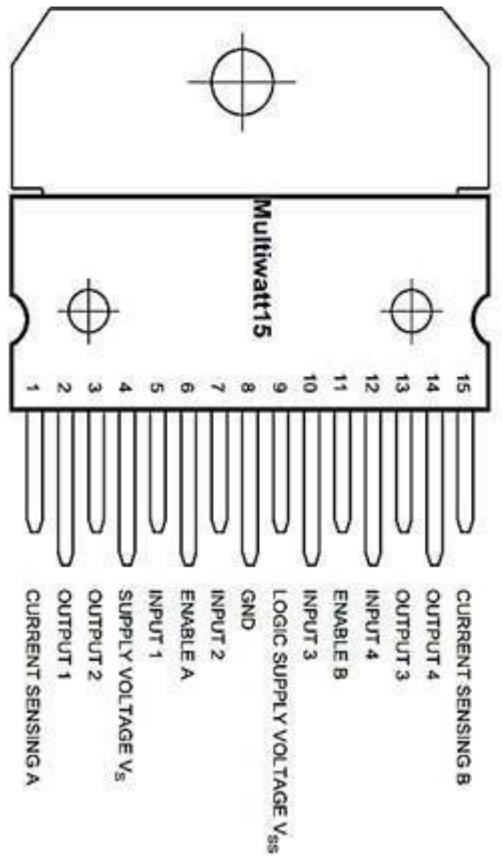
H-Bridge Motor driver

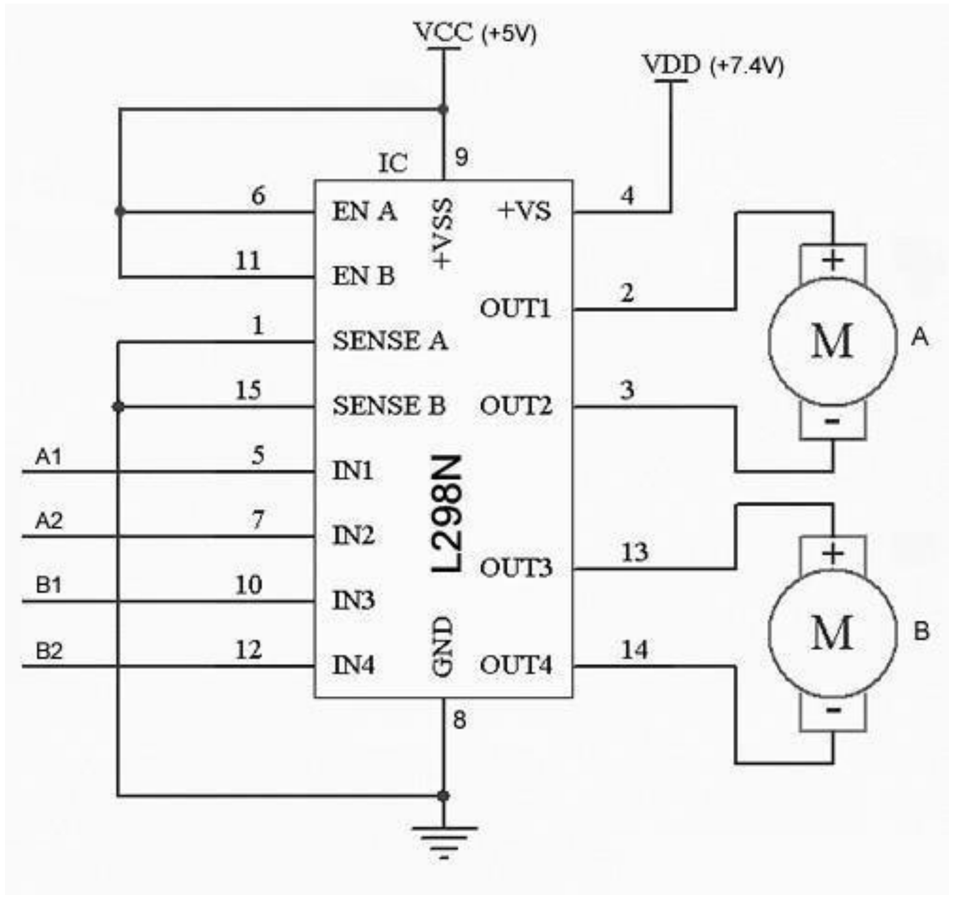


H-Bridge Motor driver



Motor Driver IC (L298N)





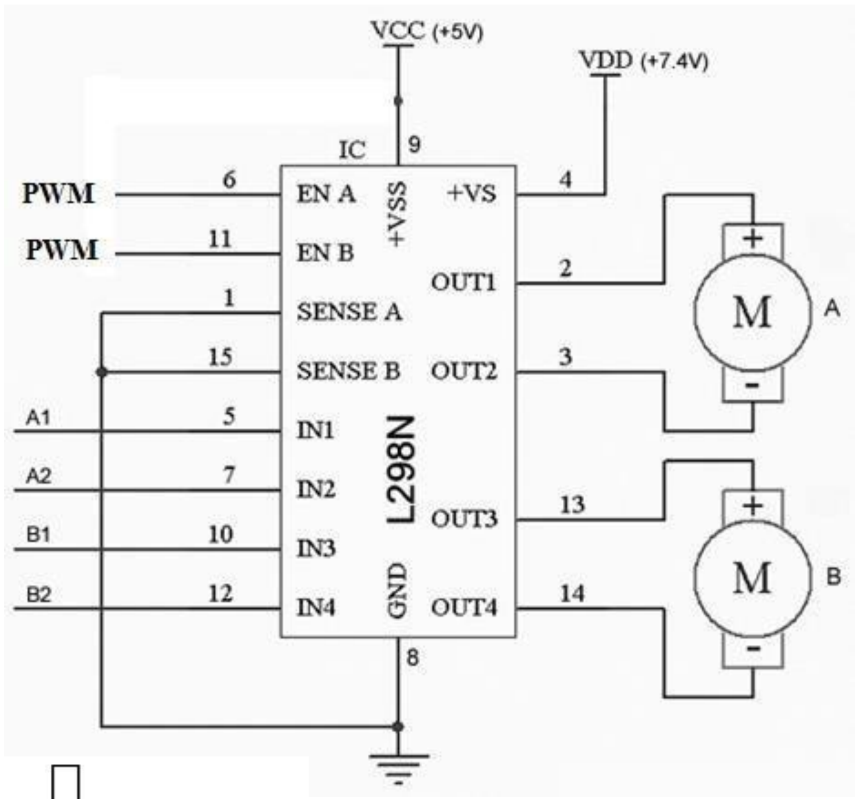
Speed Control of DC Motor

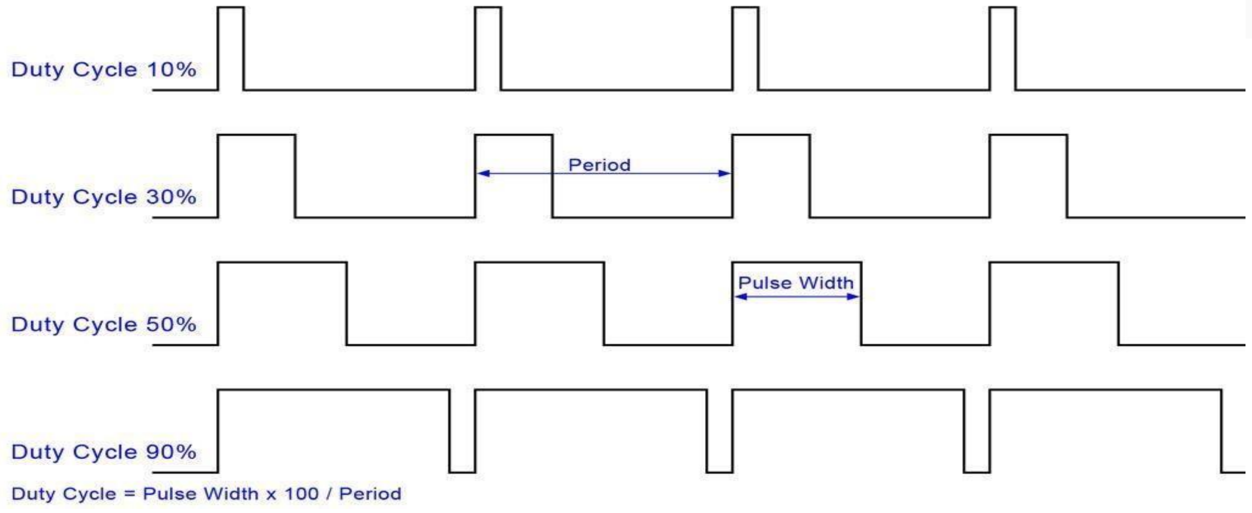
InARDUINO,

duty cycle of PWM is set by the function

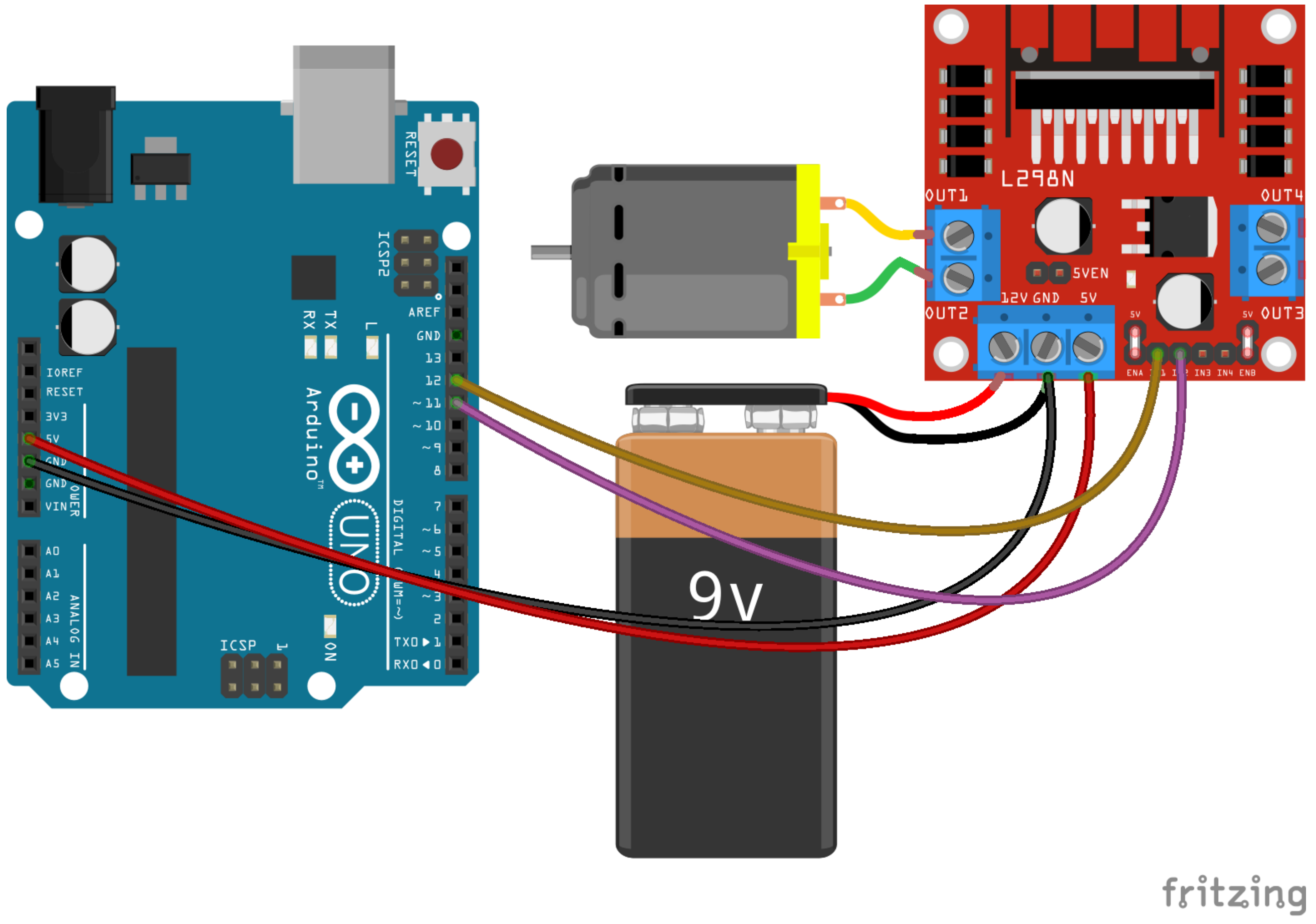
analogWrite(pin_no, duty_cycle)

Where, duty_cycle = 0 to 255 (0 to 100%)



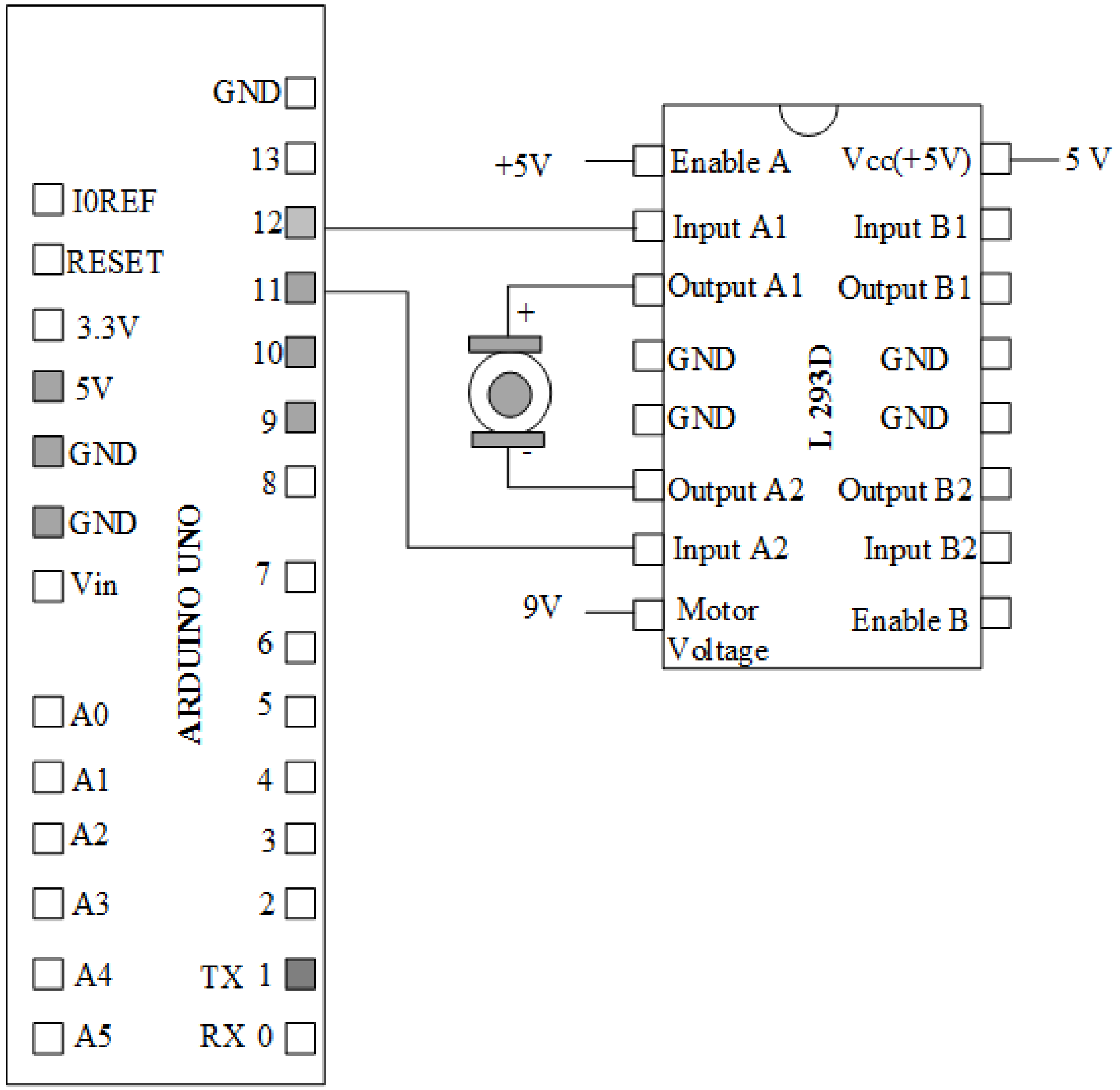


Circuit Diagram



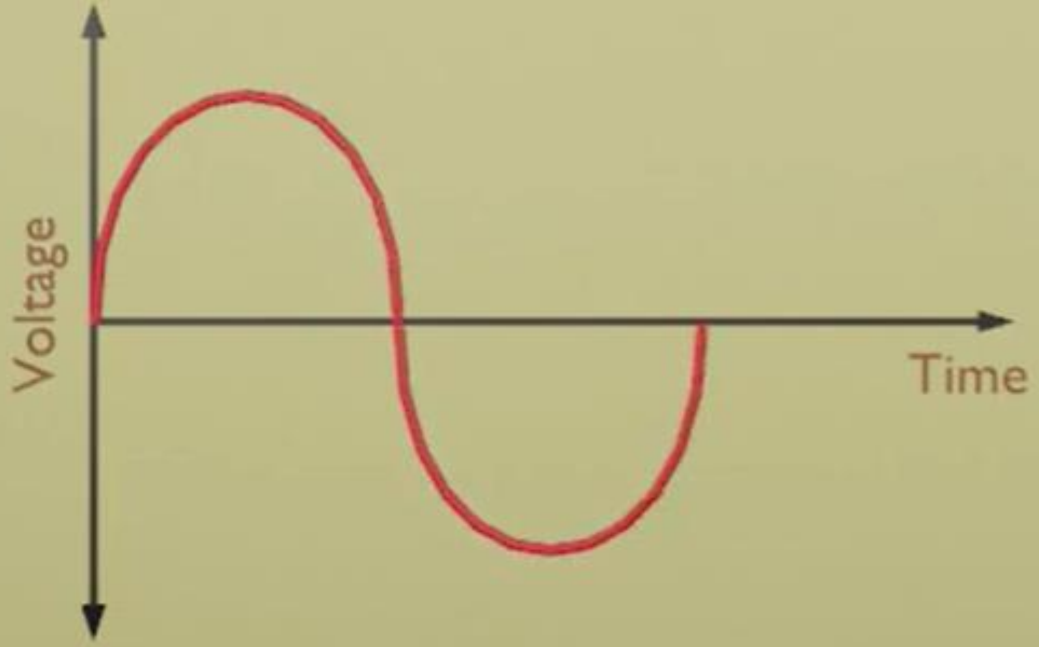
Direction control of DC motor.

```
const int MT1=12;
const int MT2=11;
void setup() {
pinMode (MT1, OUTPUT);
pinMode(MT2, OUTPUT);
}
void loop() {
digitalWrite(MT1,HIGH);
digitalWrite(MT2,LOW);
delay(5000);
digitalWrite(MT1,LOW);
digitalWrite(MT2,LOW);
delay(1000);
digitalWrite(MT1,LOW);
digitalWrite(MT2,HIGH);
delay(5000);
digitalWrite(MT1,LOW);
digitalWrite(MT2,LOW); delay(1000);
}
```

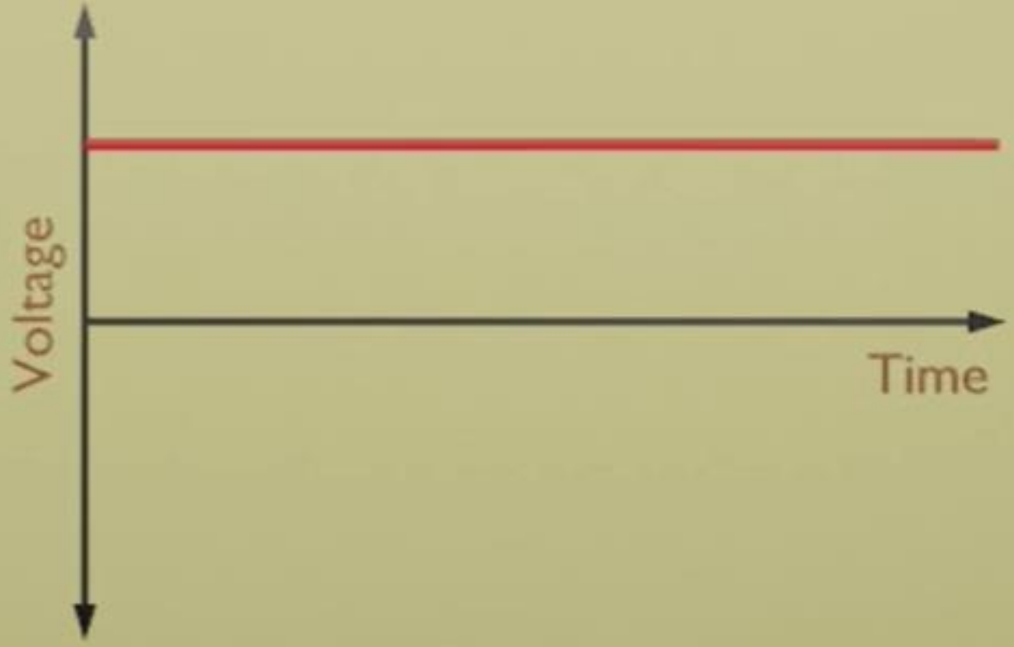


Alternating Current AC

Direct Current DC



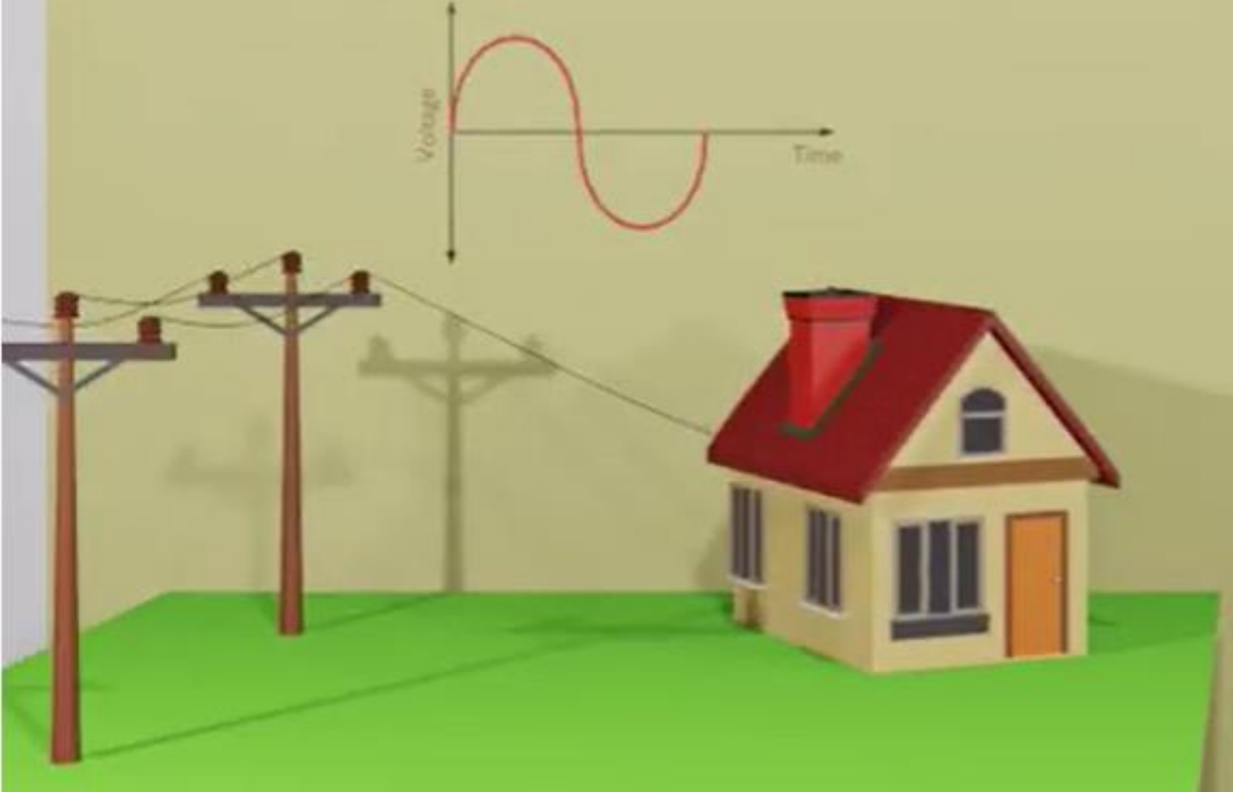
Periodically reverses direction



flows in only one direction

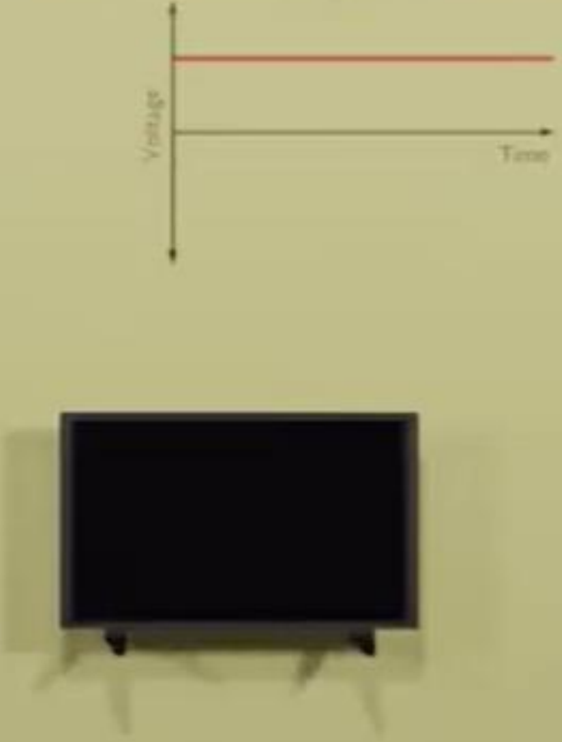
Alternating Current
AC

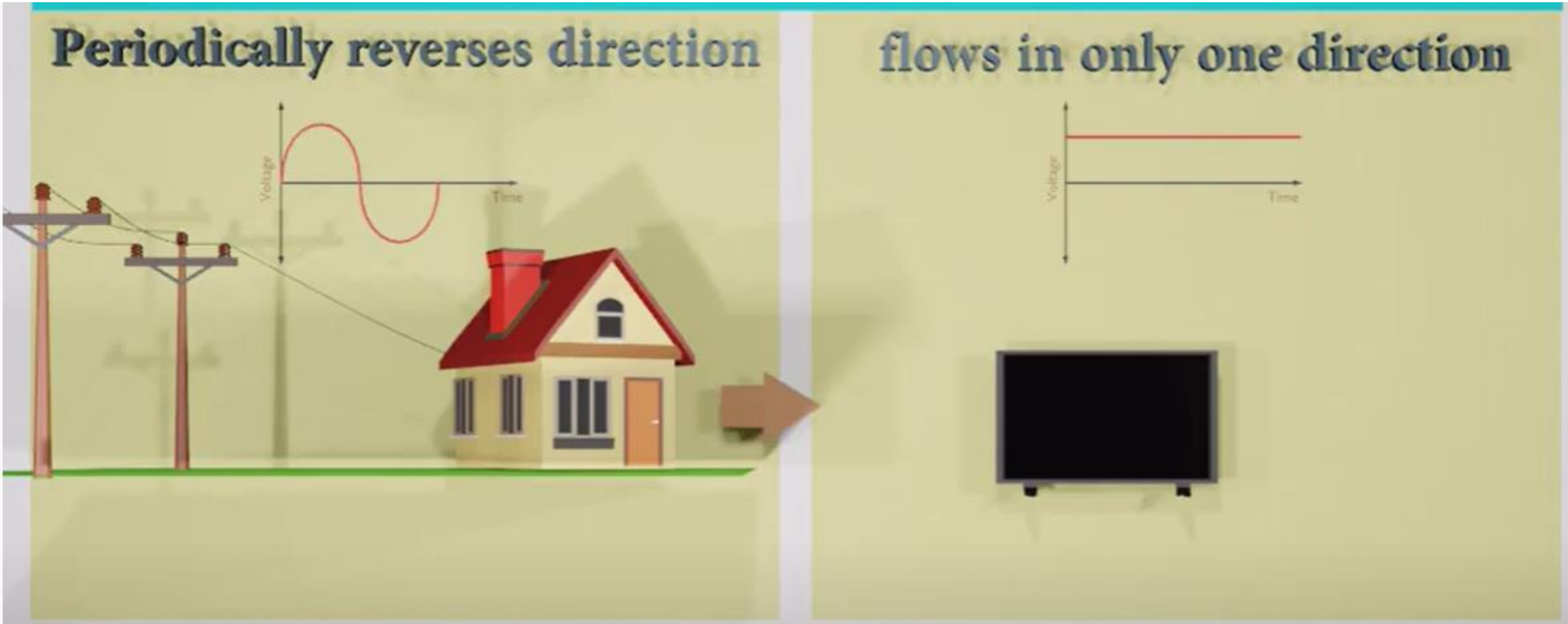
Periodically reverses direction



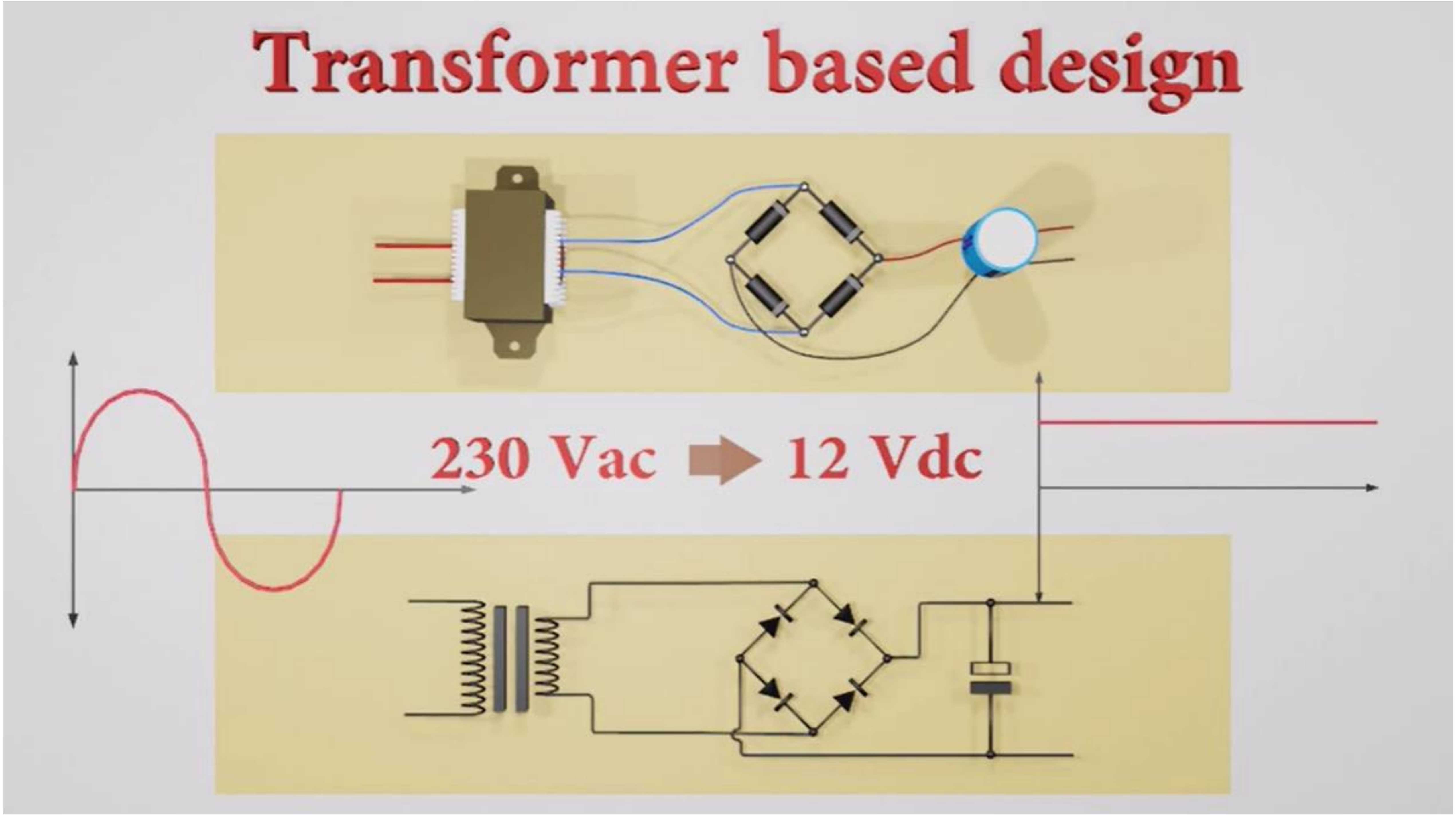
Direct Current
DC

flows in only one direction

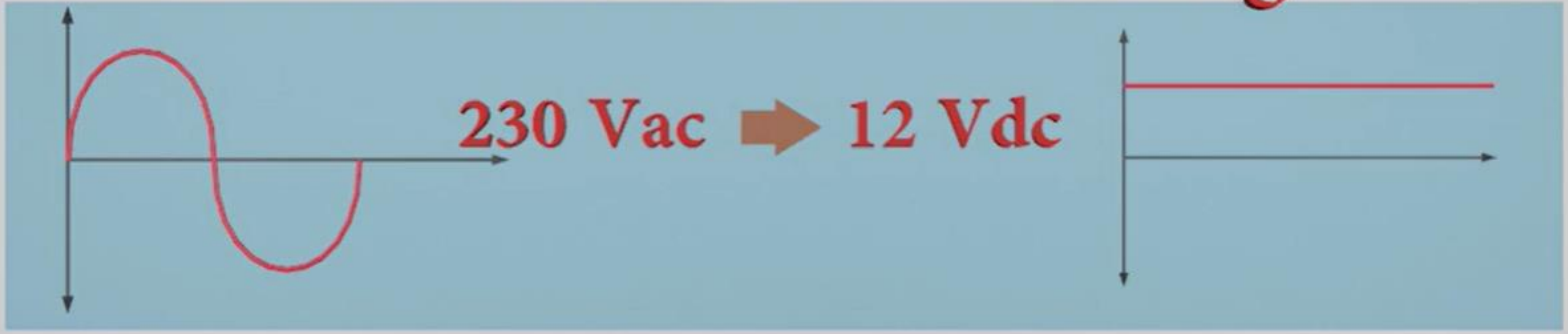




Rectification : Converting Alternating curent to Direct current


Transformer based design
230 Vac
12 Vdc

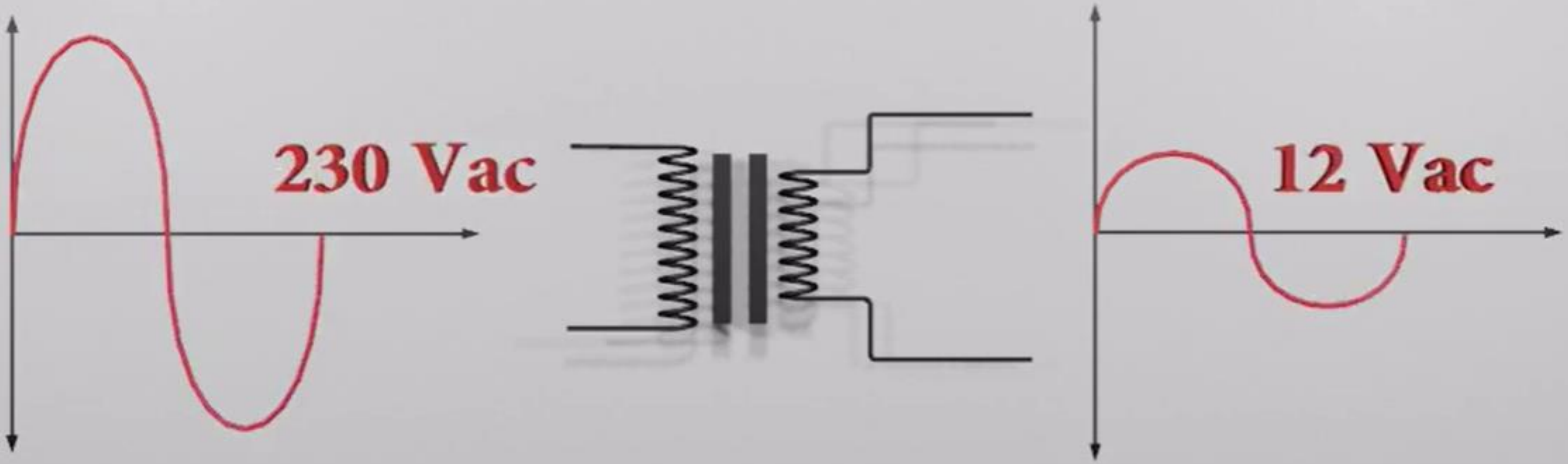
Transformer based design



1. Stepping down the Voltage Levels
2. AC to DC Power Converter Circuit
3. Obtaining Pure DC from Pulsating DC
4. Regulating DC Voltage

1. Stepping down the Voltage Levels

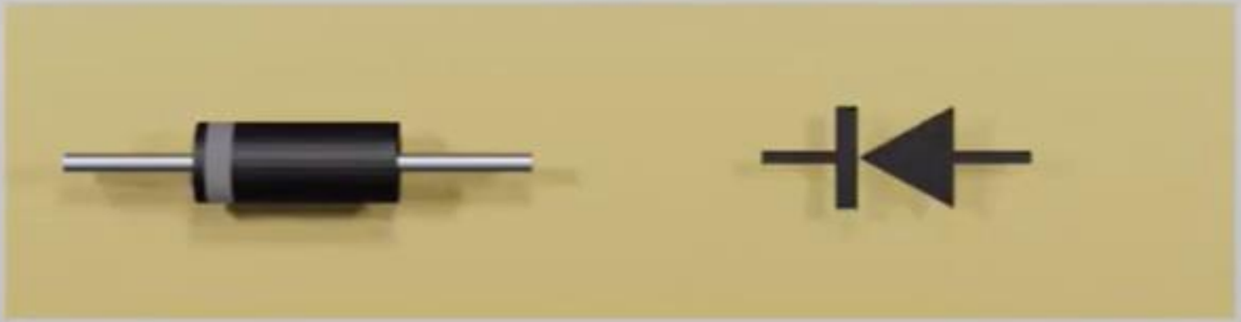
Using a step-down transformer the available 230V AC power supply is converted into 12V AC

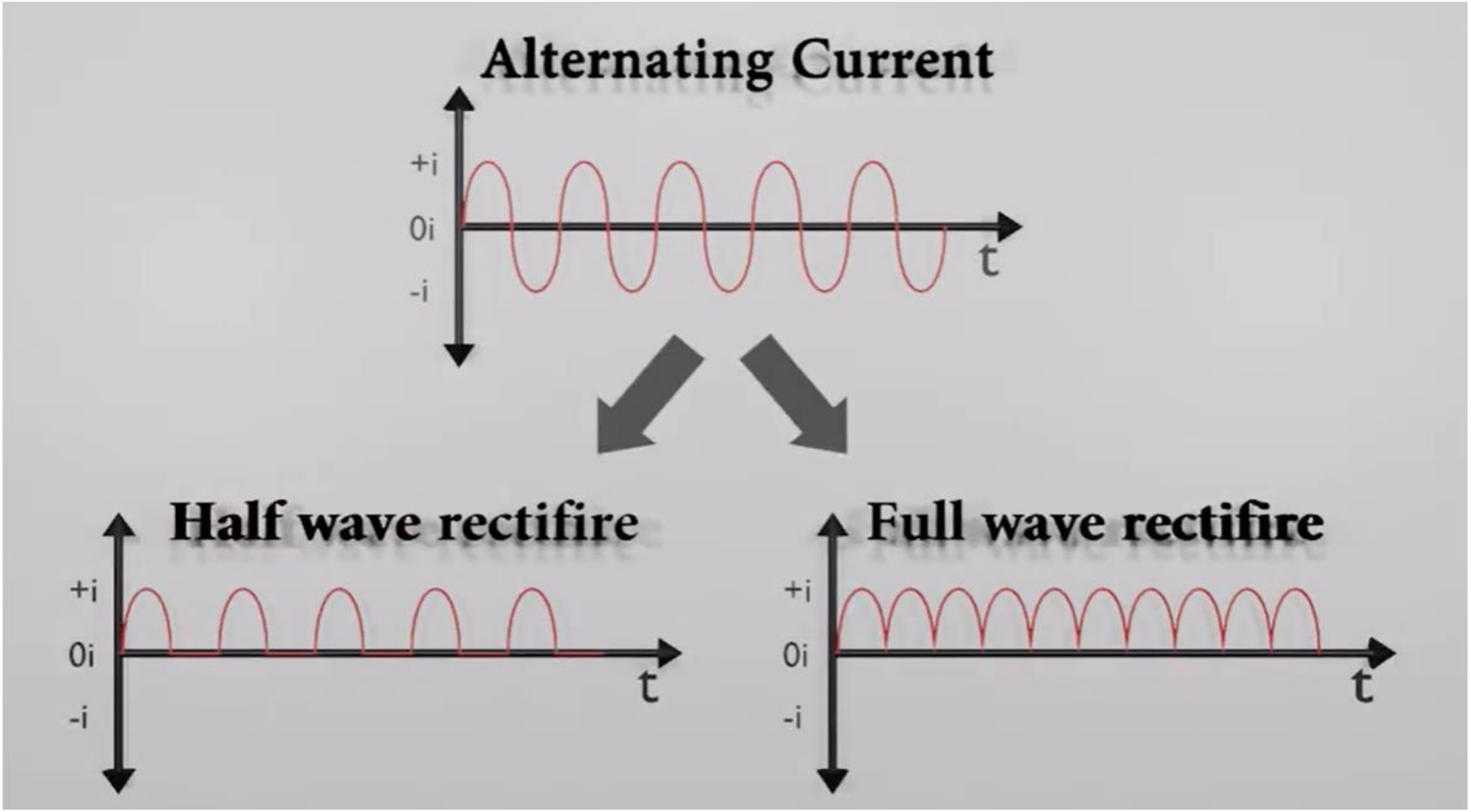


2. AC to DC Power Converter Circuit

converting alternating current into direct current is given the name rectification.

A p-n junction diode conducts current only in one direction. The same principle is use of in a rectifier to convert AC to DC

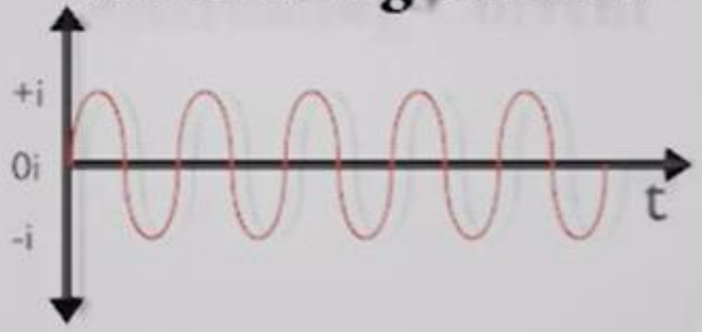


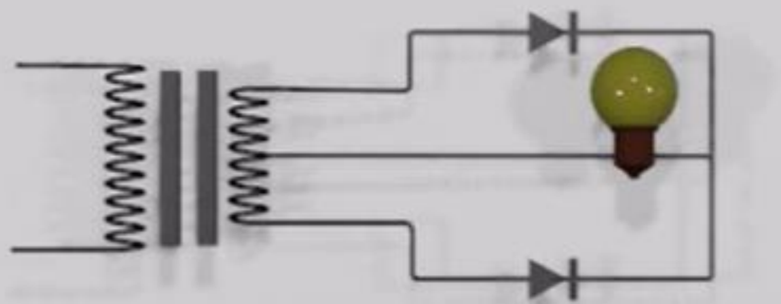

Alternating Current
+i
0i
-i
t
Half wave rectifire
+i
0i
-i
t
Full wave rectifire
+i
0i
-i
t

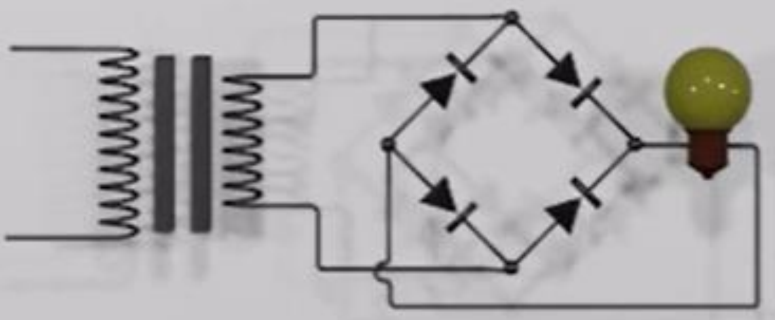
Full wave rectifire

Input current

Alternating Current

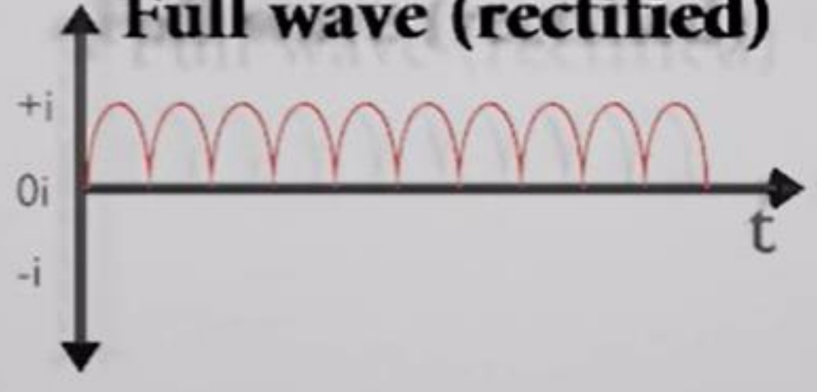




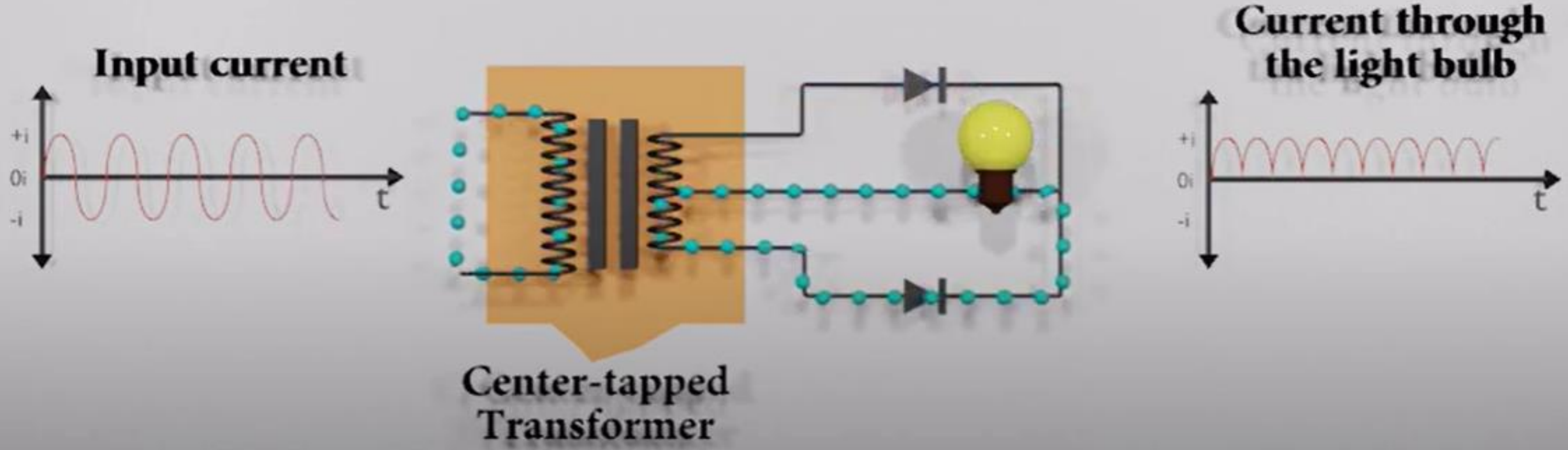


Current through the light bulb

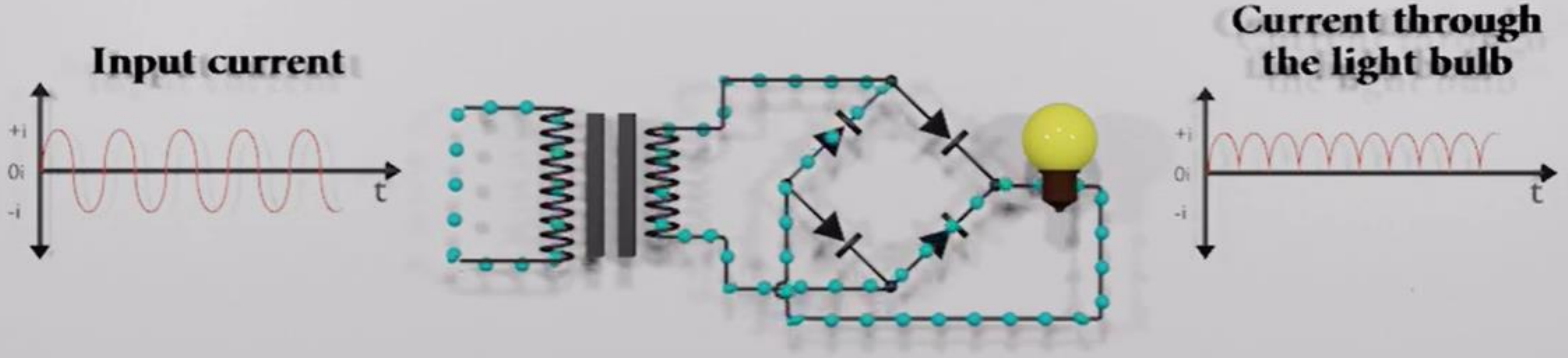
Full wave (rectified)



Full wave rectifire

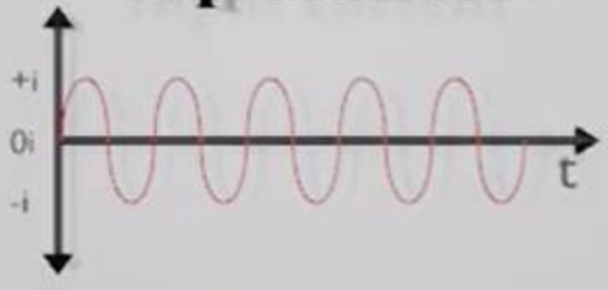


Full wave rectifire

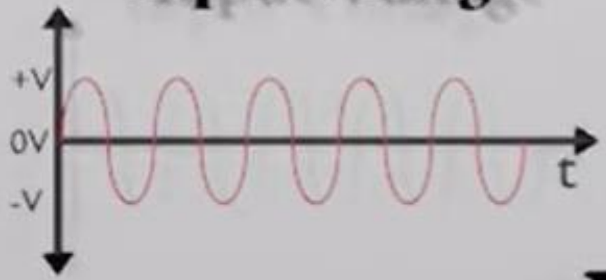


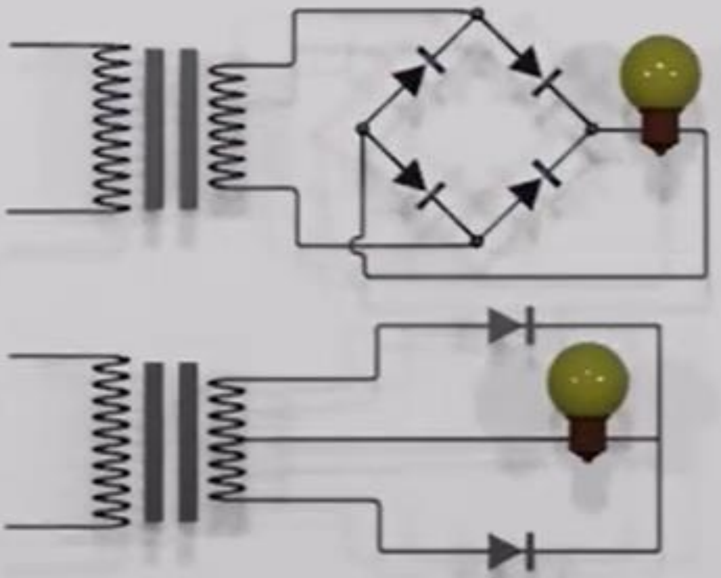
Full wave rectifire

Input current

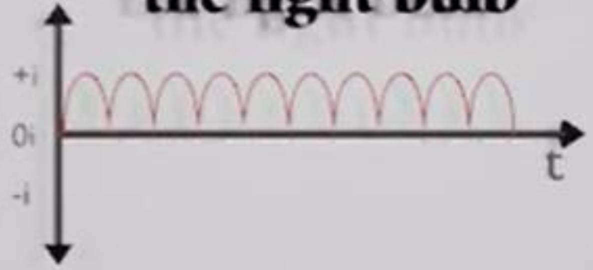


Input Voltage

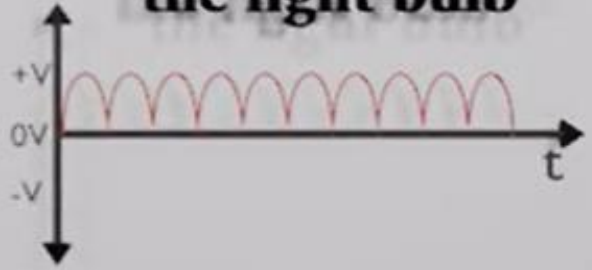




Current through the light bulb



Voltage through the light bulb

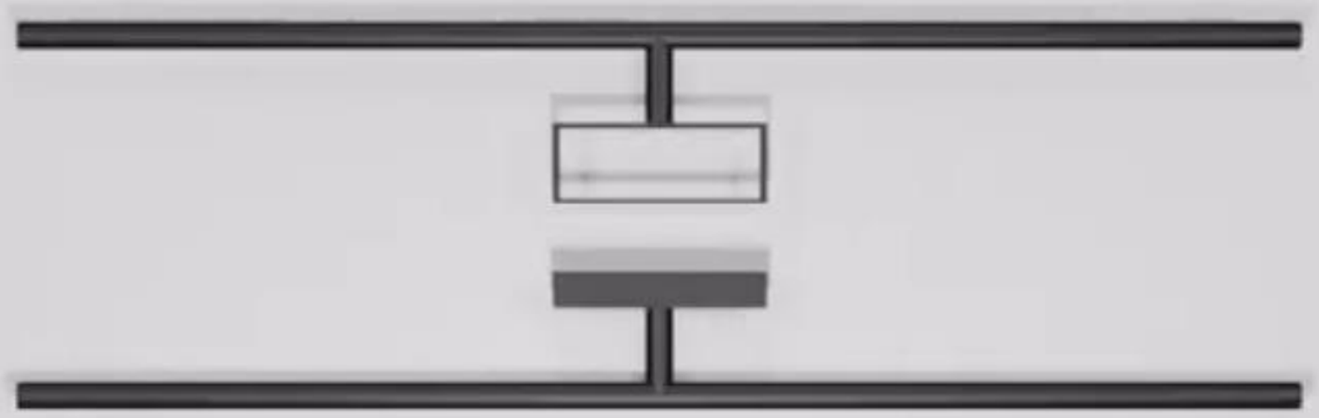


Full wave is rectified

But voltage is not stable

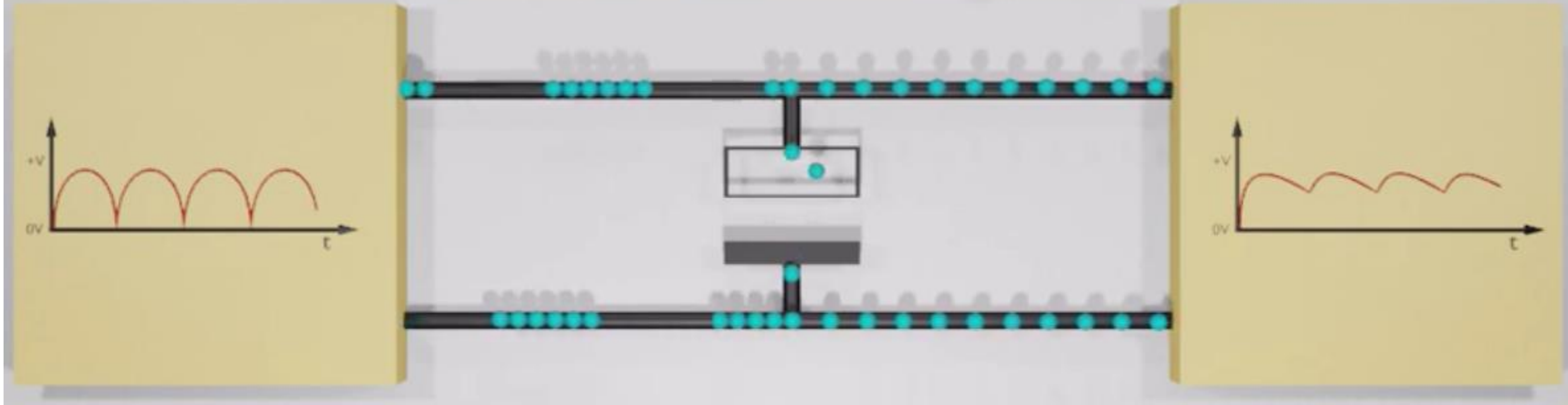
3. Obtaining Pure DC from Pulsating DC (Smoothing)

3. Obtaining Pure DC from Pulsating DC (Smoothing)



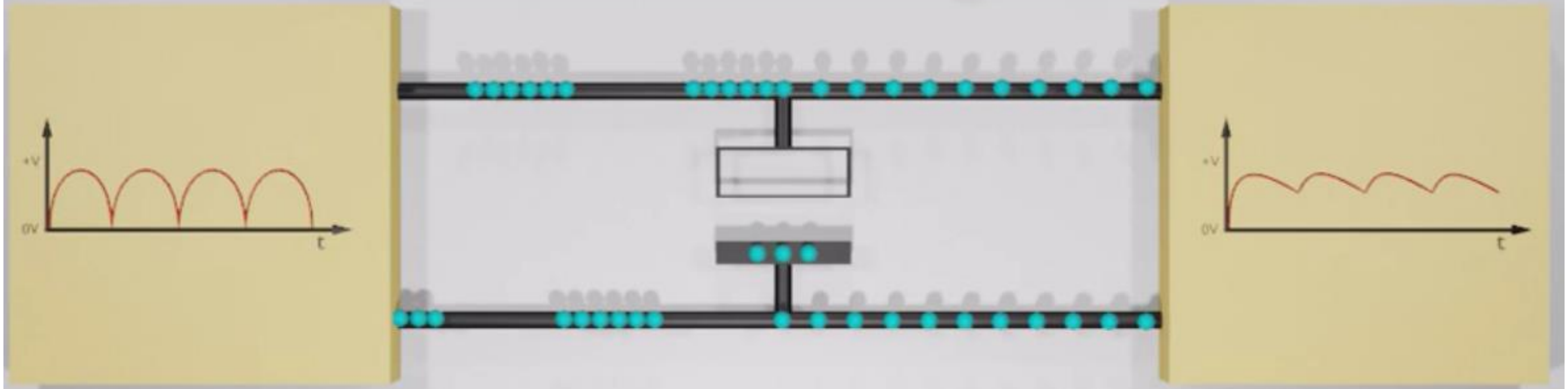
Capacitor charges up when the voltage from the rectifier rises above the capacitor Voltage and then as the rectifier voltage falls, the capacitor provides the required current from its stored charge.

3. Obtaining Pure DC from Pulsating DC (Smoothing)



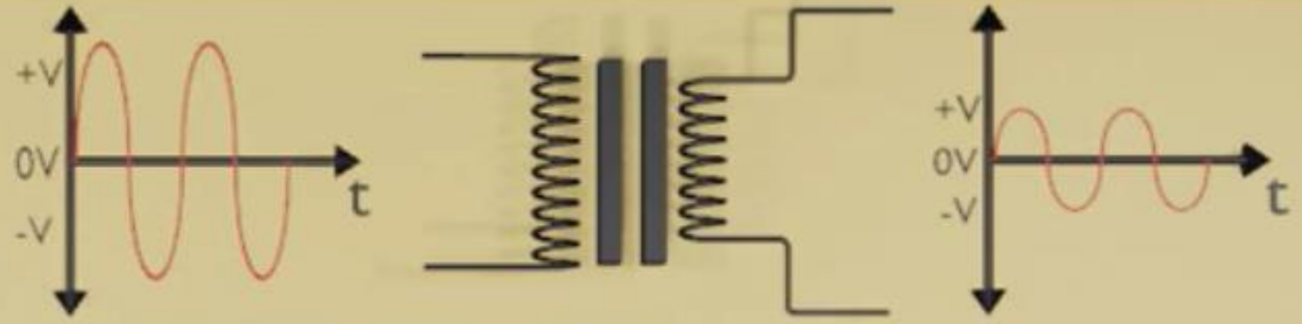
Capacitor charges up when the voltage from the rectifier rises above the capacitor Voltage and then as the rectifier voltage falls, the capacitor provides the required current from its stored charge.

3. Obtaining Pure DC from Pulsating DC (Smoothing)

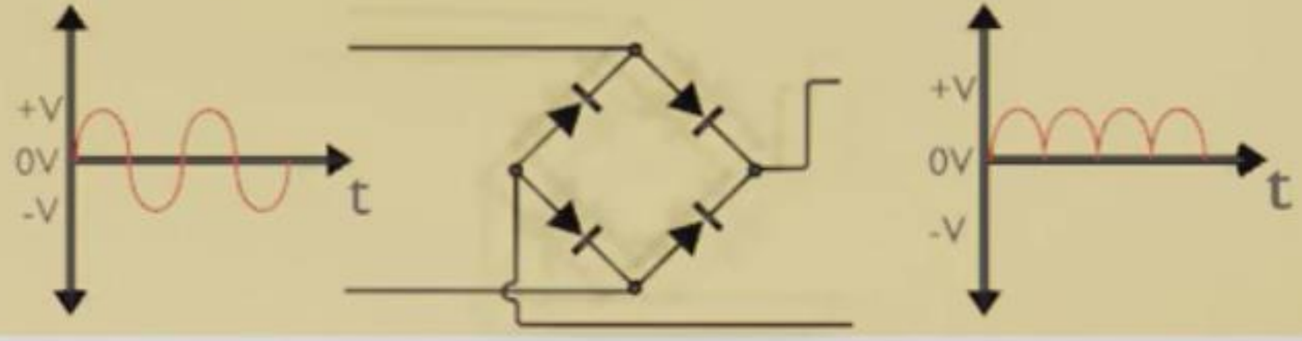


Ripple voltage is the residual periodic variation of the DC voltage

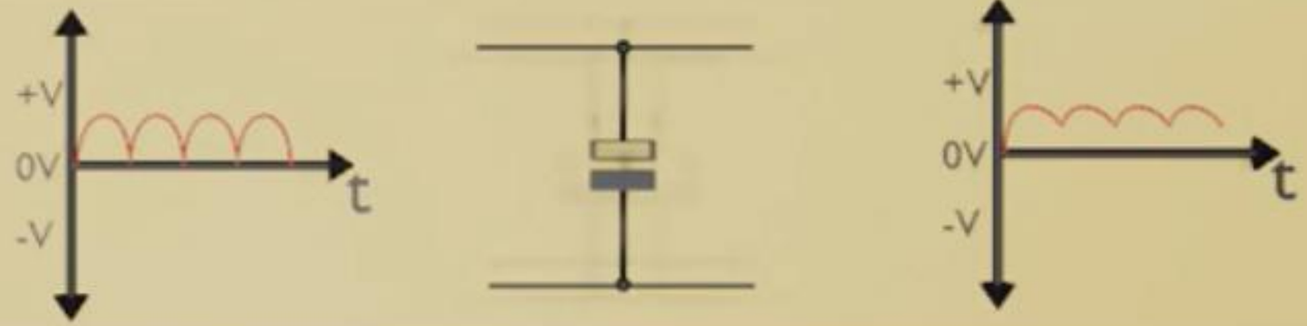
1 Step Down



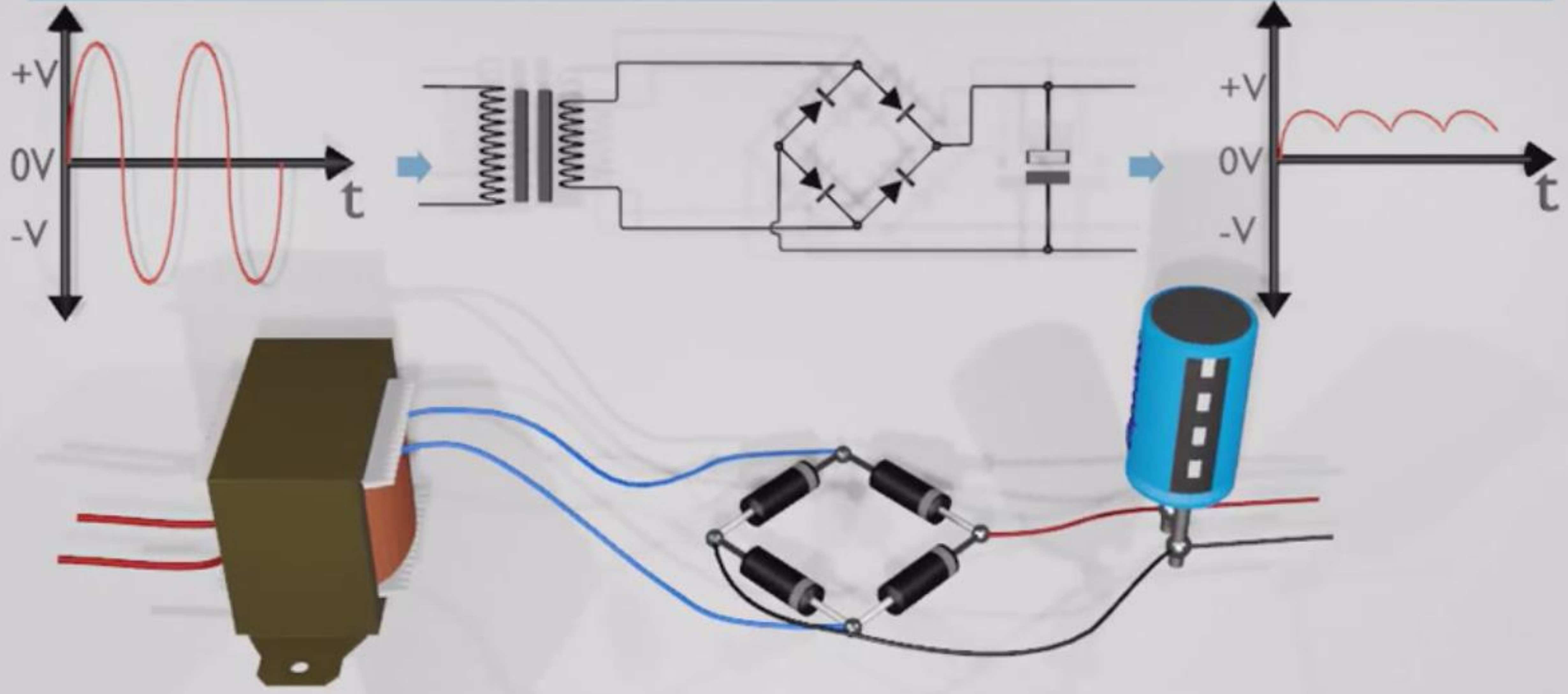
2 Rectification



3 Filtering



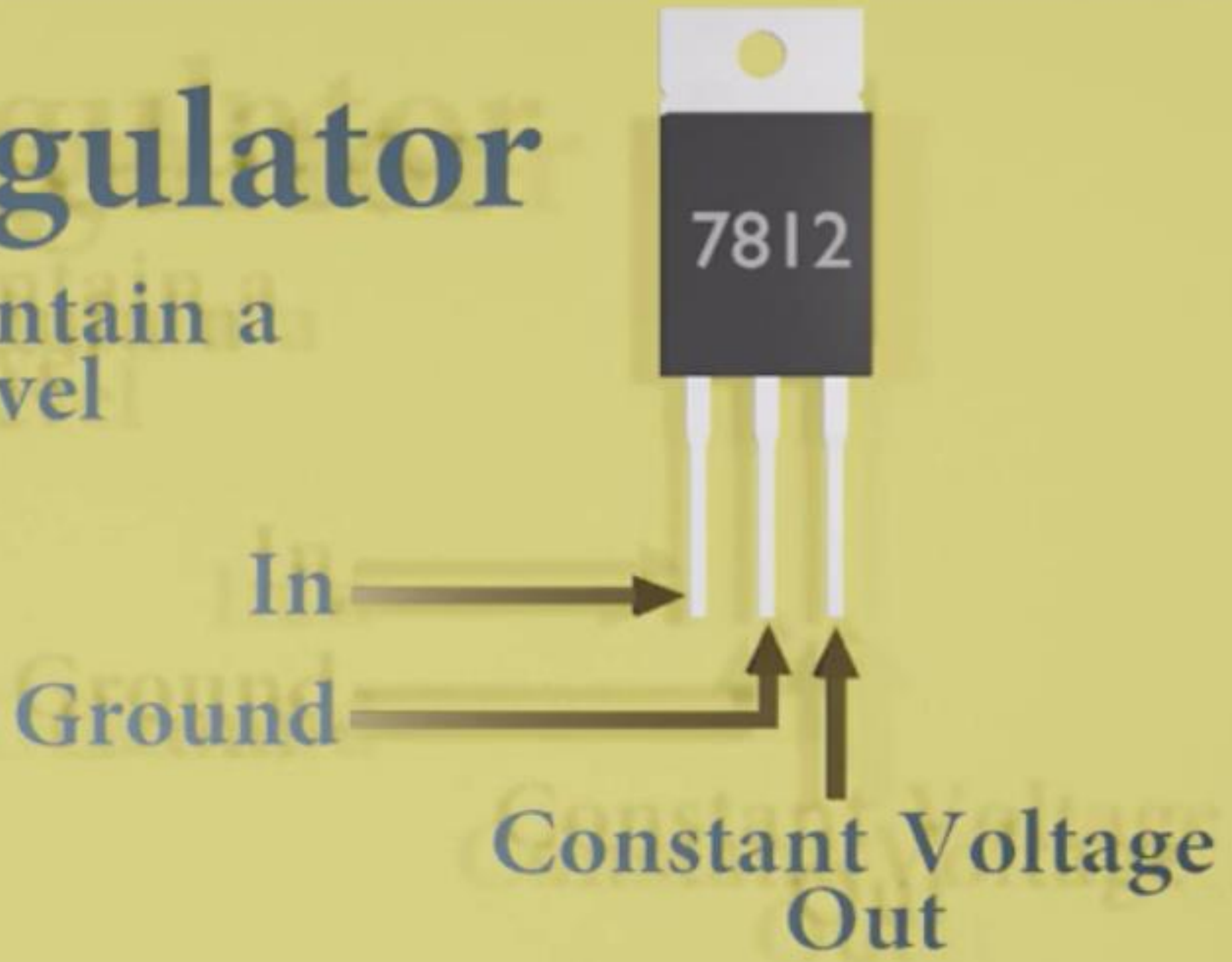
AC-DC Converter



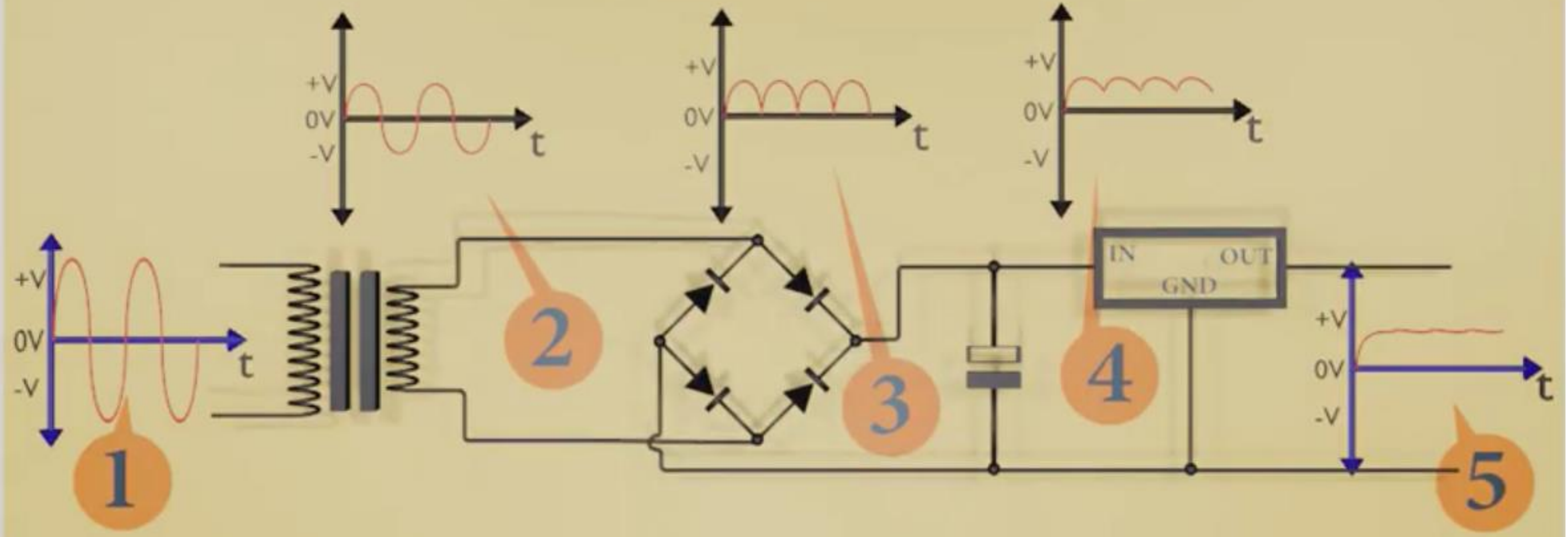
Furthermore......

Voltage regulator

Automatically maintain a constant voltage level



All together.......

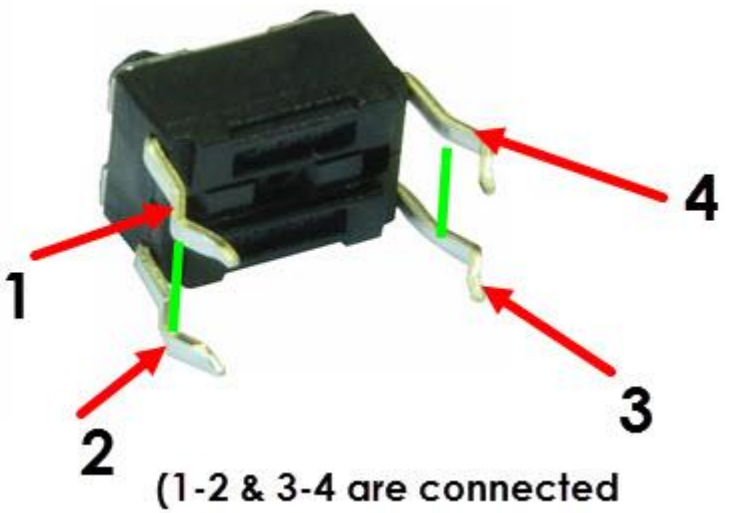


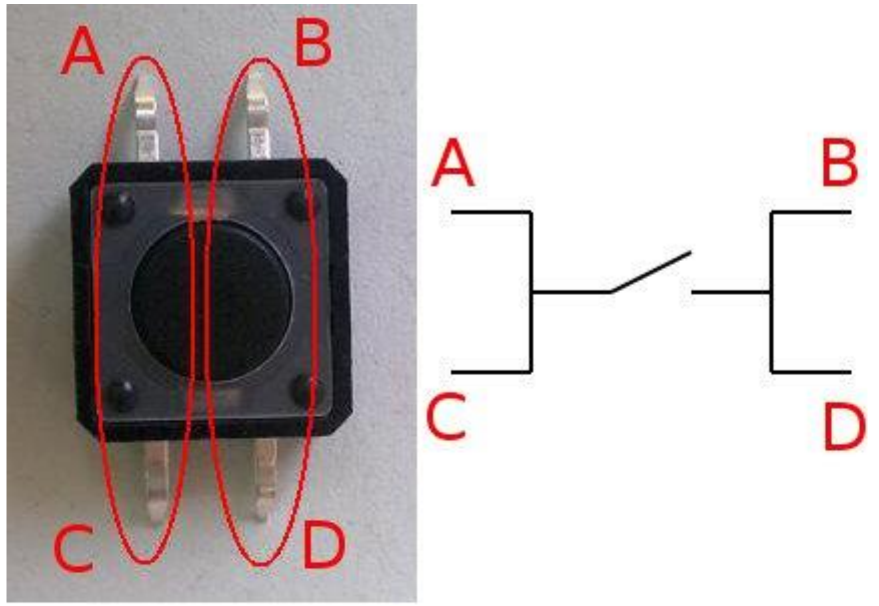
Push Button

Toggle Switch

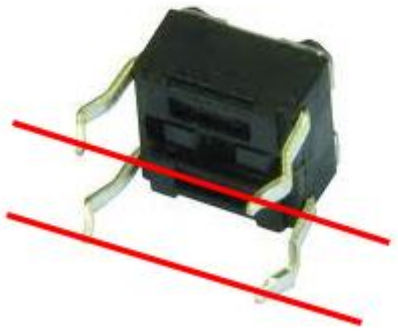
Push Button

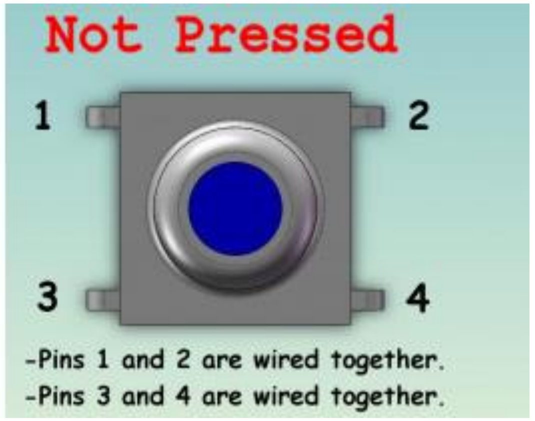
Push Button Internal Circuit

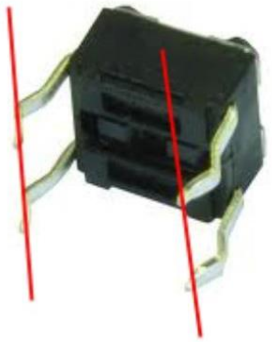


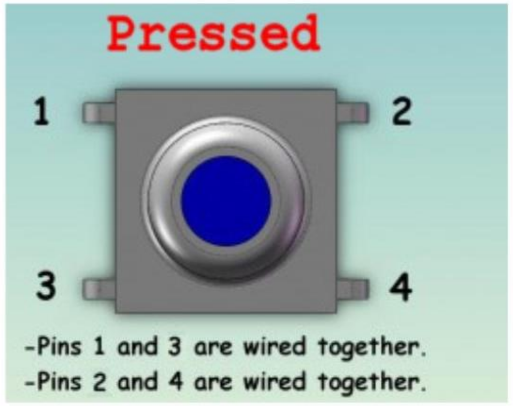


Push Button Internal Circuit

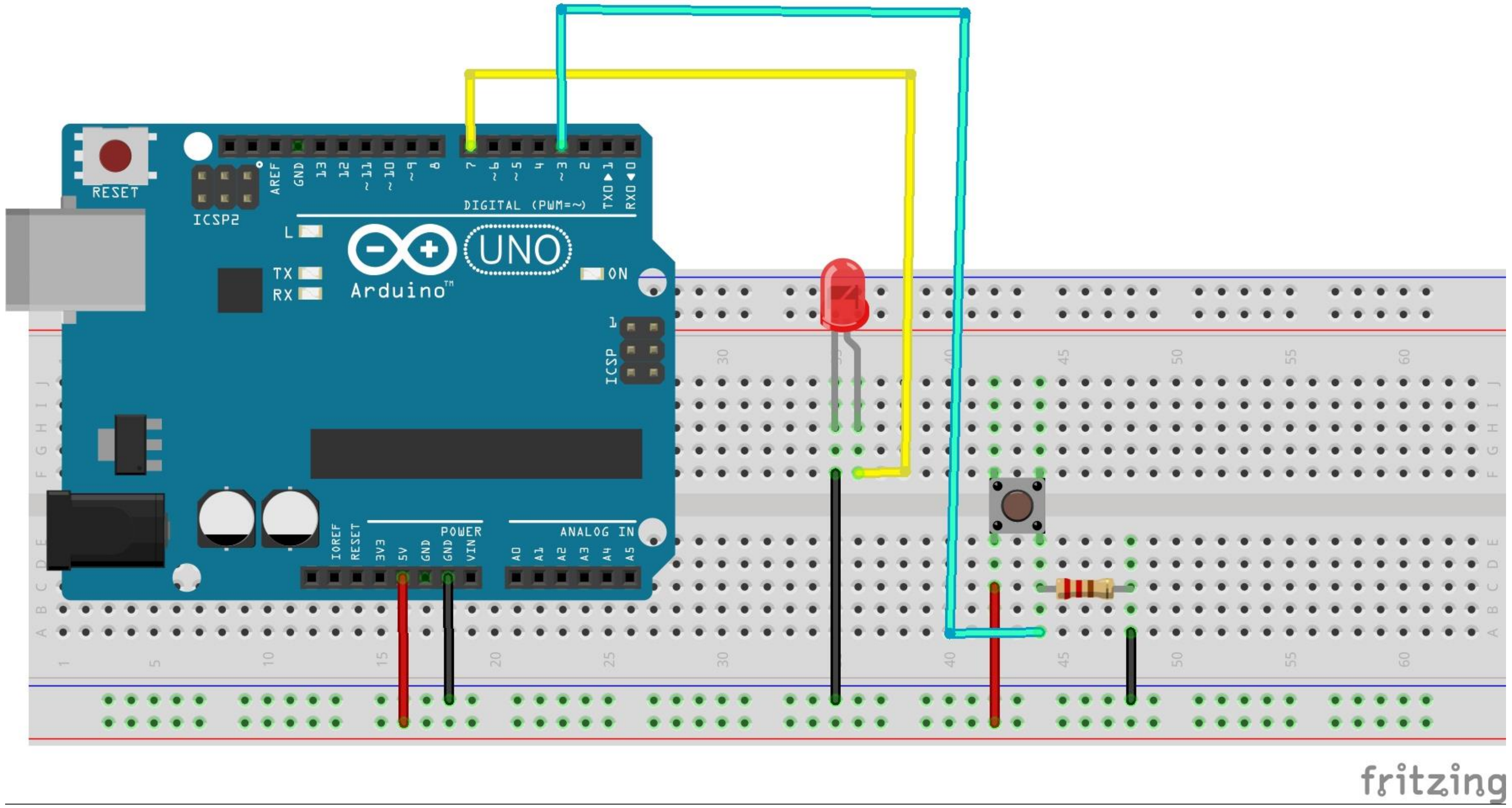



Not Pressed
1
2
3
4
-Pins 1 and 2 are wired together.
-Pins 3 and 4 are wired together.

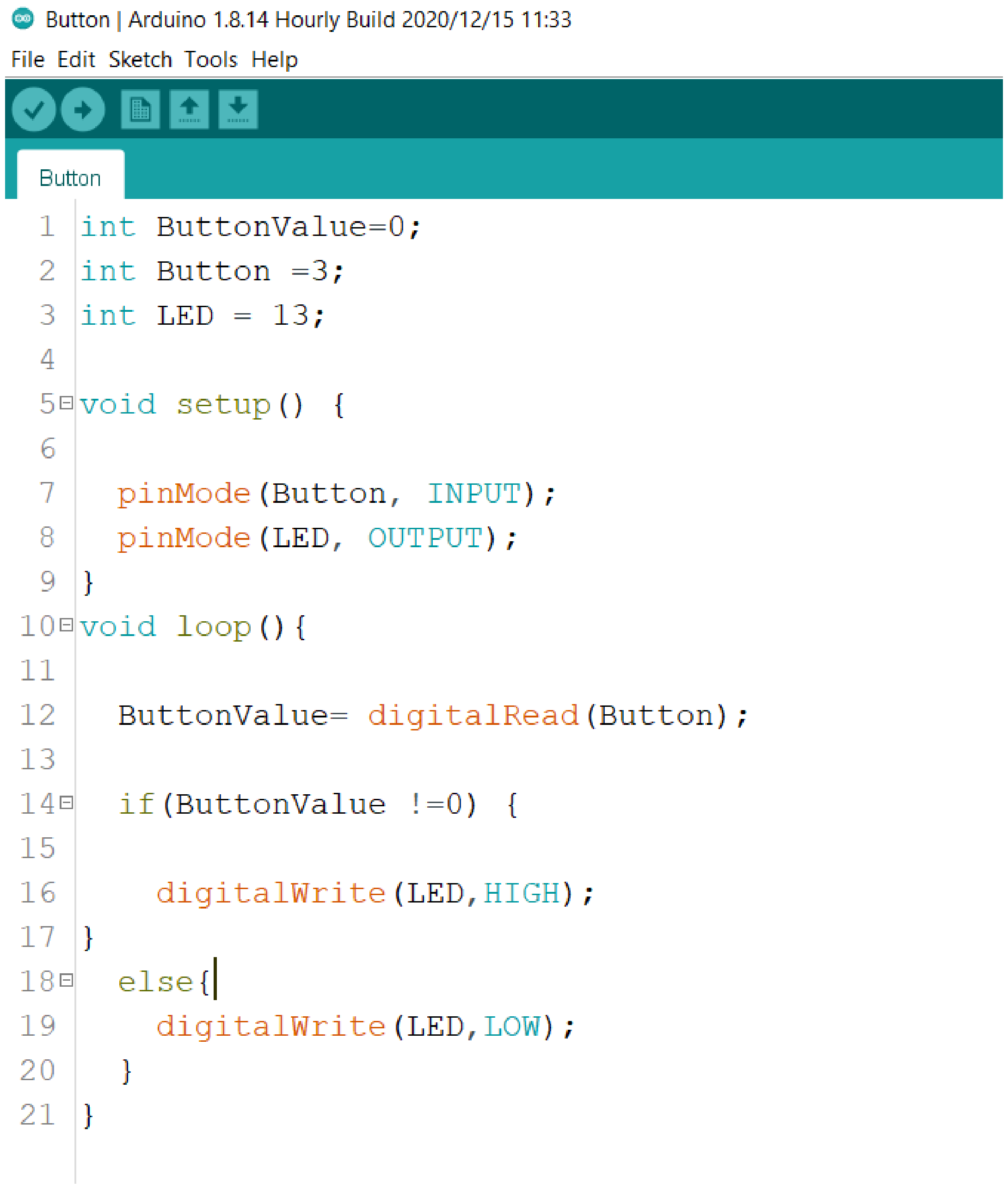



Pressed
1
2
3
4
-Pins 1 and 3 are wired together.
-Pins 2 and 4 are wired together.

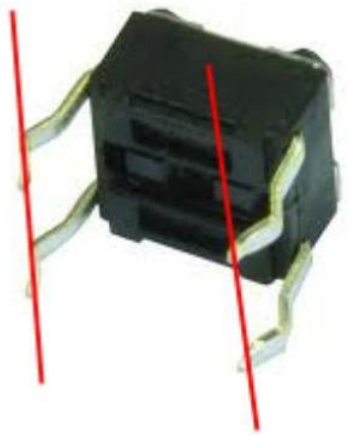
Circuit Design

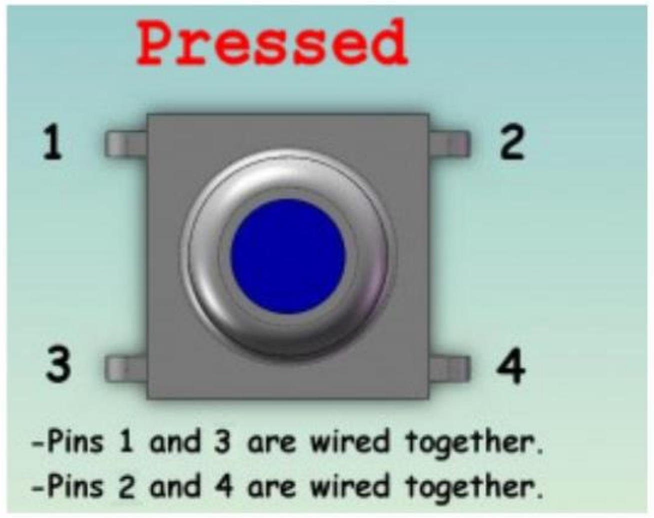


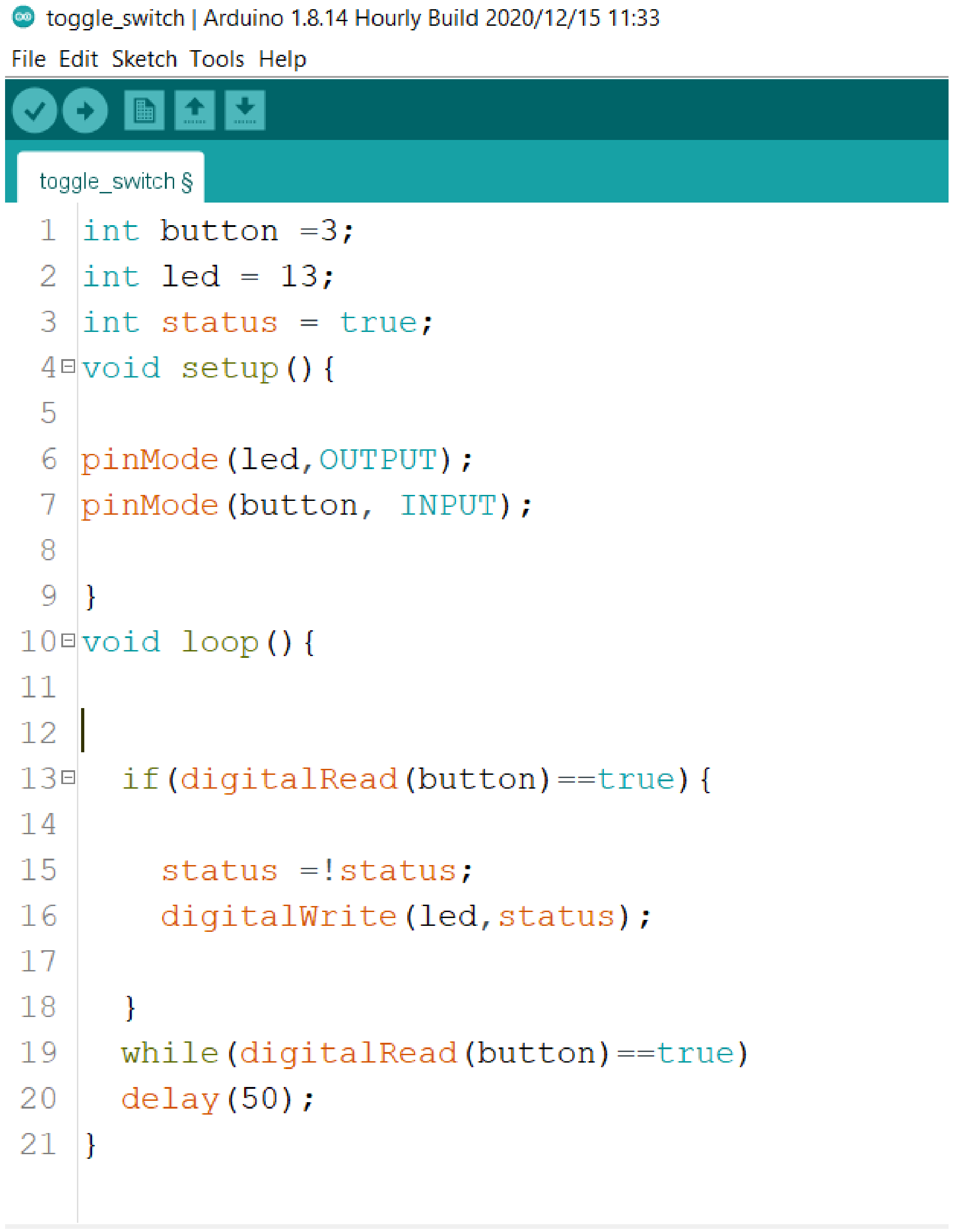
Program (Pressed Switch)


Button | Arduino 1.8.14 Hourly Build 2020/12/15 11:33
File Edit Sketch Tools Help
Button

```
int ButtonValue=0;
int Button =3;
int LED = 13;

void setup() {

  pinMode(Button, INPUT);
  pinMode(LED, OUTPUT);
}
void loop(){

  ButtonValue= digitalRead(Button);

  if(ButtonValue !=0) {

    digitalWrite(LED,HIGH);
}
  else{
    digitalWrite(LED,LOW);
  }
}
```



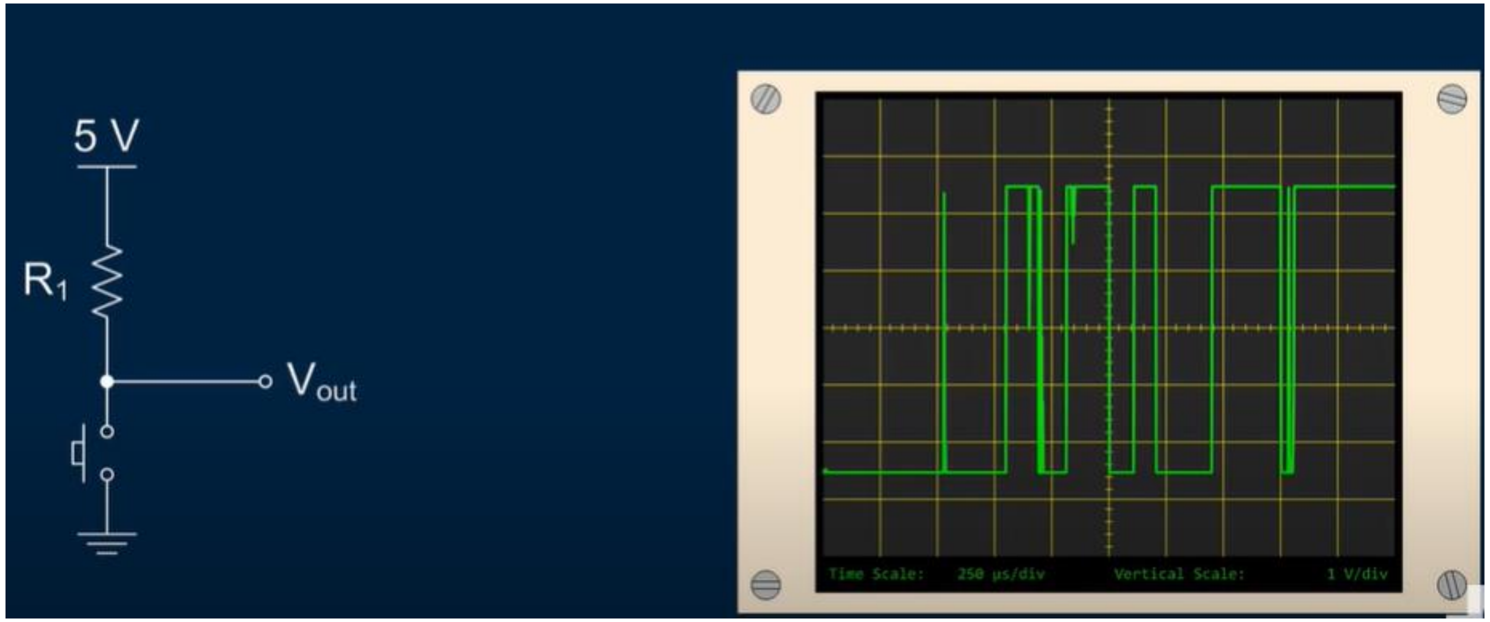

Pressed
1
2
3
4
-Pins 1 and 3 are wired together.
-Pins 2 and 4 are wired together.

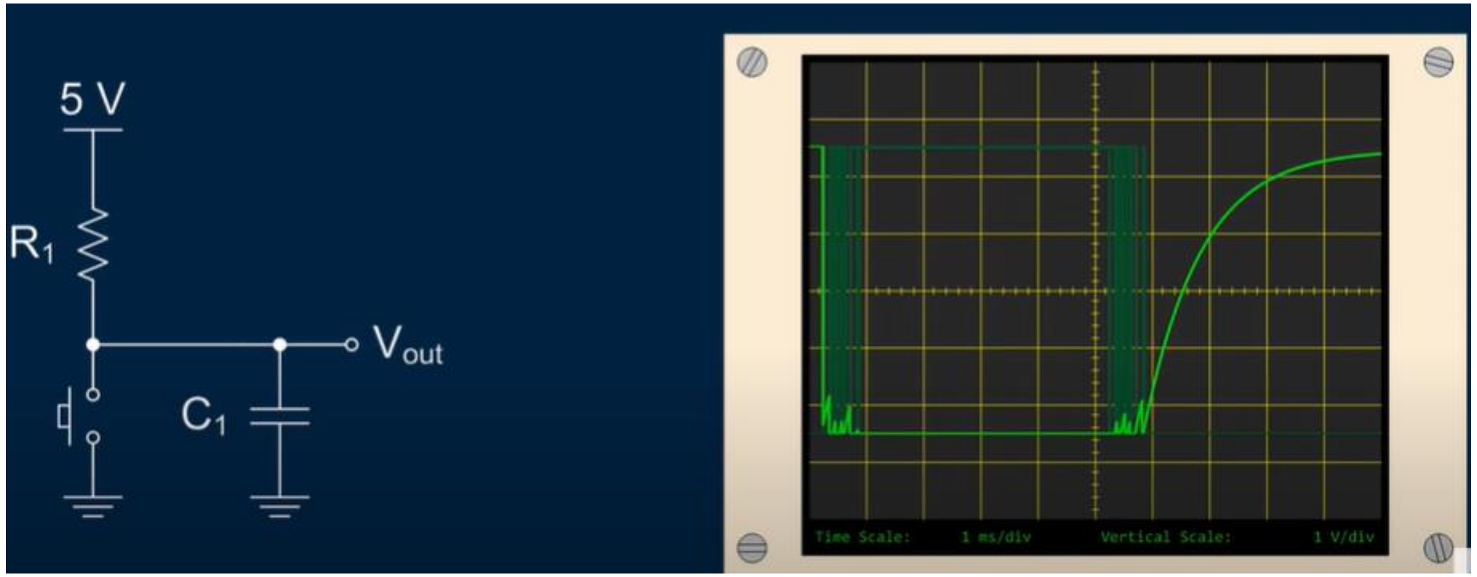

toggle_switch | Arduino 1.8.14 Hourly Build 2020/12/15 11:33

```
int button =3;
int led = 13;
int status = true;
void setup(){

pinMode(led,OUTPUT);
pinMode(button, INPUT);

}
void loop(){


  if(digitalRead(button)==true){

    status =!status;
    digitalWrite(led,status);

  }
  while(digitalRead(button)==true)
  delay(50);
}
```

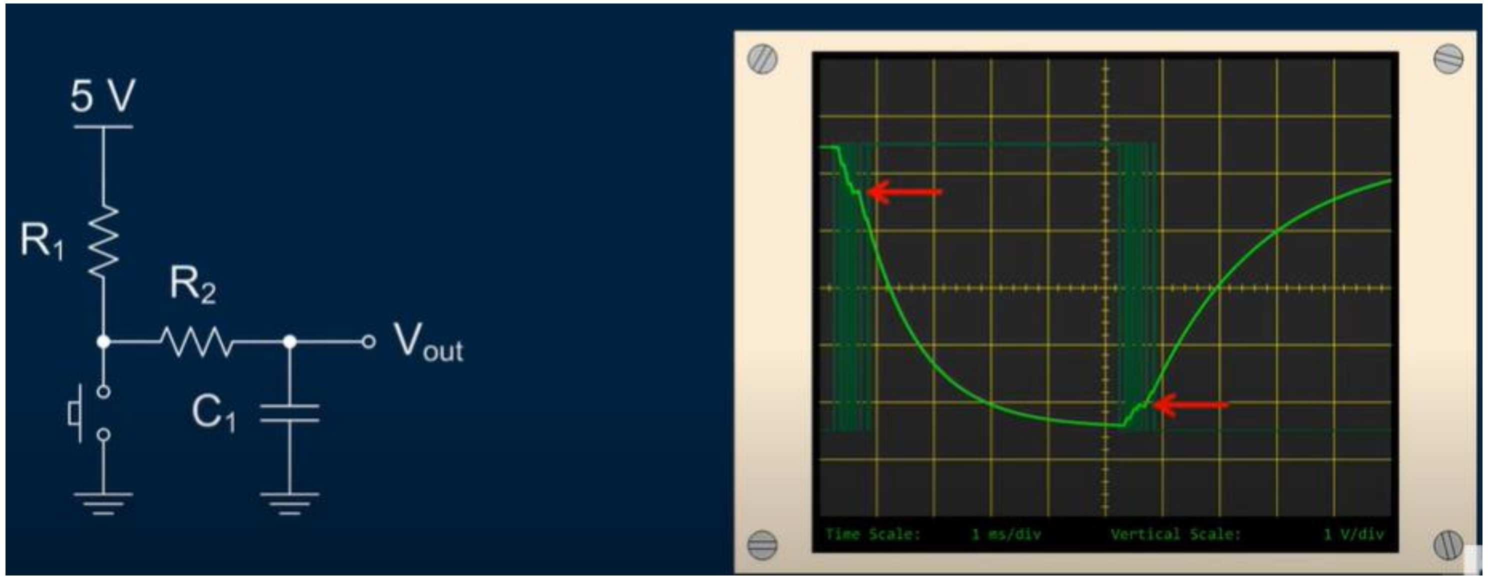
Program (Toggle Switch)

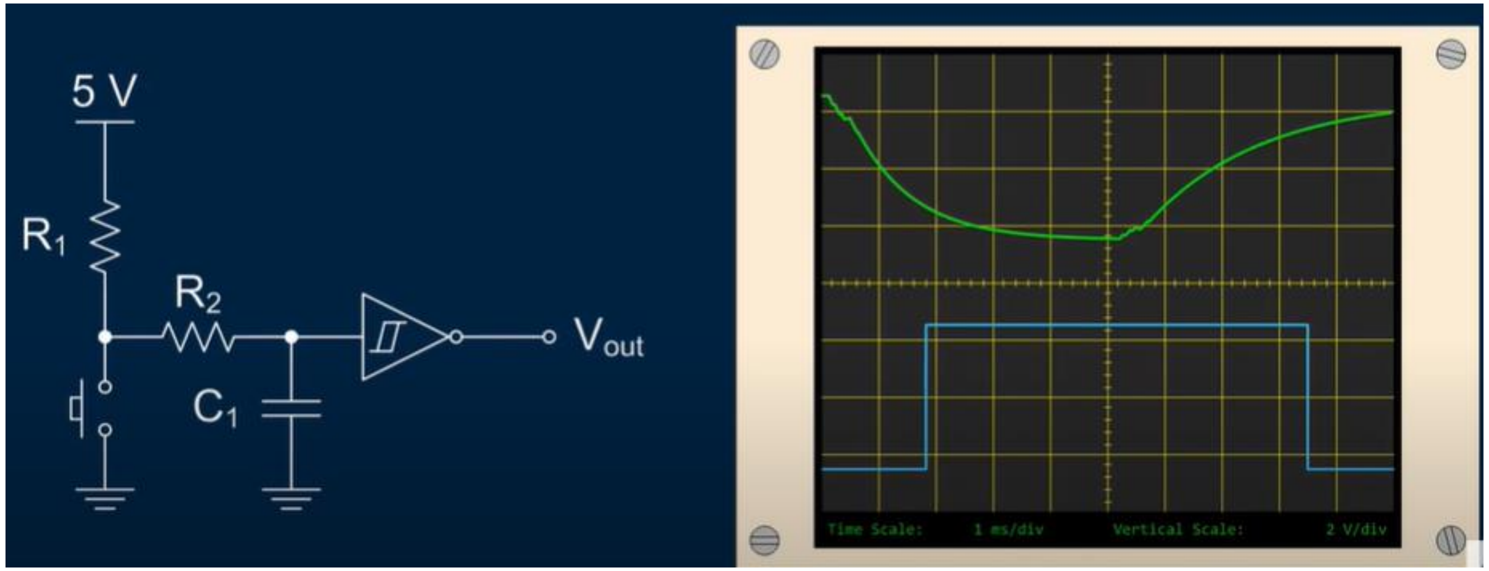
Debounce


5 V
R1
Vout
Time Scale: 250 µs/div
Vertical Scale: 1 V/div


5 V
R1
Vout
C1
Time Scale: 1 ms/div
Vertical Scale: 1 V/div

Debounce


5 V
R1
R2
Vout
C1
Time Scale: 1 ms/div
Vertical Scale: 1 V/div


5 V
R1
R2
Vout
C1
Time Scale: 1 ms/div
Vertical Scale: 2 V/div

Program

- Arduino IDE> File> Examples> Digital> Debounce

RADIO FREQUENCY

RF

CONTENT

Radio Frequency (RF) Introduction

RF Features

RF Module

Transmitter & Receiver

Types of Remote

Robot(Soccer & Battlebot)

INTRODUCING

Heinrich Hertz proved the existence of radio waves in the late 1880s.

• Radio frequency (RF) is a rate of oscillation in the range of around 3 kHz to 300 GHz, which corresponds to the frequency of radio waves, and the alternating currents which carry radio signals.

Over 40 millions systems manufactured each year utilizing lowpower wireless (RF) technology for data links, telemetry, control and security.

RF FEATURES

Power supply 4.5 V dc from three 1.5 V AAA batteries

Operating frequency: 916.50 MHz

Maximum data rate: 22.5 kbps

Good operating range

Operate over distances of 3 to 30 meters

RF MODULE

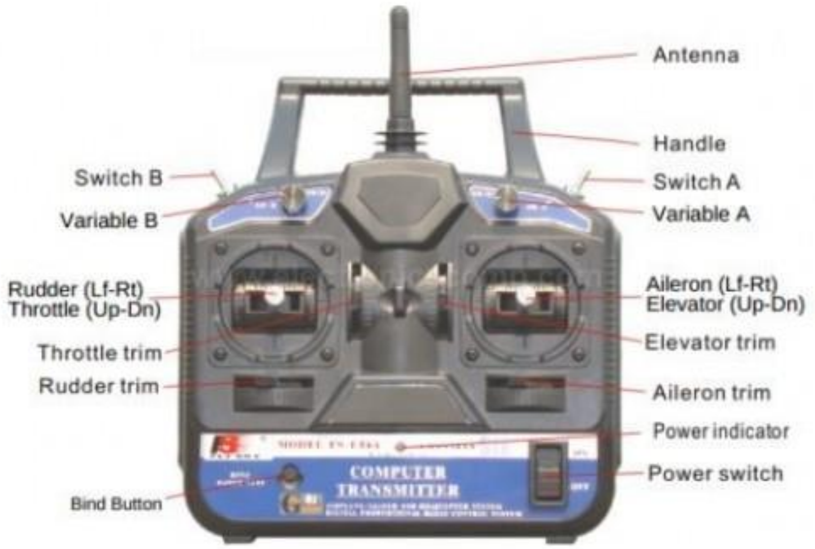
The corresponding frequency range varies between 30 kHz & 300 GHz.

Transmission through RF is better than IR (infrared) .

Signals through RF can travel through larger distance.

This RF module comprises of an RF Transmitter and an RF Receiver.

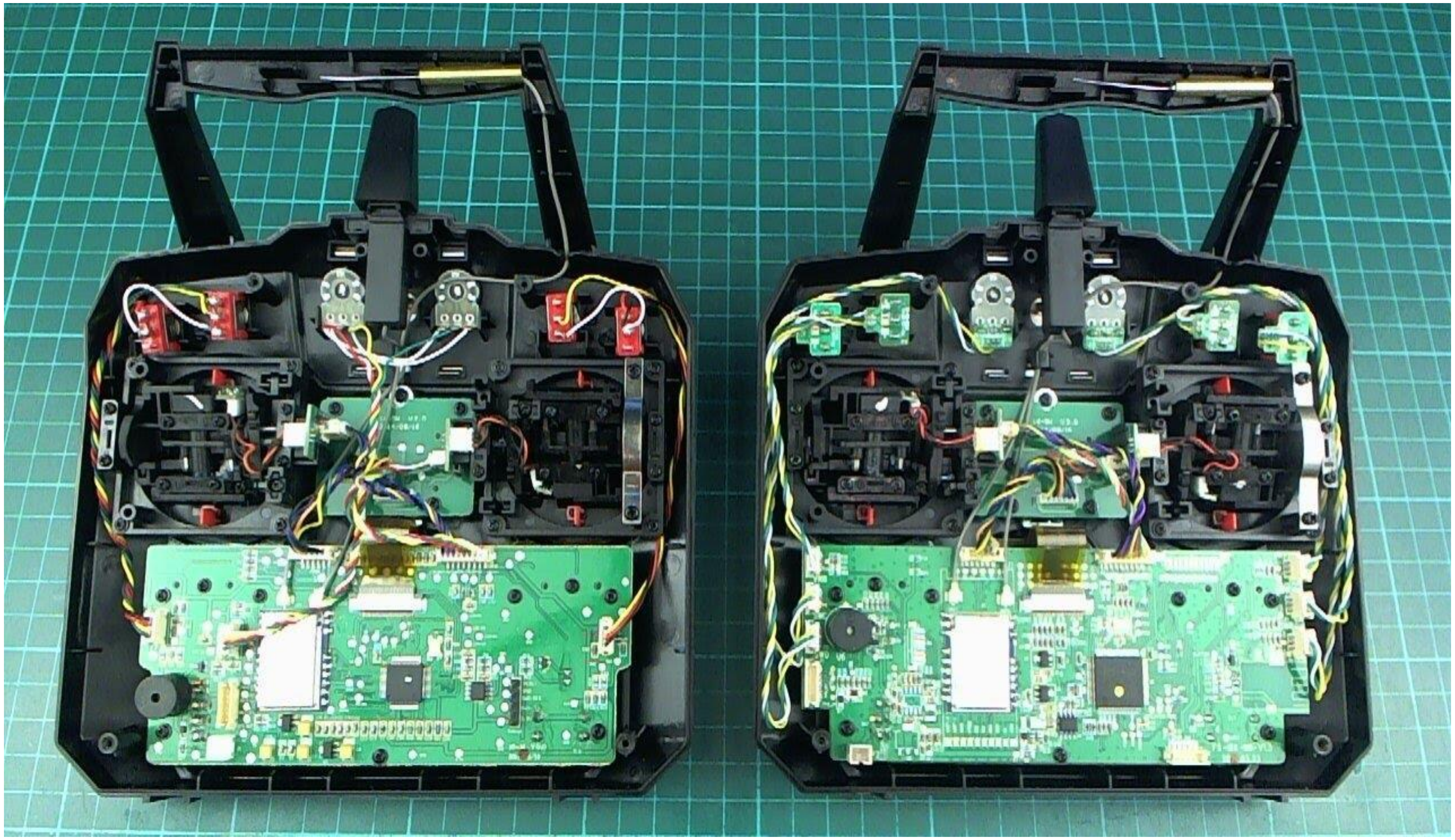
TRANSMITTER



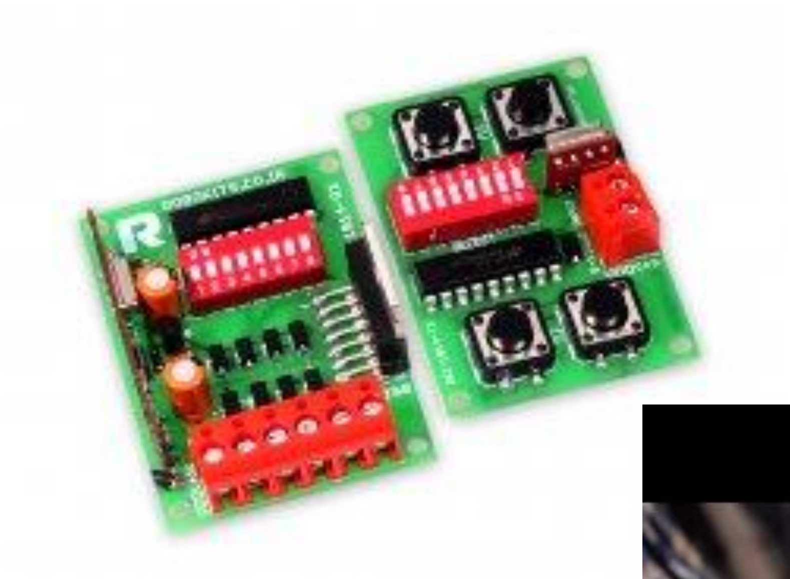
Receiver

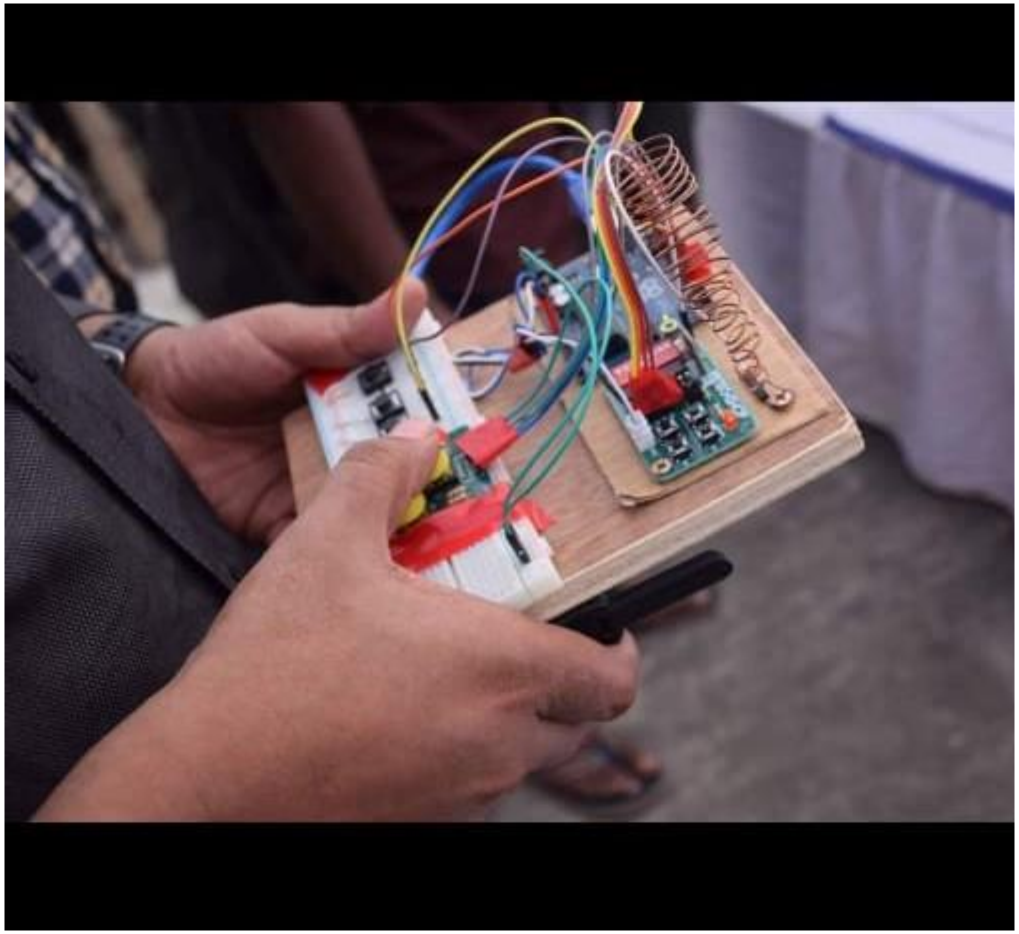
LED
UPDATE
2.4GHz 6 CHANNELS RECEIVER
4.0-6.5V/DC
FCC ID:
N4ZFLYSKYIA6B
CE0678
FS-iA6B
B/VCC
CH6
CH5
CH4
CH3
CH2
PPM/CH1
SERVO
i-BUS
SENS

5V
GND
ZLH
3.3V
FS-iA6
20130217
GND
GND
BAT



Hand Made Transmitter





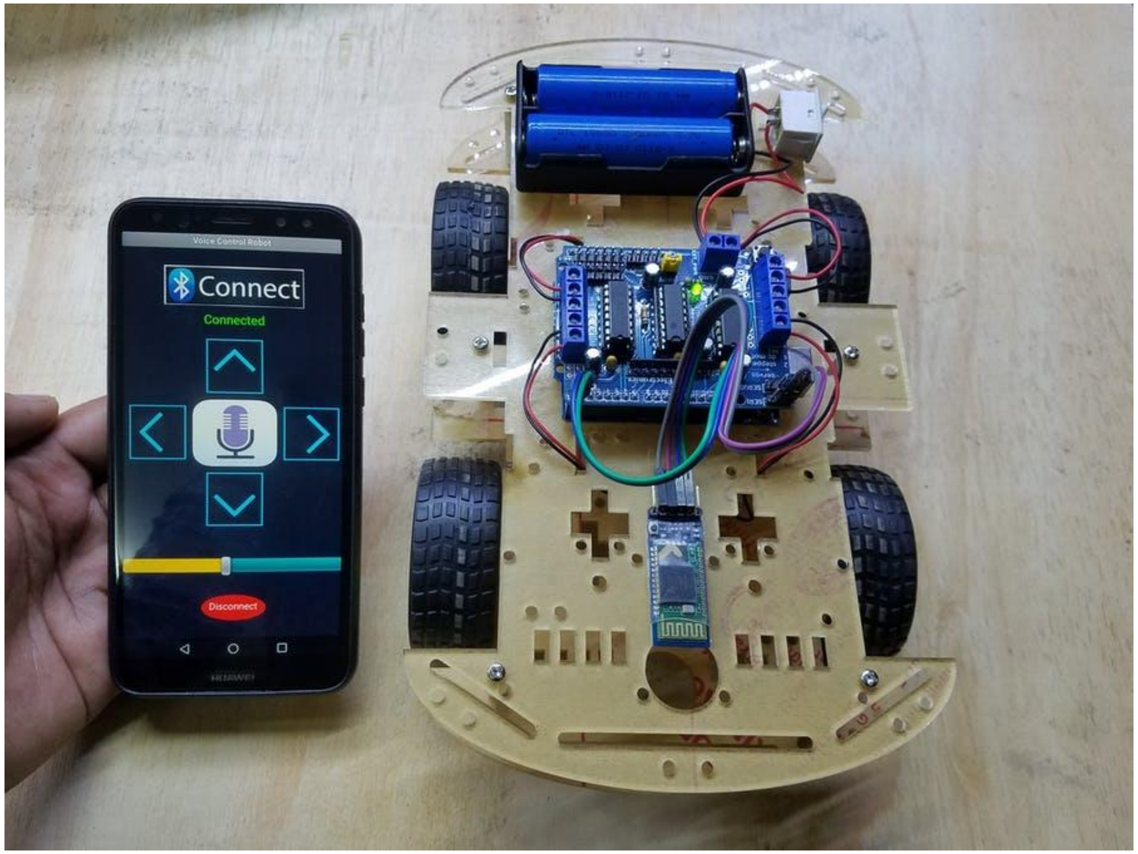
TYPES OF REMOTE

Three types of remote

1. Infrared remote control
2. Voice control
3. Radio remote control

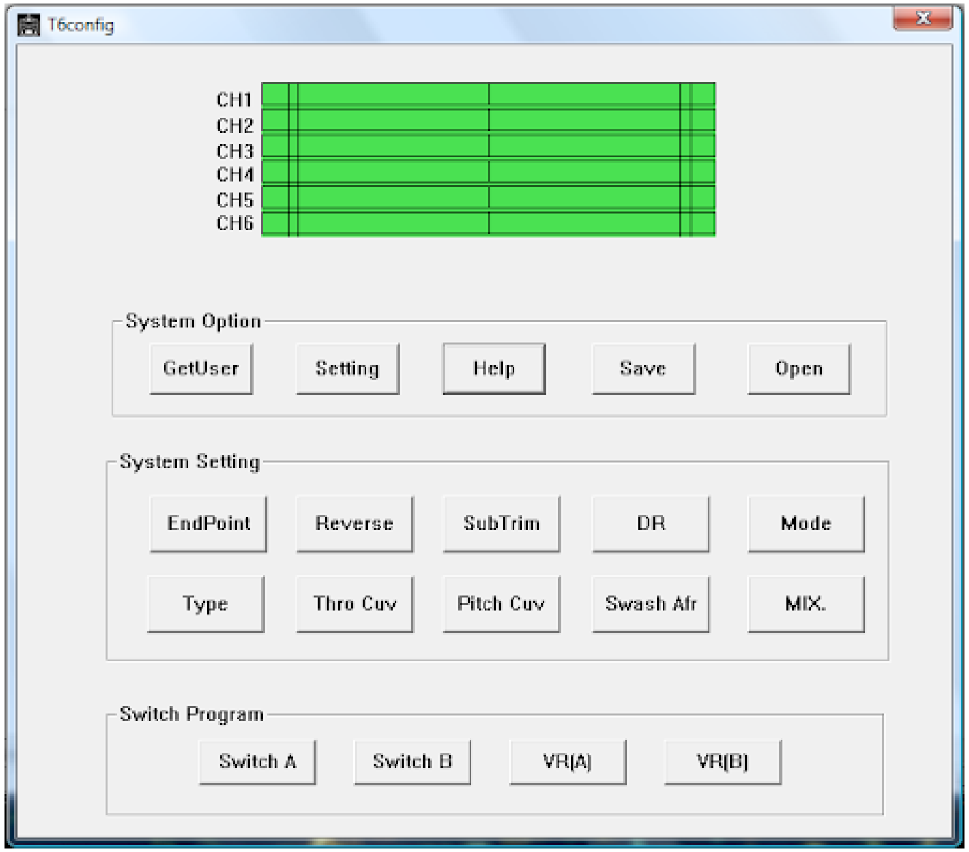
INFRARED REMOTE CONTROL

VOICE CONTROL



RADIO REMOTE CONTROL

REMOTE CONFIGURATION

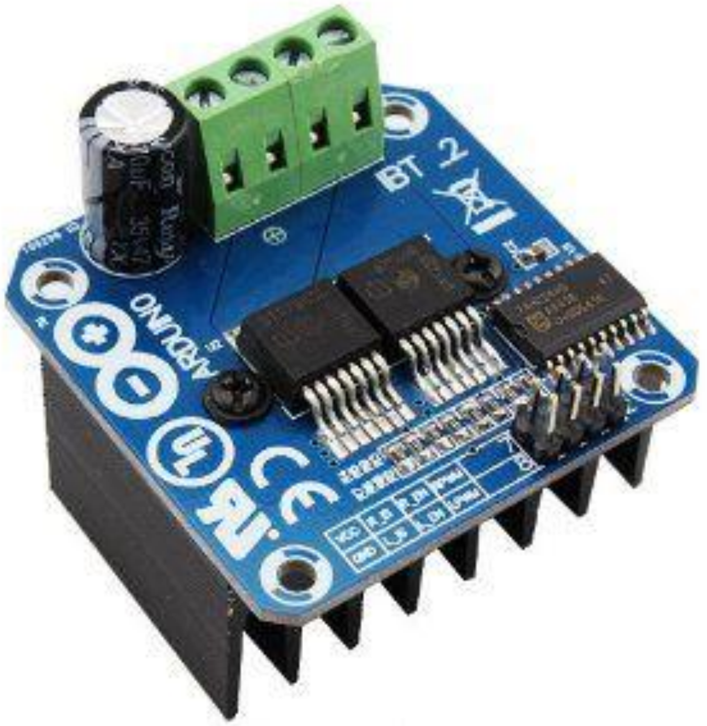


ROBOT

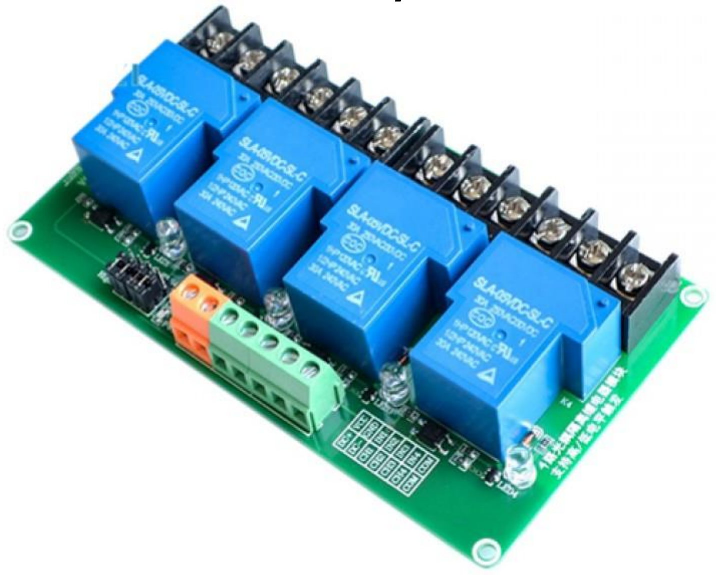
Soccer & Battle bot

SOCCER

1. Mechanical part .
2. Power supply.
3. Controlling.


BTS Motor Driver

Relay



ESC Motor Driver

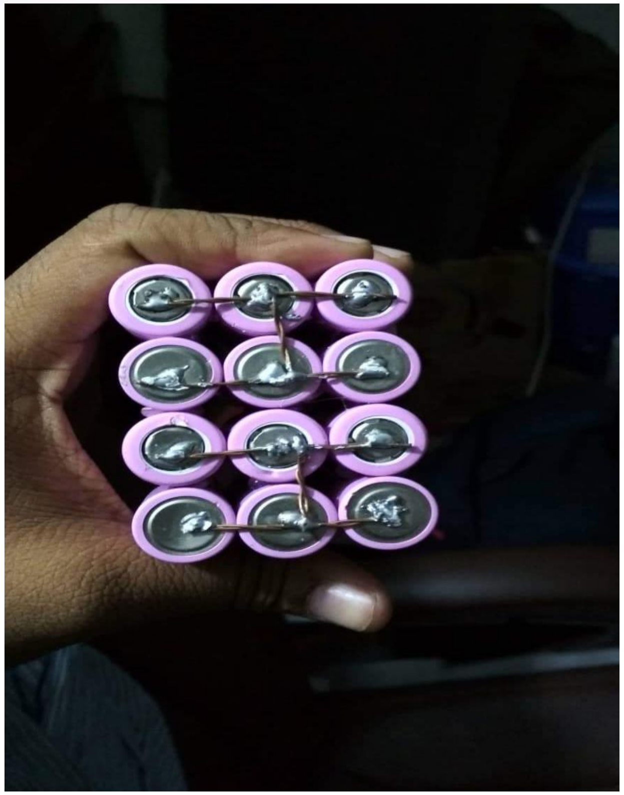
1. MECHANICAL PART

DC Motor

Power Supply

HPB POWER
Powered radio controlled models
High magnification lithium polymer battery
2200mAh
•7.4V •11.1V •14.8V •18.5V •22.2V
25C •30C •35C •45C •60C
RoHS

Li-ion 18650C 1500mAh 3.7V
Li-ion 18650C 1800mAh 3.7V
Li-ion 18650C 2000mAh 3.7V
Li-ion 18650C 2200mAh 3.7V
Li-ion 18650C 2400mAh 3.7V
Li-ion 18650C 2600mAh 3.7V



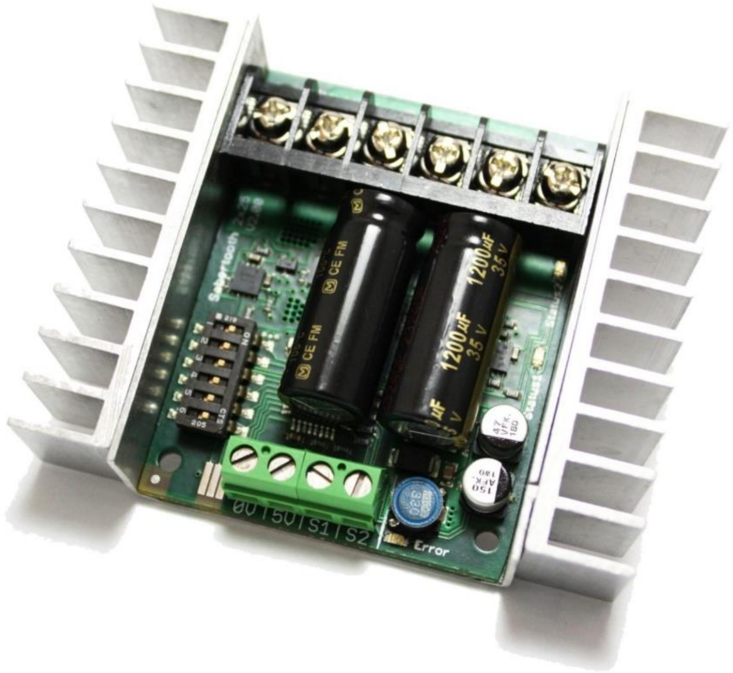
BATTLEBOT

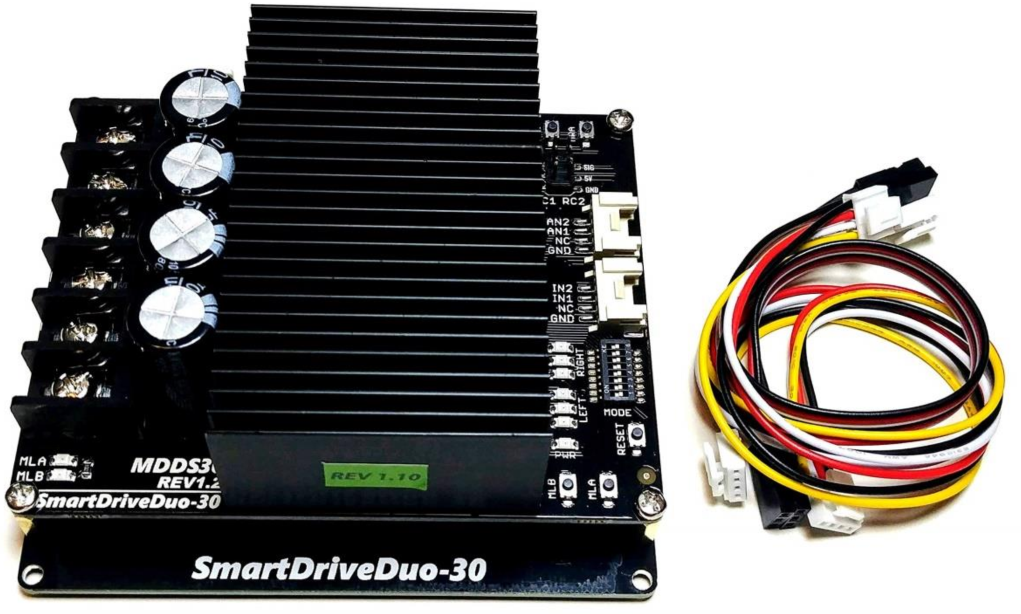
1. Mechanical part .
2. Power supply.
3. Controlling.

DC Motor

Brushed ESC

High Current Motor Driver





ESC

AEROSTAR
120A
OPTO
10-38 NiMH/5-12 Lipo
10AWG

RED BRICK
60A ESC
Ni-MH: 6-20 CELLS
LIPO: 2-7 CELLS
A
B
C
RED BRICK
70A ESC UBEC 5V/5A
Ni-MH: 6-20 CELLS
LIPO: 2-7 CELLS
A
B
C

COMPETITION
IEEE IIUC
Wake Up With Better Network

Wheel

Weapon