SOFTWARE ROBOT DESIGN LAB (SMRA2403)

List of Experiments

Experiment 1

Aim: Study of Arduino Uno R3 microcontroller, breadboard, components and cables and application

Arduino is an open source programmable circuit board that can be integrated into a wide variety of projects both simple and complex. This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world. By responding to sensors and inputs, the Arduino is able to interact with a large array of outputs such as LEDs, motors and displays. Because of it's flexibility and low cost, Arduino has become a very popular choice for students looking to create interactive hardware projects.

Arduino was introduced back in 2005 in Italy by Massimo Banzi as a way for non-engineers to have access to a low cost, simple tool for creating hardware projects. Since the board is open-source, it is released under a Creative Commons license which allows anyone to produce their own board. If we search the web, we will find there are hundreds of Arduino compatible clones and variations available but the only official boards have Arduino in it's name. In the next section, we're going to discuss Arduino Uno board that we will use in our lab for coding and experiment.

Arduino Uno

One of the most popular Arduino boards out there is the Arduino Uno. While it was not actually the first board to be released, it remains to be the most actively used and most widely documented on the market. Because of its extreme popularity, the Arduino Uno has a ton of project tutorials and forums around the web. Arduino Uno board is shown in the Figure 1 with important components.

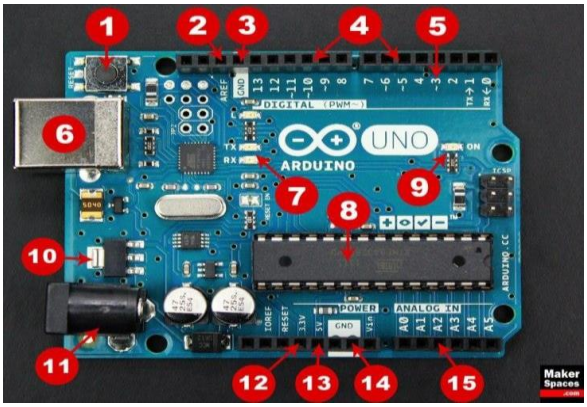


Figure 1 Arduino Uno board with important components

Board Breakdown

Here are the components that make up an Arduino board and what each of theirfunctions are.

1) **Reset Button** – This will restart any code that is loaded to the Arduino board
2) **AREF** – Stands for "Analog Reference" and is used to set an external referencevoltage
3) **Ground Pin** – There are a few ground pins on the Arduino and they all work thesame
4) **Digital Input/Output** – Pins 0-13 can be used for digital input or output

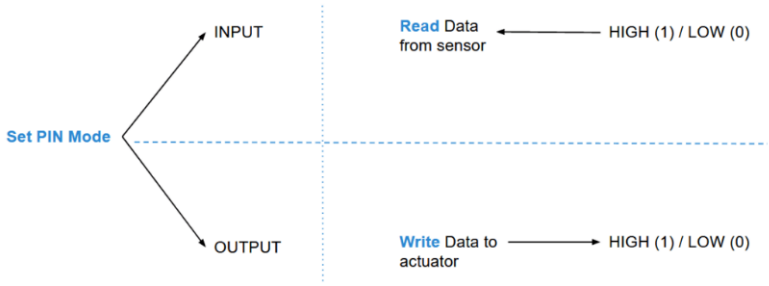


Figure 2 Input and output mode of digital pin

- OUTPUT: this is to write data to an actuator, for example an LED.
- INPUT: in this case you're going to read data from the sensor. The value you'll get will be HIGH or LOW (binary).

5) **PWM** – The pins marked with the (~) symbol can simulate analog output
6) **USB Connection** – Used for powering up your Arduino and uploading sketches
7) **TX/RX** – Transmit and receive data indication LEDs
8) **ATmega Microcontroller** – This is the brains and is where the programs arestored
9) **Power LED Indicator** – This LED lights up anytime the board is plugged in a power source
10) **Voltage Regulator** – This controls the amount of voltage going into the Arduinoboard
11) **DC Power Barrel Jack** – This is used for powering your Arduino with a powersupply
12) **3.3V Pin** – This pin supplies 3.3 volts of power to your projects
13) **5V Pin** – This pin supplies 5 volts of power to your projects
14) **Ground Pins** – There are a few ground pins on the Arduino and they all work thesame
15) **Analog Pins** – These pins can read the signal from an analog sensor and convert it to digital. Analog pin is 10-Bit. So, it can 2^{10} = 1024 (0-1023) value. The total voltage it can address is 0-5 V. so 1V = 1023/5 = 204.5 value or 1 value = 5/1023 = 4.9 mV.

Arduino Power Supply

The Arduino Uno needs a power source in order for it to operate and can be powered in a variety of ways. You can do what most people do and connect the board directly to your

computer via a USB cable. If you want your project to be mobile, consider using a 9V battery pack to give it juice. The last method would be to use a 9V AC power supply.

Arduino Breadboard

Another very important item when working with Arduino is a solderless breadboard (Figure 3). This device allows you to prototype your Arduino project without havingto permanently solder the circuit together. Using a breadboard allows you to create temporary prototypes and experiment with different circuit designs. Inside the holes (tie points) of the plastic housing, are metal clips which are connected to each other bystrips of conductive material.

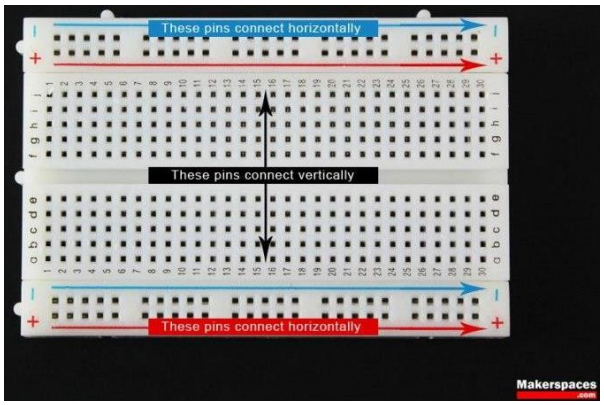


Figure 3 Breadboard internal pins connection

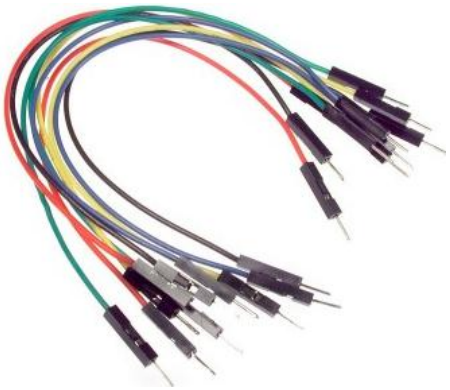


Figure 4 Jumper wire

On a side note, the breadboard is not powered on its own and needs power brought to it from the Arduino board using jumper wires (Figure 4).

Button Switch

A switch is a component which controls the open-ness or closed-ness of an electric circuit. Pushbutton switches (Figure 5) are two-position devices actuated with a button that is pressed and released.

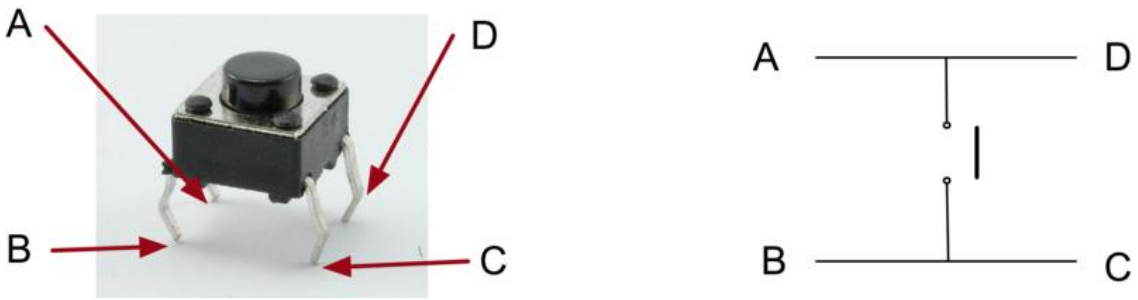


Figure 5 Push button switch internal pin connection

Light Emitting Diode (LED)

An LED (Figure 6) is a semiconductor device that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor.

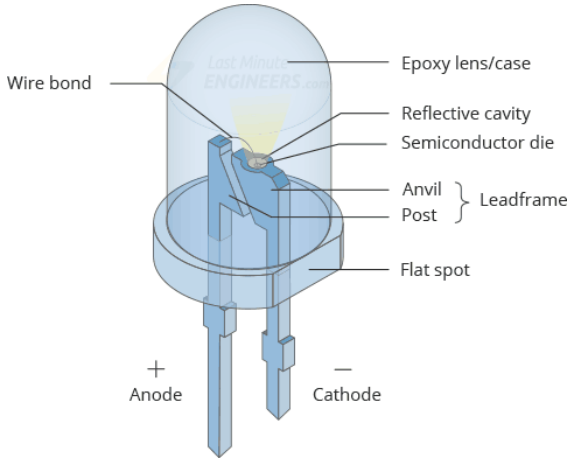
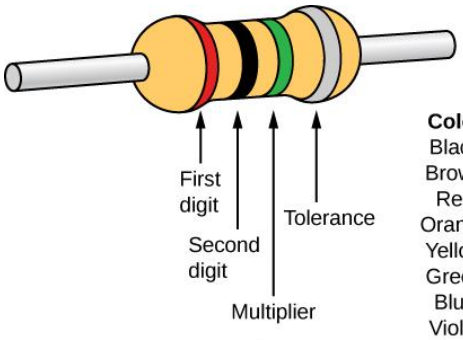


Figure 6 LED Schematic

Resistance and resistor

Resistance is a measure of the opposition to current flow in an electrical circuit. Resistance is measured in ohms, symbolized by the Greek letter omega (Ω). A resistor (Figure 7) is a device designed to produce resistance. Resistors can be used to limit current, divide voltage, or generate heat. There are two main types of resistors: fixed and variable.



Color	Digit	Multiplier
Black	0	10^0
Brown	1	10^1
Red	2	10^2
Orange	3	10^3
Yellow	4	10^4
Green	5	10^5
Blue	6	10^6
Violet	7	10^7
Gray	8	10^8
White	9	10^9

Tolerance	
Gold	5%
Silver	10%

Figure 7 Resistor and Its color code

Potentiometer

A potentiometer (Figure 8) is a manually adjustable variable resistor with 3 terminals. Two of the terminals are connected to the opposite ends of a resistive element, and the third terminal connects to a sliding contact, called a wiper, moving over the resistive element.

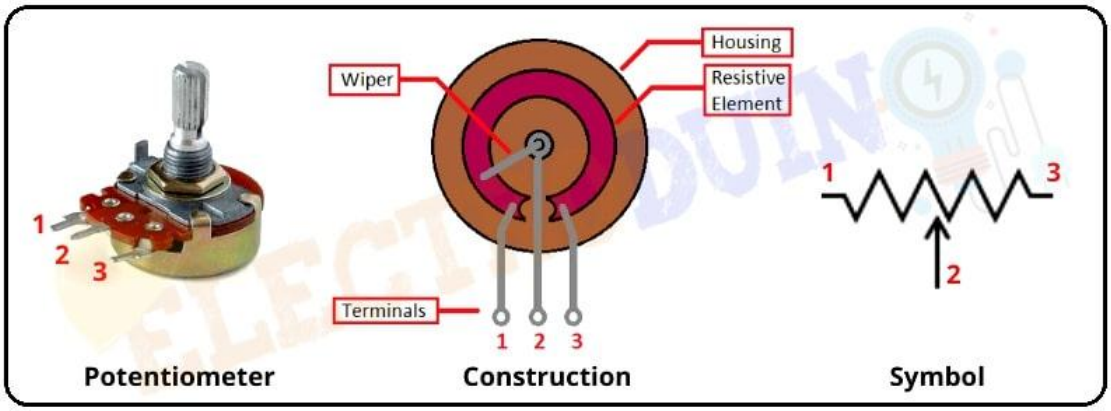


Figure 8 Potentiometer and its pin schematic

Capacitor

Capacitors (Figure 9) are passive electronic components consisting of two or more pieces of conducting material separated by an insulating material. The capacitor is a component which has the ability or "capacity" to store energy in the form of an electrical charge producing a potential difference (Static Voltage) across its plates, much like a small rechargeable battery.

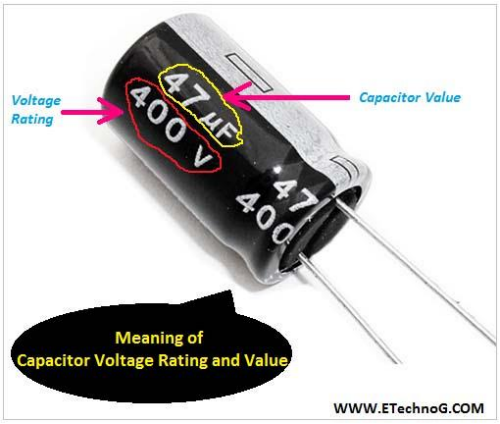


Figure 9 Capacitor shown with value and ratings

Some Important Concepts and Commands

Analog and Digital

Analog and digital signals (Figure 10) are the types of signals carrying information. The major difference between both signals is that the analog signals have continuous electrical signals, while digital signals have non-continuous electrical signals. Temperature sensors, FM radio, Photocells, Light sensor, Resistive touch screen, analog watch and multimeter are examples of Analog device. Computers, CDs, DVDs, Mobile, IPod, Digital multimeter are some examples of Digital signal.

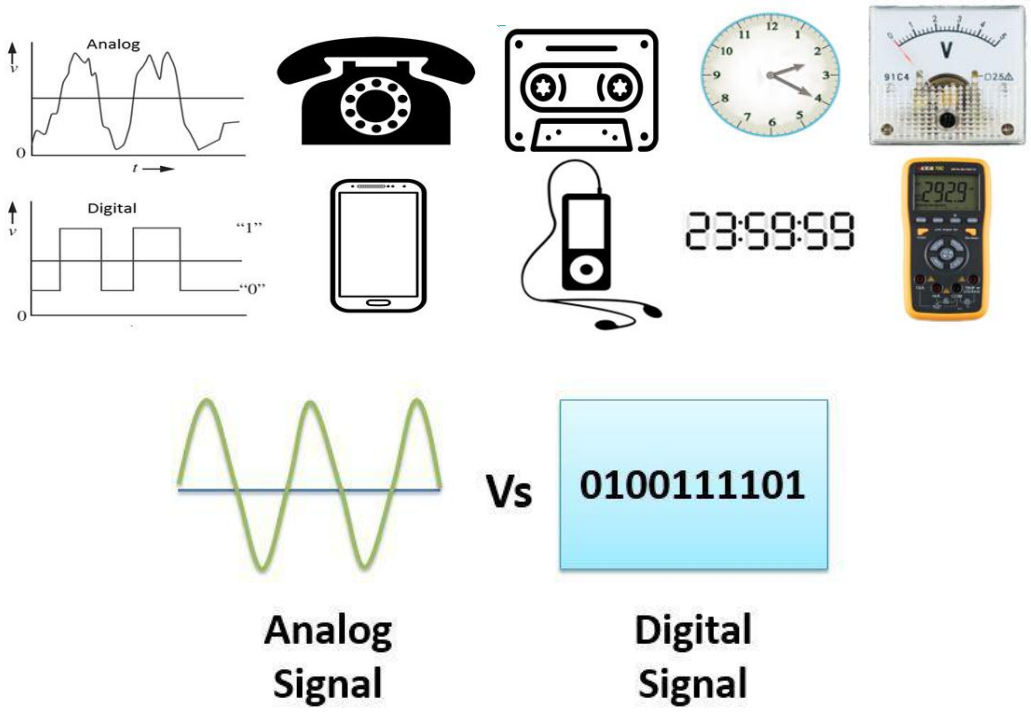


Figure 10 Analog and digital signal, device and difference

Pulse Width Modulation (PWM)

The PWM (Figure 11) is a method of controlling the average voltage. It is a stream of voltage pulses that reduces the electric power supplied by the electrical signal. The effective voltage is controlled by the width of individual pulses in a stream of voltage pulses of a PWM signal. The common use of PWM pins includes controlling LEDs and DC Motors. The PWM in LED controls the frequency of the light. It means the LED will be ON/OFF at a frequency detectable by our eyes. The PWM in DC Motors acts like a pulse train of a DC signal. The DC motors

receive a high or low electrical power input based on the width of the PWM pulses. So the speed is controlled using PWM. PWM is generated by analogWrite() command. AnalogWrite resolution in Arduino is 8 Bit. So the resolution value will 256 (0-255). In the small bracket we will fill the resolution value as per the percentage of duty cycle as shown in Figure

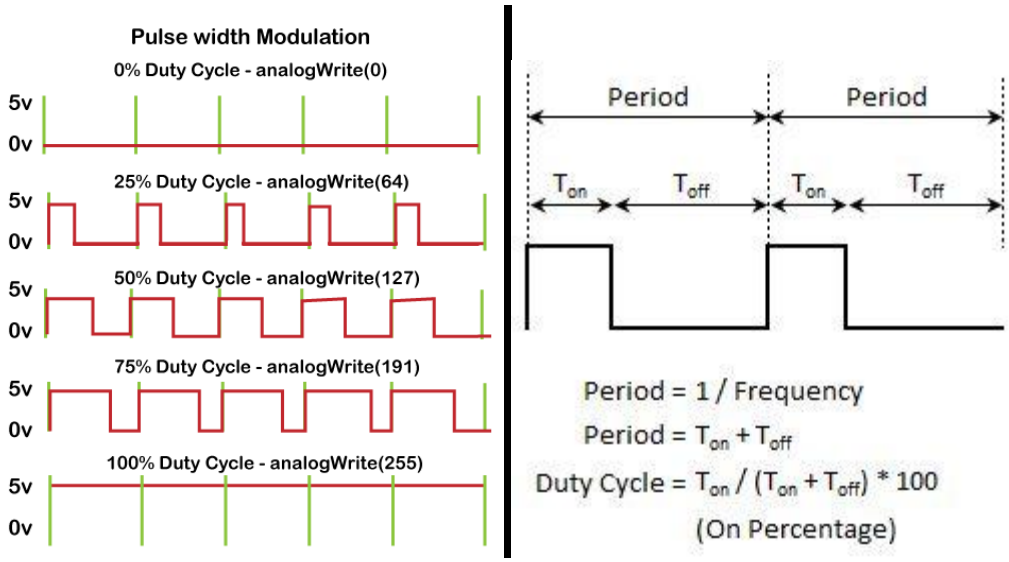


Figure 11 PWM and duty cycle calculation

Pull up and pull down resistor

Pull-up and Pull-down resistors (Figure 12) are used to correctly bias the inputs of digital gates to stop them from floating about randomly when there is no input condition. When the switch is open, the voltage of the gate input is pulled up to the level of the input voltage. When the switch is closed, the input voltage at the gate goes directly to the ground. So, pull up makes high output (1) in normal condition (unpressed button) and low output in pressed condition. Similarly, pull down makes low output (0) in normal condition (unpressed button) and high output (1) in pressed condition.

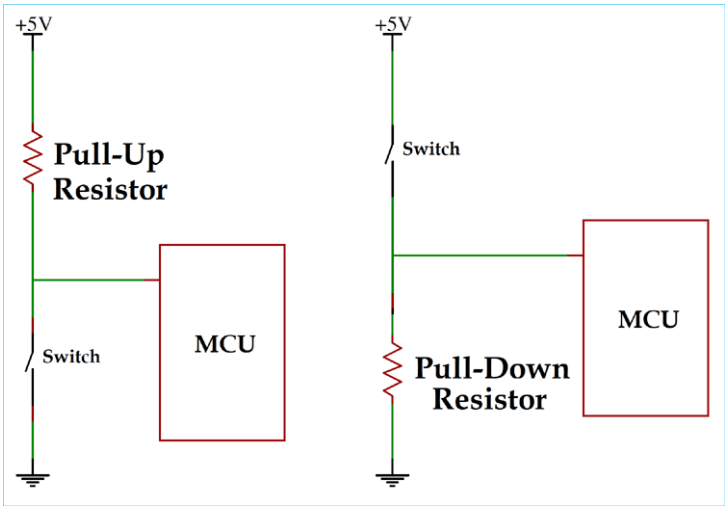


Figure 12 Pull up and pull down resistor connection

Experiment 2

Aim: Study of Arduino Installation, Programming and Coding

Once the circuit has been created on the breadboard, we'll need to upload the program (known as a sketch) to the Arduino. The sketch is a set of instructions that tells the board what functions it needs to perform. An Arduino board can only hold and perform one sketch at a time. The software used to create Arduino sketches is called the IDE which stands for Integrated Development Environment. The software is free to download and can be found at https://www.arduino.cc/en/software.

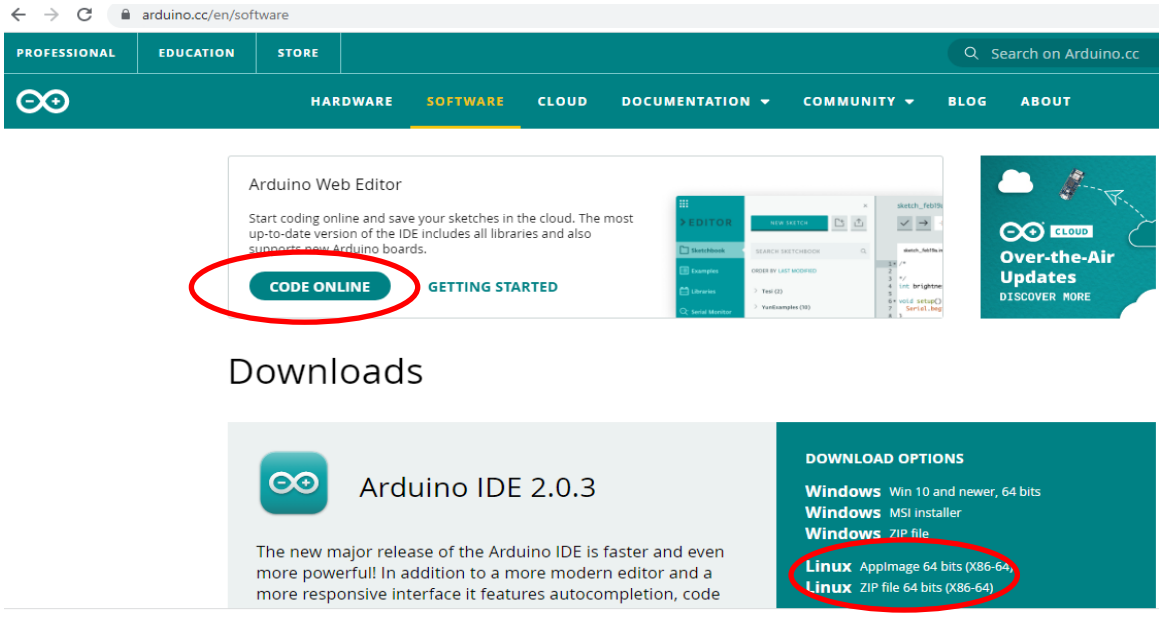


Figure 1 Arduino software downloading option

In the Arduino software link, we will get downloading option. It's better to download Windows ZIP file, so that we can get driver folder. There is also an option for coding online.

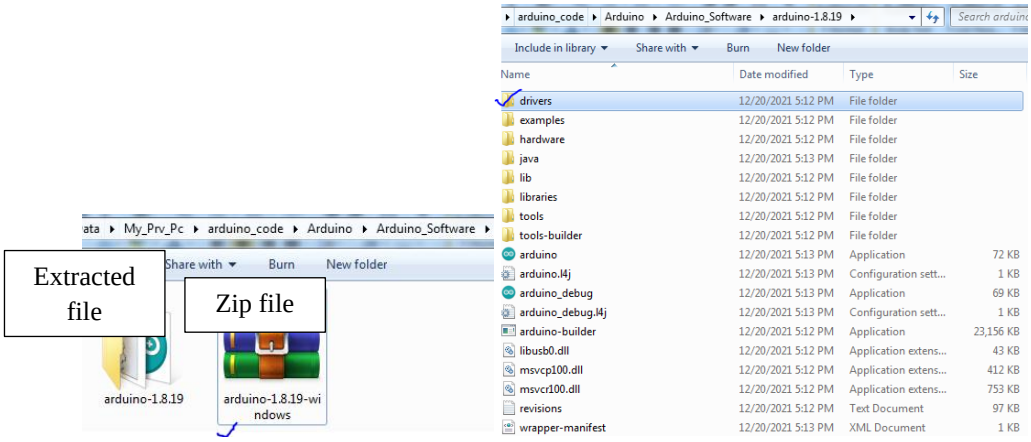


Figure 2 Arduino software zip with application and driver

The Basics

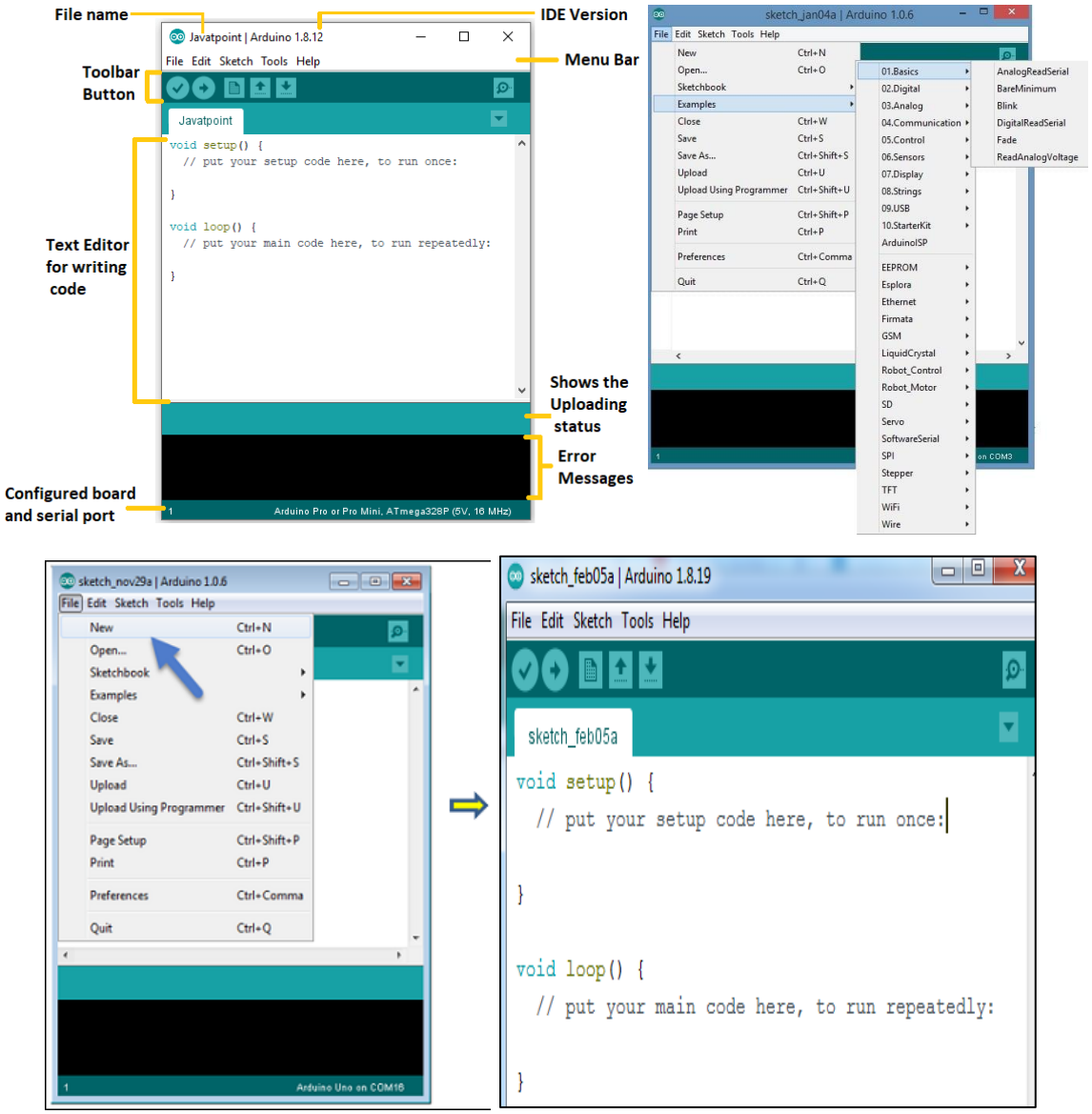


Figure 3 Arduino software tools, example and new sketch for coding

As soon as you open the Arduino IDE, you will be greeted by the setup() and loop() commands. These are the only two examples of "sketches" you will find in almost all Arduino codes.

Setup(): Every time a sketch starts, there is a setup command which will help you initialize the variables, start using libraries and such. Refer the previous example of digital LED blinking without delay in the article on Arduino IDE installation.

```
void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}
```

Loop(): A loop follows setup and is really the heart of the program, making it respond infinitely to any logic. For example, the above example is about LED blinking infinitely without delay.

```
void loop() {
  // here is where you'd put code that needs to be running all the time.
// check to see if it's time to blink the LED...
}
```

After you become familiar with the sketches, we need to know the control commands. The most important ones are:

Break: if you want to exit from a command, you need to hit break.

```
if (sens > threshold) {
   // bail out on sensor detect   x = 0;     break;
}
```

If or else: logical commands that initiate an action each time a condition is fulfilled. Again, go back to the example of digital LED blinking. Remember the loop() instruction where a code has to run infinitely.

Other useful control commands which can deliver certain logic, and which you might use, include:

- **return**: return a certain value.
- **while**: another loop that goes on continuously for a certain condition. For example, while (dist <3 cm) do something repetitive 200 times)
- **goto**: as the name obviously suggests, this command lets you go to a certain line in the code.

The Booleans and the Arithmetic Operators

Next to sketch and control, you must know some Boolean and arithmetic operators to command the programs.

The operands: is equal to (=), addition (+), subtraction (-), multiplication (*) and division (/).

Advanced operands: is not equal to (!=), less than or equal to (<=), greater than or equal to (>=), remainder (%).

Important Variables

At some point, you need to introduce a few variables to deal with the various logical operations. The important ones are:

HIGH|LOW: This gives the high and low end values of the constants.

```
// Variables will change:
 int ledState = LOW;
```

LED_BUILTIN: gives the number of LED pins. In the above example of on-board LED light blinking in Uno, that would be LED pin number 13.

Other important variables to remember include **True/False**, **sizeof()**, **void**, **int** and **string**. They follow the same meaning as any other conventional program including Python, C++, etc.

Advanced Functions

Finally, you need to know a few advanced functions that control the Arduino board. These include:

digitalRead(): Reads the value from a given digital pin. There is also **digitalWrite()**.

random(): this helps generate random numbers.

Tone() and **noTone()**: If we want a tone sound in your pin? Then this command will take care of that, whereas the noTone() will keep things silent.

Delay(): recall the example of LED light blinking. You can introduce a delay into it.

Observation

The above commands are some of the most useful in dealing with Arduino boards on your IDE. The list is very small and not exhaustive, but it can help you get started with your projects. For details, you need to study the sketches by other Arduino hobbyists to understand what you can learn from them.

Experiment 3

Aim: Coding to blink an LED

Components: List of components required to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno R3
- 1 × LED
- 1 × 330Ω Resistor
- 2 × Jumper wire

Circuit Diagram

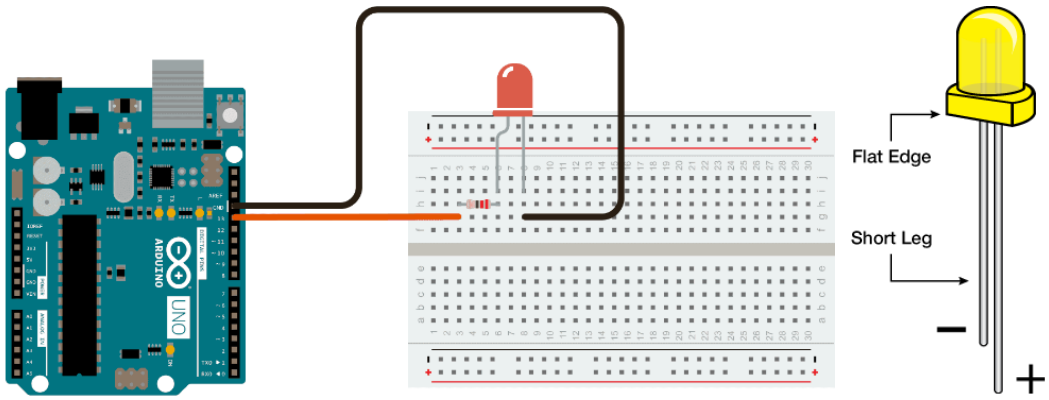


Figure 1 Circuit diagram for LED blinking

Procedure

In this +ve pin (longer pin) of led is connected to pin 13 through resistor (330Ω) and shorter pin is connected to GND as shown in Figure 1.

Programming

After circuit connection, open the Arduino IDE software on your computer. Open the new sketch File by clicking File- New. And type the following code:

```
#define LED_PIN 13
void setup()
{
pinMode(LED_PIN, OUTPUT);
```

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

Results & Observation

Experiment 4

Aim: To Turn LED ON and OFF With Button using Arduino controller

Components: List of components required to build the circuit:

- 1 × Arduino board
- 1 × Breadboard
- 1 × LED – any color
- 1 × Push button
- 1 × Any resistor from 330 to 1k Ohm
- 1 × 10k Ohm resistor (if not go with 20k-50k Ohm).
- 6 × Jumper wire

Circuit Diagram

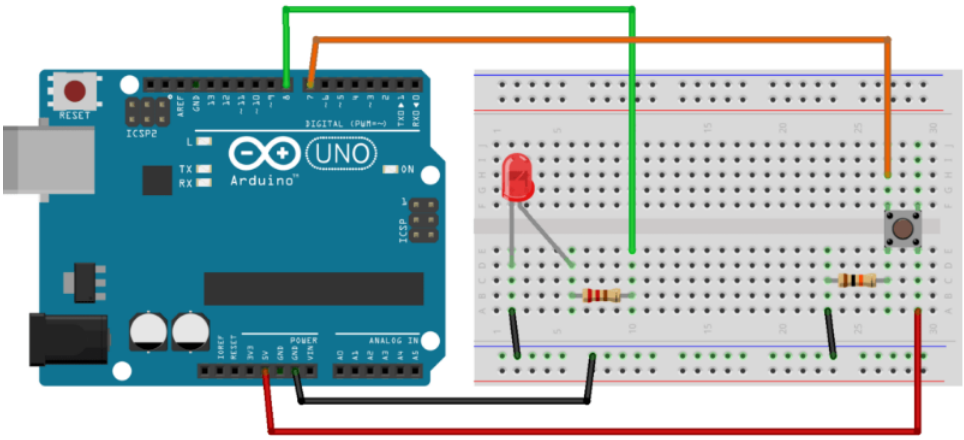


Figure 1 Arduino circuit diagram for led blink using external pull down resistor

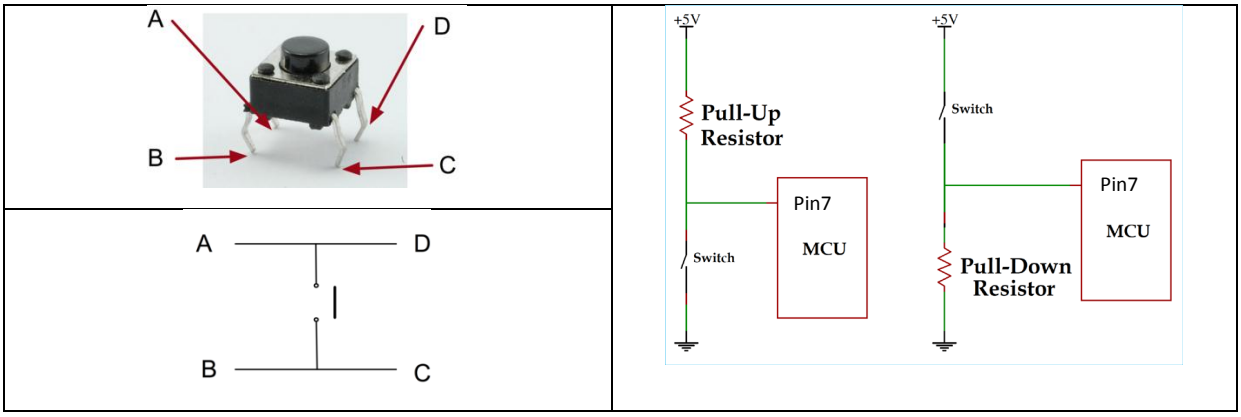


Figure 2 (a) Circuit diagram Schematic of Push Button Switch (b) Arduino (MCU) with external pull down resistor (https://roboticsbackend.com/arduino-input_pullup-pinmode/)

Procedure

- First, make sure to power off your Arduino – remove any USB cable.
- Plug a black wire between the blue line of the breadboard and a ground (GND) pin on the Arduino board.
- Plug the LED. You can notice that the LED has a leg shorter than the other. Plug this shorter leg to the ground (blue line here) of the circuit.
- Connect the longer leg of the LED to a digital pin (here pin no 8, you can change it). Add a 330 Ohm resistor in between to limit the current going through the LED.
- Add the push button to the breadboard, like in the picture.
- Connect one leg of the button to the ground, and put a 10k Ohm resistor in between. This resistor will act as a "pull down" resistor, which means that the default button's state will be LOW.
- Add a red wire between another leg of the button and VCC (5V).
- Finally, connect a leg of the button (same side as the pull down resistor) to a digital pin (here pin no 7, you can change it).

Arduino Code

After circuit connection, open the Arduino IDE software on your computer. Open the new sketch File by clicking File- New. And type the following code for turn on the LED when button is pressed, turn it off otherwise.

```
#define LED_PIN 8
#define BUTTON_PIN 7
void setup() {
pinMode(LED_PIN, OUTPUT);
pinMode(BUTTON_PIN, INPUT);
}
void loop() {
if (digitalRead(BUTTON_PIN) == HIGH) {
digitalWrite(LED_PIN, HIGH);
}
else {
digitalWrite(LED_PIN, LOW);
}
}
```

Results & Observation

Experiment 5

Aim: Coding to increase and decrease the intensity of brightness the LED blinks

Components: List of components required to build the circuit:

- 1 × Arduino board
- 1 × Breadboard
- 1 × LED – any color
- 1 × Any resistor from 330 to 1k Ohm
- 1 × Potentiometer - 10k Ohm
- 1 × Jumper wire

Theory

In this project, we will control LED brightness using the Potentiometer. This project concept is based on the "Arduino AnalogRead using the Potentiometer" and "Digital pin PWM output Arduino LED Fading", which is explained in the previous tutorials. Here we will read analog output from the potentiometer using Arduino and control the LED using PWM output. Also, use the map( ) function in the Arduino code.

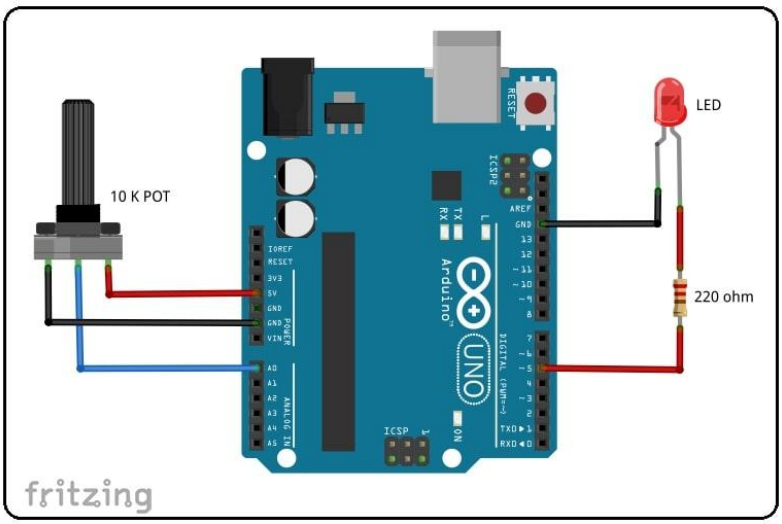


Figure 1 Circuit Diagram for LED Brightness Control using Potentiometer (https://www.electroduino.com/led-brightness-control-using-potentiometer/)

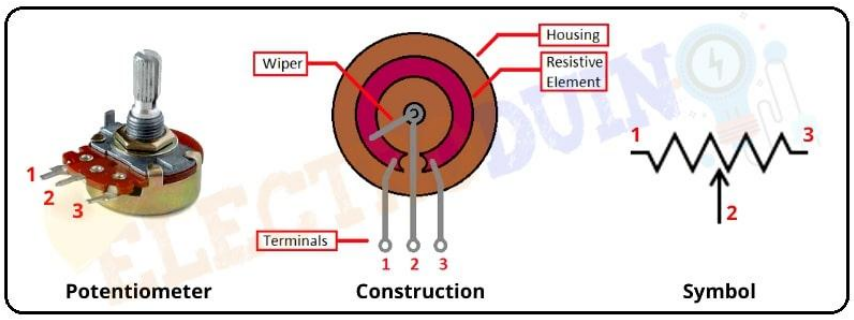


Figure 2 Potentiometer internal construction and symbol

Procedure

- First of all, we need to connect the Potentiometer terminal 1 is connected to +5v Vcc, and terminal 3 is connected to the ground.
- Now we can read the output from terminal 2 of the potentiometer, and we need to connect this terminal to an analog pin of the Arduino board.
- Arduino analog pin has 10-bit Analog to Digital Converter(ADC), which converts the potentiometer output voltage into integer value between the range of 0 to 1023 volts (2^10 = 1024) as input. Where the value "0" represents "0 volts" and the value "1023" represents "5 volts". When we will rotate the potentiometer knob, then the output voltage is change and the Arduino reads this changing of output voltage as the input voltage.
- Another side, the positive terminal of the LED connected to the PWM (Pulse Width Modulation) pin. This pin provides analog results with digital means at the range of 0 to 255. Where the value "0" represents "0 volts" and 255 represents "5 volts".
- The LED brightness changes Low to High according to the change of PWM output value 0 to 255.

Programming

After circuit connection, open the Arduino IDE software on your computer. Then open the new sketch File by clicking File -New. And type the following code:

```
// These constants won't change. They're used to give names to the pins used:
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)
```

```
void setup() {
// initialize serial communications at 9600 bps:
  Serial.begin(9600);
}
void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(2);
```

Results & Observation

Experiment 6

Aim: Coding to blink multiple LEDs at the same time

Components: List of components required to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno R3
- 3 × LED - any color
- 3 × 330Ω Resistor
- 3 × Jumper wire

Circuit Diagram

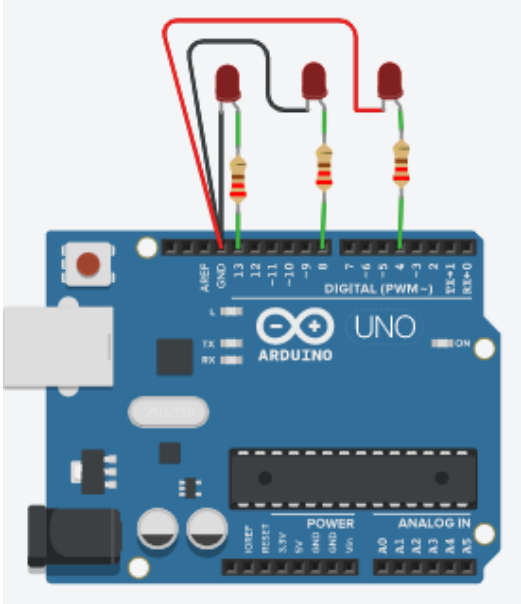


Figure 1 Circuit Diagram for blinking multiple LED at same time

Procedure

In this +ve pin (longer pin) of all led are connected to pin 4, 8 and 13 through resistor (330Ω) and shorter pin is connected to GND as shown in Figure 1.

Programming

After circuit connection, open the Arduino IDE software on your computer. Then open the new sketch File by clicking File -New. And type the following code:

```
// initialize the pins and tell which pin are connected to which LED (int led no = arduino pin no)
int led1 = 4;          // or #define LED_PIN 4
```

```
int led2 = 8;          // or #define LED_PIN 8
int led3 = 13;         // or #define LED_PIN 13
// Setup the 8 pins as OUTPUT
void setup() {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led1, HIGH);   // turn the LED on (HIGH is the voltage level at 5 volt)
digitalWrite(led2, HIGH);   // turn the LED on (HIGH is the voltage level at 5 volt)
digitalWrite(led3, HIGH);   // turn the LED on (HIGH is the voltage level at 5 volt)
delay(100);                 // wait for a second
digitalWrite(led1, LOW);   // turn the LED on (LOW  is the voltage level at 0 volt)
digitalWrite(led2, LOW);   // turn the LED on (LOW  is the voltage level at 0 volt)
digitalWrite(led3, LOW);   // turn the LED on (LOW  is the voltage level at 0 volt)
delay(100);                // wait for a second
}
```

Results & Observation

Experiment 7

Aim: Coding for RGB LED to scroll through a variety of colors

Components: List of components required to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno R3
- 3 × LED – Red, Blue and Green (RGB) color
- 3 × 330Ω Resistor
- 4 × Jumper wire

Circuit Diagram

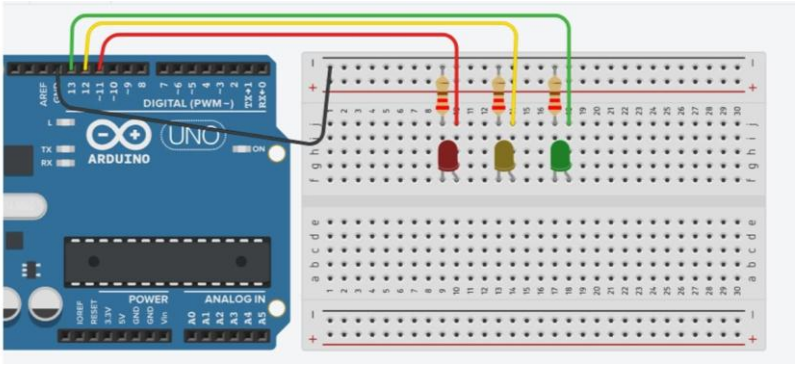


Figure 1 Circuit Diagram for scrolling RGB LEDs

Procedure

In this +ve pin (longer pin) of RGB led are connected to pin 11,12 and 13 through resistor (330Ω) and shorter pin is connected to GND as shown in Figure 1.

Programming

Open the Arduino program and upload the following code to your Arduino board. This code is the very basic of controlling LEDs.

```
// Code 1
// initialize the pins and tell which pin are connected to which LED (int led no = arduino pin no)
int led1 = 11;        // or #define LED_PIN 11
int led2 = 12;        // or #define LED_PIN 12
int led3 = 13;        // or #define LED_PIN 13
// Setup the 8 pins as OUTPUT
void setup() {
```

```
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led1, HIGH);   // turn the LED on (HIGH is the voltage level at 5 volt)
delay(100);                 // wait for a second
digitalWrite(led2, HIGH);   // turn the LED on (HIGH is the voltage level at 5 volt)
delay(100);                 // wait for a second
digitalWrite(led3, HIGH);   // turn the LED on (HIGH is the voltage level at 5 volt)
delay(100);                 // wait for a second
digitalWrite(led1, LOW);    // turn the LED on (LOW  is the voltage level at 0 volt)
delay(100);                 // wait for a second
digitalWrite(led2, LOW);    // turn the LED on (LOW  is the voltage level at 0 volt)
delay(100);                 // wait for a second
digitalWrite(led3, LOW);    // turn the LED on (LOW  is the voltage level at 0 volt)
delay(100);                 // wait for a second
}
```

Results & Observation

Experiment 8

Aim: Coding for piezo buzzer/speaker

Components: List of components required to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno
- 1 × Piezo buzzer
- 2 × Jumper wire

Circuit Diagram

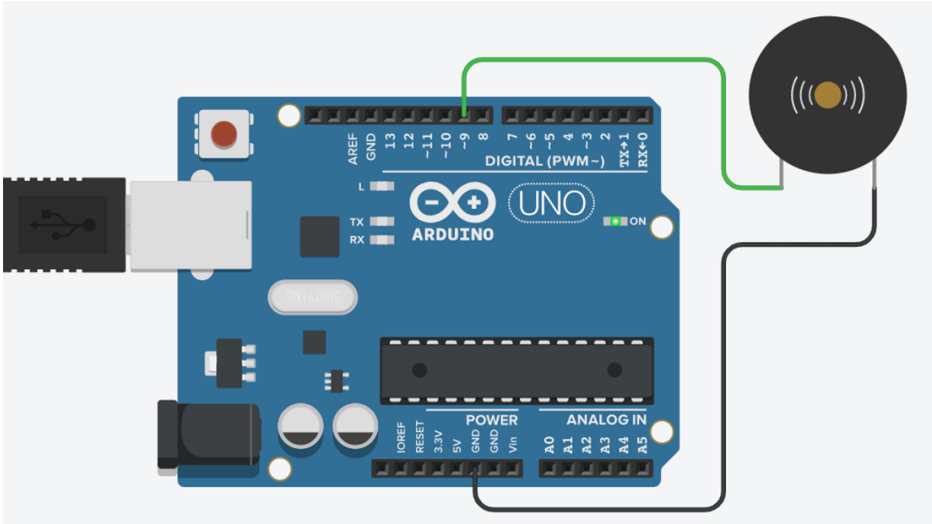


Figure 1 Circuit Diagram for Piezo buzzer

(https://www.electrovigyan.com/arduino/piezo-buzzer/)

Theory

A "piezo buzzer" is basically a tiny speaker that you can connect directly to an Arduino. "Piezoelectricity" is an effect where certain crystals will change shape when you apply electricity to them. By applying an electric signal at the right frequency, the crystal can make sound.

Procedure

The +ve pin (longer pin) of all led are connected to pin 4, 8 and 13 through resistor (330Ω) and shorter pin is connected to GND as shown in Figure 1.

Programming

After circuit connection, open the Arduino IDE software on your computer. Then open the new sketch File by clicking File -New. And type the following code:

```
int buzzer=3;
void setup() {
  // put your setup code here, to run once:
pinMode(buzzer,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
tone(buzzer,12000);
}
```

Results & Observation

Experiment 9

Aim: Coding of a temperature/fire sensor

Components: List of components required to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno
- 1 × LM35
- 3 × Jumper wire

Theory

LM35 is a precision temperature sensor with its output proportional to the temperature (in °C). With LM35, temperature can be measured more accurately than with a thermistor. It also possess low self-heating and does not cause more than 0.1°C temperature rise in still air.

The operating temperature range is from -55°C to 150°C. The output voltage varies by 10mV in response to ambient temperature; its scale factor is 0.01V/ °C or 10mv/ °C. LM35 can be easily interfaced with Arduino to measure and monitor the temperature change.

Circuit Diagram

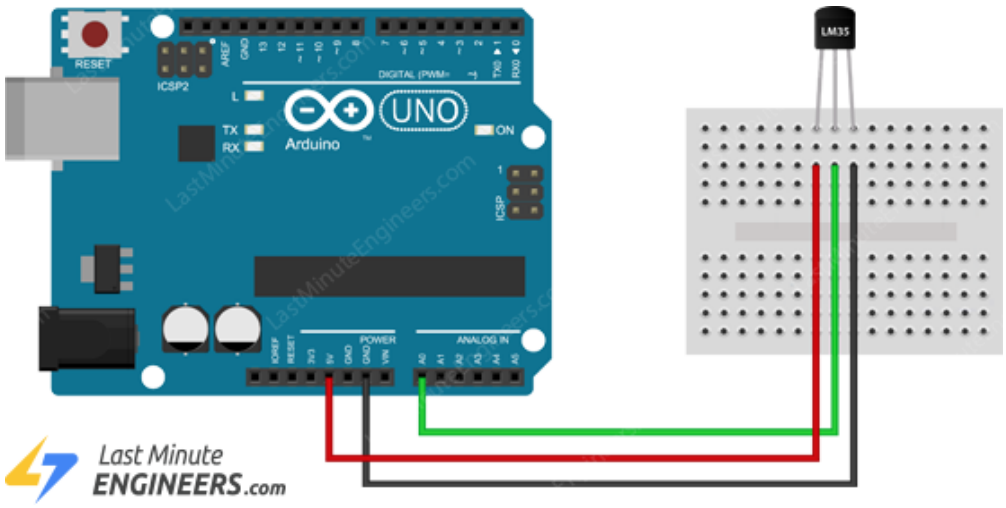


Figure 1 Circuit Diagram for temperature sensor using LM35

Procedure

- First, place the LM35 anywhere horizontally on your breadboard, the flat side of the

sensor must be facing you. Then, connect three wires under the three pins of the sensor.

- The wire on the left will go to the 5v (+5 volts) on the Arduino.
- The middle wire will go to A0 (analog pin 1).
- The wire on the right will go to GND (-) on the Arduino.
- Upload the code and open the serial monitor as readings of the temperature is shown. Make sure that the serial monitor is on 9600 bauds.

Programming

```
int val;
int tempPin = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  val = analogRead(tempPin);
  float mv = ( val/1024.0)*5000;
  float cel = mv/10;
  float farh = (cel*9)/5 + 32;
  Serial.print("TEMPRATURE = ");
  Serial.print(cel);
  Serial.print("*C");
  Serial.println();
  delay(1000);
/* uncomment this to get temperature in farenhite
Serial.print("TEMPRATURE = ");
Serial.print(farh);
Serial.print("*F");
Serial.println();
*/
}
```

Results & Observation

Experiment 10

Aim: Coding of a Infra-red obstacle sensor

Components: List of components required to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno R3
- 1 × Infrared IR Sensor
- 1 × LED – any color
- 1 × 330Ω Resistor
- 5 × Jumper wire

Theory

Infrared IR Sensor Obstacle Avoidance Sensor board is an inexpensive solution to avoidance detection for robotics, smart car, and other electronic uses. The IR transmitter sends an infrared signal that, in case of a reflecting surface (e.g. white color), bounces off in some directions including that of the IR receiver that captures the signal detecting the object. The module has 3 pins — Vcc, GND, and the output.

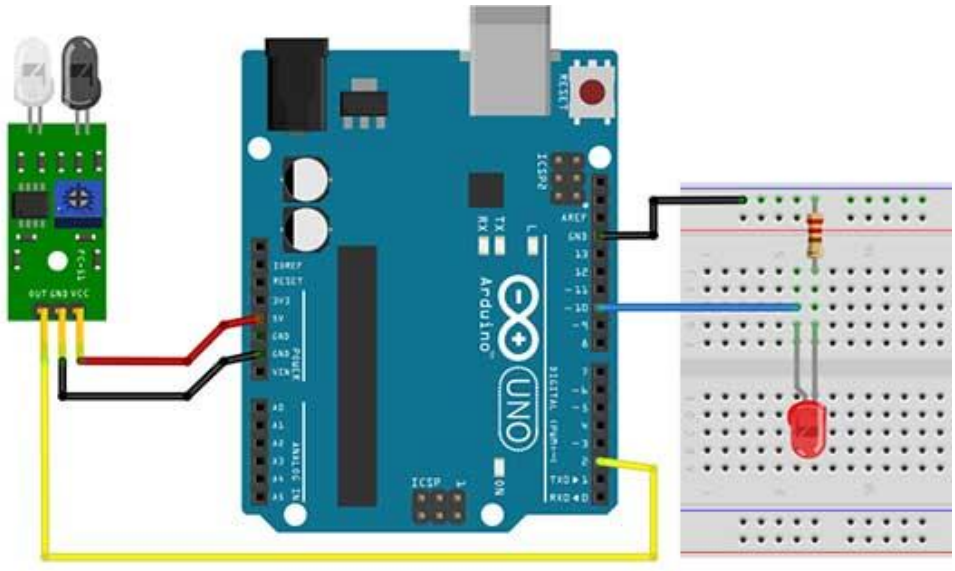


Figure 1 Circuit Diagram for Obstacle Avoidance Sensor using Infrared IR Sensor

Procedure

- Connect the Vcc of the Sensor Module to the Vcc of Arduino Board
- Connect GND of the Sensor Module to the GND of the Arduino Board.

- Connect the output pin of the sensor module to pin 2 of the Arduino Board.
- When the Obstacle Avoidance Sensor detects an obstacle, the LED will be on (connected to Pin 10). Otherwise, it will be off.

Programming

After circuit is completed, connect the Arduino board to computer using the USB cable. The green power LED (labelled PWR) should go on. Open the Arduino IDE and choose corresponding board type and port type (COM). Then load up the following sketch onto your Arduino.

```
//Switch an LED using an IR sensor
        int ir = 2;
        int PinLed = 10;
        int val=0;

        void setup()
        {
                pinMode(ir, INPUT);
                pinMode(PinLed, OUTPUT);
        }

        void loop()
        {
                if(digitalRead(ir) == LOW)
                {
                        digitalWrite(PinLed, HIGH);
                }
                else
                {
                        digitalWrite(PinLed, LOW);
                }
        }
```

Results & Observation

Experiment 12

Aim: Coding for servo back and forth range of motion

Components: List of components required to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno
- 1 × Servo motor (SG 90-2.5kg/cm)
- 5 × Jumper wire

Theory

A servomotor (or servo motor) is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity, and acceleration.

Diagram Circuit

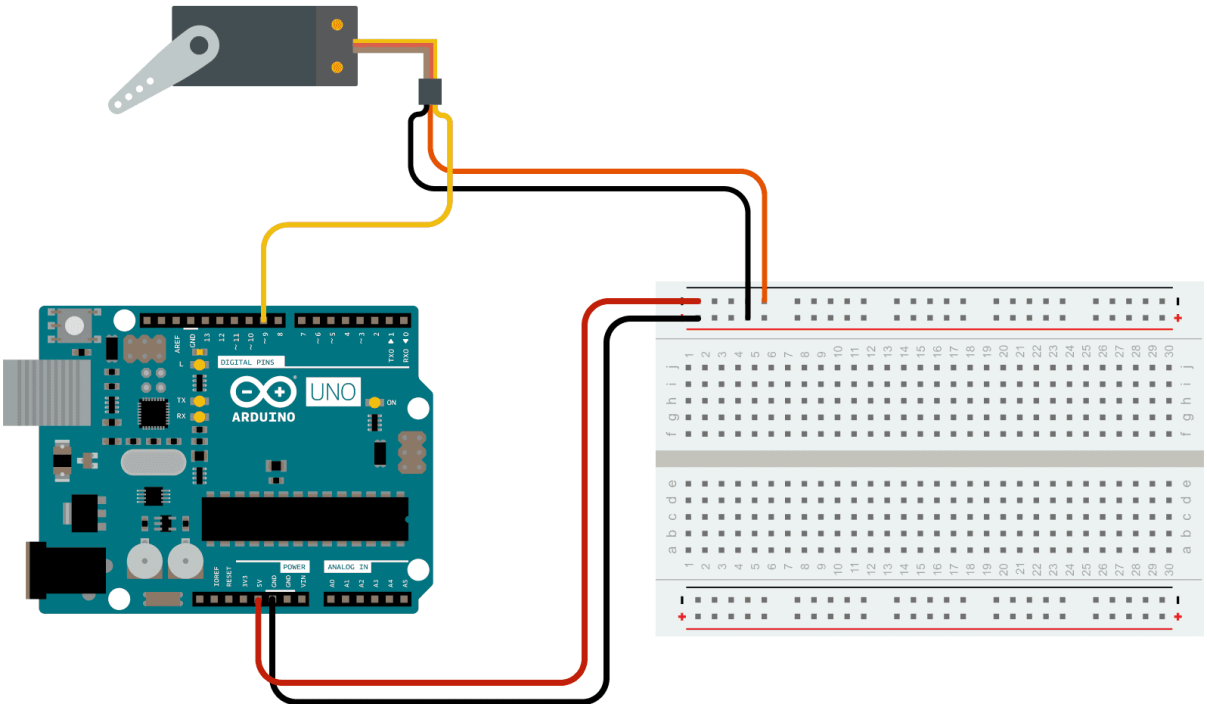


Figure 1 Circuit diagram for servo back and forth range of motion

Procedure

- For this, connect the servo motor to +5V, GND and pin 9 as shown in Figure 1.

Code

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position
void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

Results & Observation

Experiment 13

Aim: Coding for DC stepper / servo motor control

Components: List of components to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno
- 1 × Servo motor (SG 90-2.5kg/cm)
- 1 × Potentiometer – any value
- 6 × Jumper wire

Theory

A servo motor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity, and acceleration. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a servo mechanism (Figure 1).

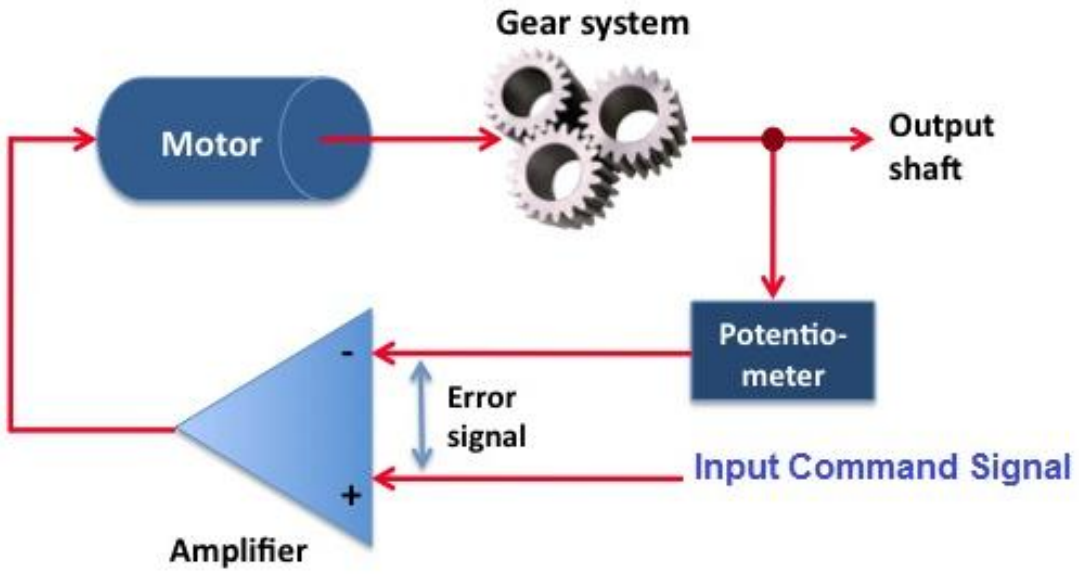


Figure 1 Servo motor and its mechanism

Circuit Diagram

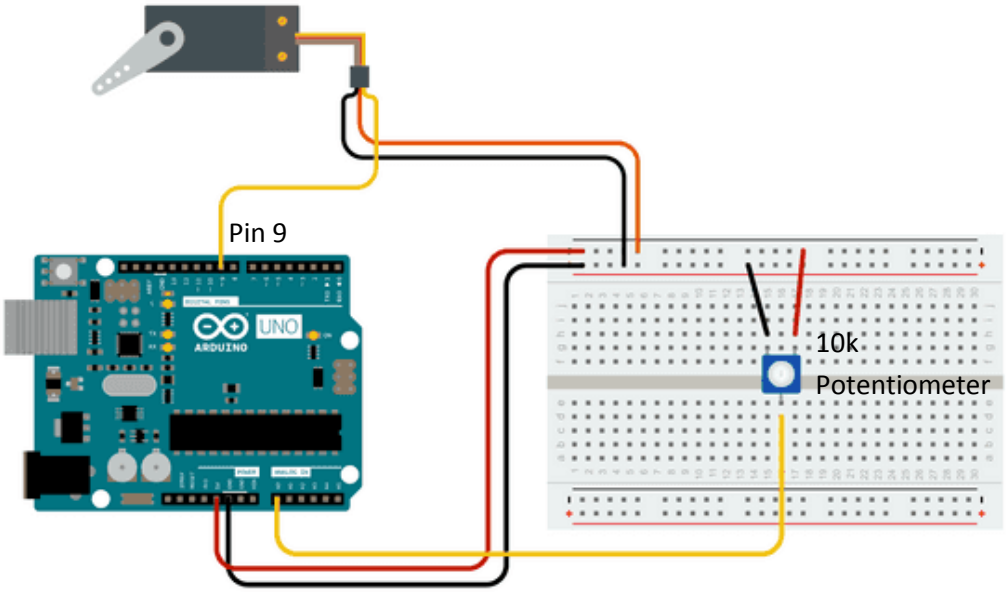


Figure 2 Circuit diagram for servo motor control with Arduino

Procedure

- First wire the potentiometer so that its two outer pins are connected to power (+5V) and ground, and its middle pin is connected to A0 on the board.
- Then, connect the servo motor to +5V, GND and pin 9.

Code

After circuit connection, open the Arduino IDE software on your computer. Then open the new sketch File by clicking File -New. And type the following code

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin
void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop() {
```

```
  val = analogRead(potpin);            // reads the value of the potentiometer (value between 0
and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and
180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```

Results and Observation

Experiment 15

Aim: Coding for display text / names / number / time on the LCD screen

Components: List of components required to build the circuit:

- 1 × Breadboard
- 1 × Arduino Uno
- 1 × Potentiometer – 10kΩ
- 1 × LCD 16×2
- 16 × Jumper wire

Theory

In LCD 16×2, the term LCD stands for Liquid Crystal Display that uses a plane panel display technology, used in screens of computer monitors & TVs, smartphones, tablets, mobile devices, etc. Both the displays like LCD & CRTs look the same but their operation is different. Instead of electrons diffraction at a glass display, a liquid crystal display has a backlight that provides light to each pixel that is arranged in a rectangular network.

An electronic device that is used to display data and the message is known as LCD 16×2. As the name suggests, it includes 16 Columns & 2 Rows so it can display 32 characters (16×2=32) in total & every character will be made with 5×8 (40) Pixel Dots. So the total pixels within this LCD can be calculated as 32 x 40 otherwise 1280 pixels.

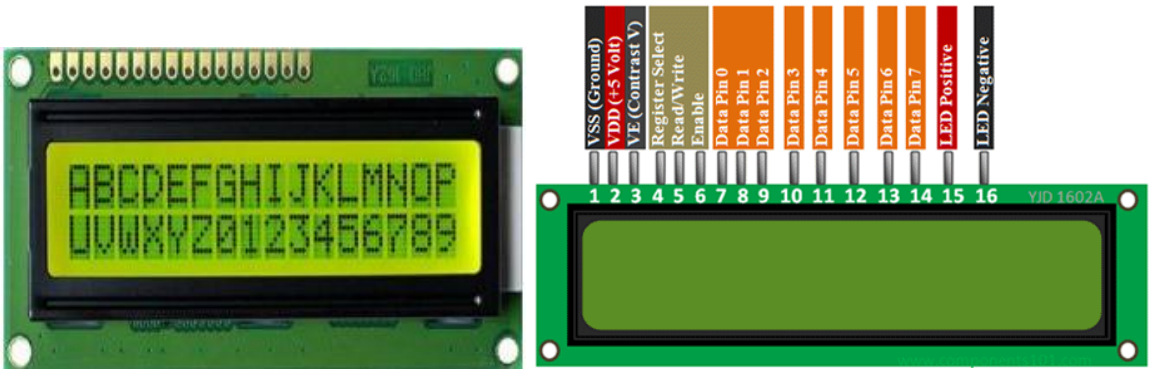


Figure 1 LCD display characters print and pins specification

Circuit Diagram

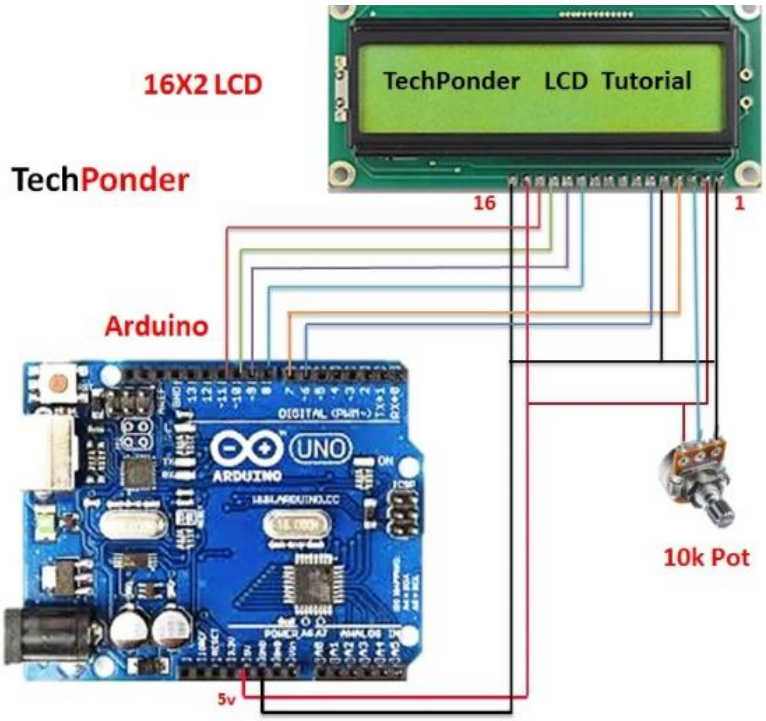


Figure 2 Circuit diagram for LCD display

Procedure

- First connect all the pins of the LCD with breadboard and then connect with Arduino board through jumper wire as per circuit diagram shown in Figure 2.
- Then, connect potentiometer (10kΩ) as shown in Figure 2.

Code

Copy and Paste below code in Arduino New sketch. Compile and Upload it to Arduino Uno.

```
// include the library code:
#include<LiquidCystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 6, 8, 9, 10, 11);
void setup()
{
// set up the LCD's number of columns and rows:
lcd.begin(16, 2); // Print a message to the LCD.
lcd.print("TechPonder"); }
void loop()
```

```
{ // set the cursor to column 0, line 1 // (note: line 1 is the second row, since counting begins
with 0): lcd.setCursor(0, 0);
lcd.print(" TechPonder ");
lcd.setCursor(0,1);
lcd.print(" Arduino LCD ");
}
```

Results & Observation