ARDUINO UNO EDUCATIONAL KIT

ABRA

www.abra-electronics.com

ARD-EDU-KIT

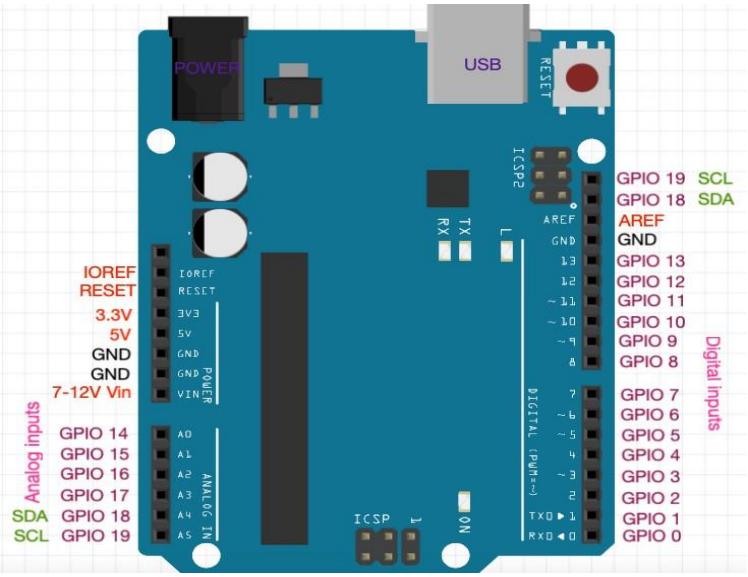
TEL: 1-800-361-5237

S.no	Name of the Component	Part number	Quantity
1	Arduino UNO Board	ABRAUNO	1
2	5V Relay	SRD-5VDC-SL-C	1
3	Remote Control with IR Receiver	WL-110	1
4	Servo Motor SG90	SG90	1
5	16x2 LCD Module with I2C	LCD-MOD-13	1
6	Humidity Sensor Module	SENS-DHT11-BB	1
7	PS2 Joystick Module	AM-JOYSTICK	1
8	Power Supply Module	PSM-297	1
9	Stepper Motor with Driver	MOT-28BYJ48	1
10	Passive Buzzer	BUZ-125	1
11	Active Buzzer	BUZ-120	1
12	Breadboard	ABRA-12-LC	1
13	1 Digit RED 7- Segment Display	S-5101AS	1
14	Tilt Sensor Module	SENS-39	1
15	Ultrasonic Sensor Module	HC-SR04	1
16	4x3 Keypad	419-ADA	1
17	PIR Sensor	SENS-PIR	1
18	Water Level Sensor Module	SENS-78	1
19	RTC Module	ARD-DS3231	1
20	Sound Sensor Module	SENS-42	1
21	Small Button Switch	PBS-315BK	5
22	Photoresistor	PHOTO-300	2
23	NPN Transistor	S8050	5
24	NPN Transistor	PN2222A	5
25	1N4007 Diode	1N4007	1
26	Resistors (10,100,220,330,1K,2K,5K,10K,100K,1M)		100
27	10K Thermistor	334-103	1
28	5mm LED (Red, Green, Yellow, Blue, White)		25
29	RGB LED	LED-5RGB-4-CC	1
30	Motor Driver	L293D	1
31	Shift Register	74HC595	1
32	Capacitor (10uf 50V)	10R50	2
33	Capacitor (100uf 50V)	100R50	2
34	Ceramic Capacitor(22pf)	CD220	5
35	Ceramic Capacitor (104)	CD104	5
36	DC Motor FAN	MOT-PROP-L	1
37	DC Motor	MOT-500	1
38	Potentiometer	P10K-MIN-PC	2
39	Rotary Encoder	AM-127	1
40	Female to Male	JW-MF-20-6	1
41	Male to Male	JW-MM-20-6	1
42	9V AC Adapter	PSC12R-090	1
43	Plastic Case	CB-13	1
44	Resistor card		1
45	Manual		1

Contents

1 Introduction to Arduino

Arduino Uno is a microcontroller board based on the ATmega328P.

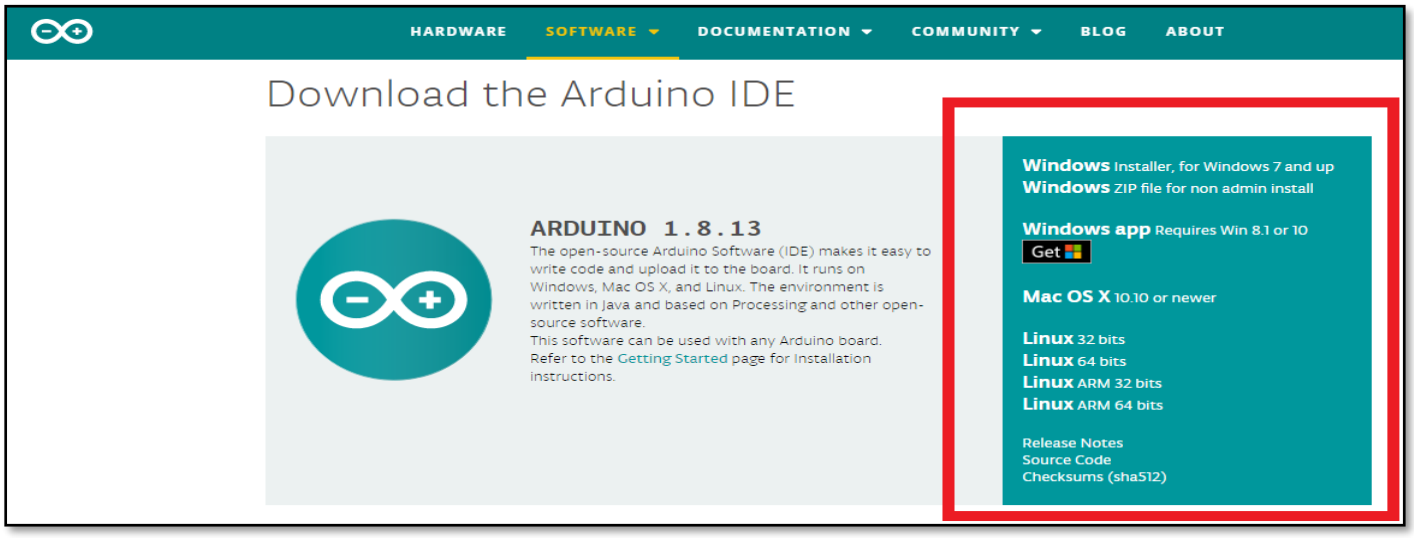


A0-A5: Analog data such as sensor photoresistor are read by these pins and Analog to digital converter (ADC) will convert the data into digital.

1.1 Installing Arduino

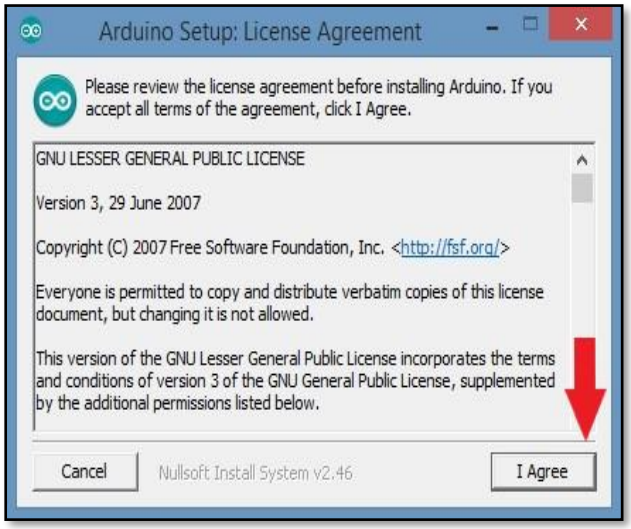
Step 1: Visit https://www.arduino.cc/. In software section select Downloads.

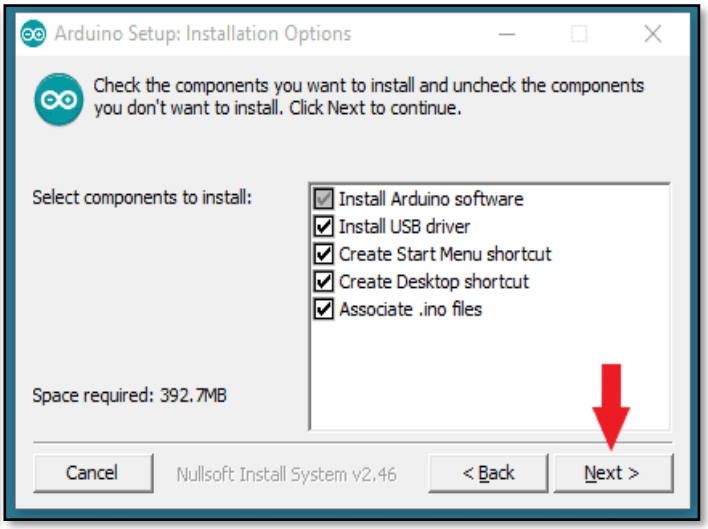
Step 2: Depending on the operating system in your computer select Arduino IDE accordingly. If windows click on the installer instead of ZIP file as it is easy to install.



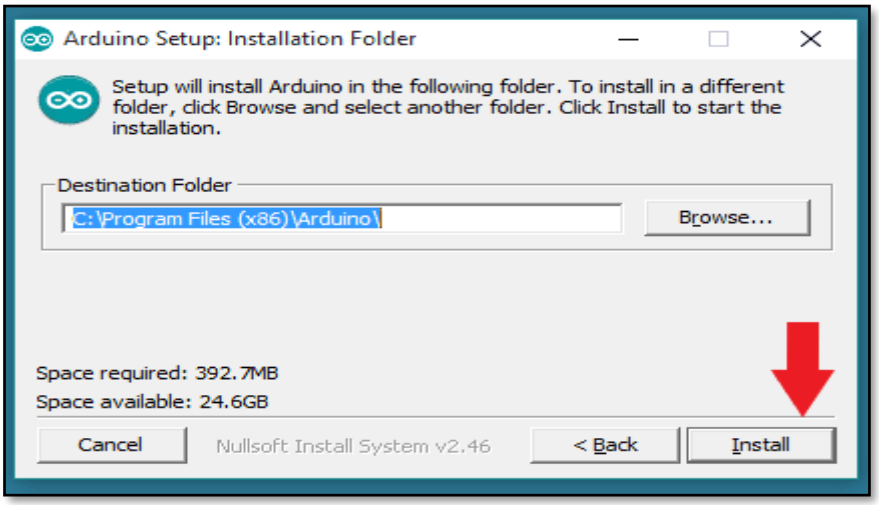
Step 3: Click **I agree**

Step 4: Click **Next**



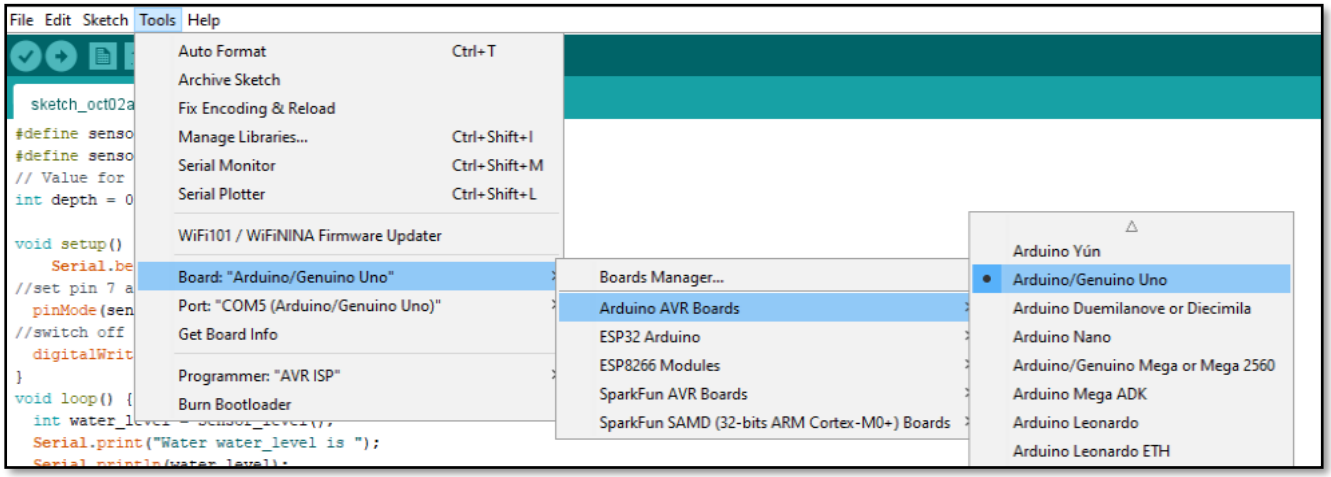


Step 5: Select the path where the Arduino IDE should be installed. By default it is C drive but you always have the option to browse and select your own path.

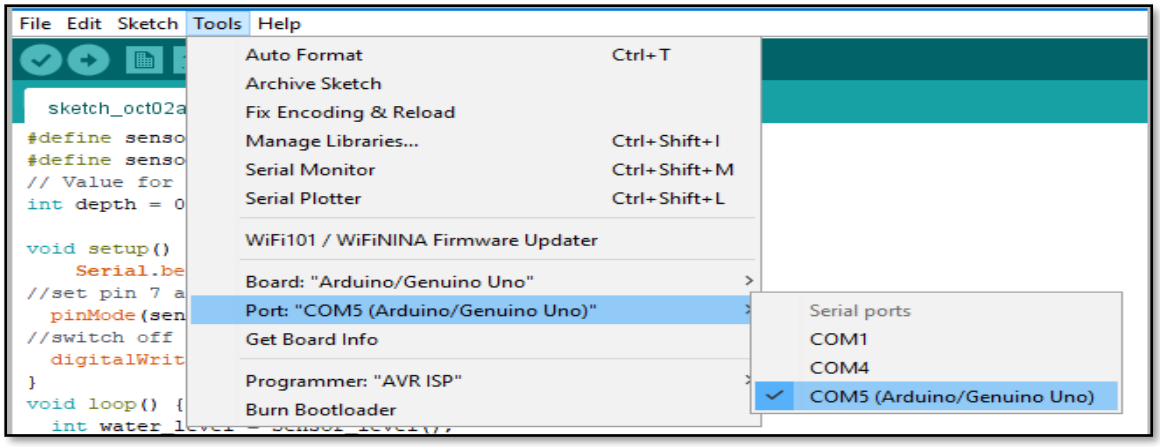


1.2 Getting started with IDE

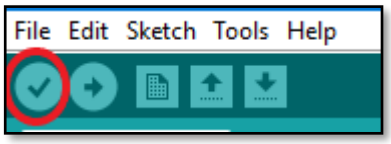
Open the IDE and connect your Arduino board to your computer. Before writing your code make sure you have selected the board. **Tools->Board->Arduino AVR boards-> Arduino/Genuino Uno**



Now select the port for the Arduino board. In this example its **Com 5 (Arduino/Genuino Uno).**



Now write your code and Compile and uploads the code to the board.

To compile your code click on the tick mark button shown in the 

figure

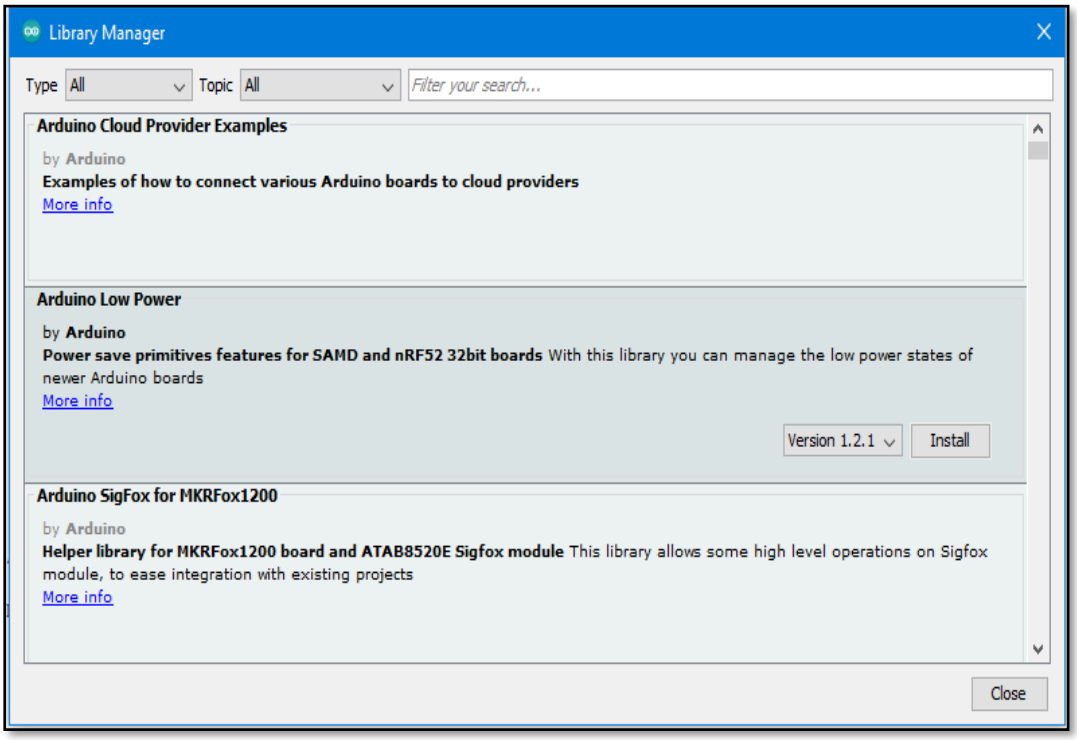
To upload the code select **Sketch-> upload**

1.3 Adding Libraries

The Arduino environment can be extended using libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from Sketch > Import Library.

How to Install a Library

- *Using the Library Manager:* To install a new library into your Arduino IDE you can use the Library Manager. Open the IDE and click to the "Sketch" menu and then ***Include Library > Manage Libraries.*** The search for the required library you are looking for and Click install.



- *Importing a .zip Library:* Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, leave it as is. In the Arduino IDE, navigate to **Sketch > Include**

Library > Add .ZIP Library. You will be prompted to select the library you would like to add. Navigate to the .zip file's location and open it.

1.4 Code Basics

Certain basic functions that makes coding in Arduino easier are listed below.

Digital I/O for digital pins

- digitalRead() : Reads the value from a specified digital pin, either HIGH or LOW.
  Syntax: digitalRead(pin)
- digitalWrite() : Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin.
  Syntax: digitalWrite (pin,value)

- pinMode() : Configures the specified pin to behave either as an input or an output.
  Syntax: pinMode(pin, mode).
  pin: the Arduino pin number to set the mode of. Mode is either INPUT or OUTPUT

```
int ledPin = 12;   // LED connected to digital pin 13
int inPin = 7;     // pushbutton connected to digital pin 7
int val = 0;       // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT);  // sets the digital pin 12 as output
  pinMode(inPin, INPUT);    // sets the digital pin 7 as input
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output

}
void loop() {
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);  // sets the LED to the button's value
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

Analog I/O for analog pins

- analogRead() : Reads the value from the specified analog pin
  Syntax: analogRead(pin)
  Example: digitalread(10); //reads the value either LOW or HIGH in pin number 10

- analogWrite() : Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds.

Syntax: analogWrite(pin, value)

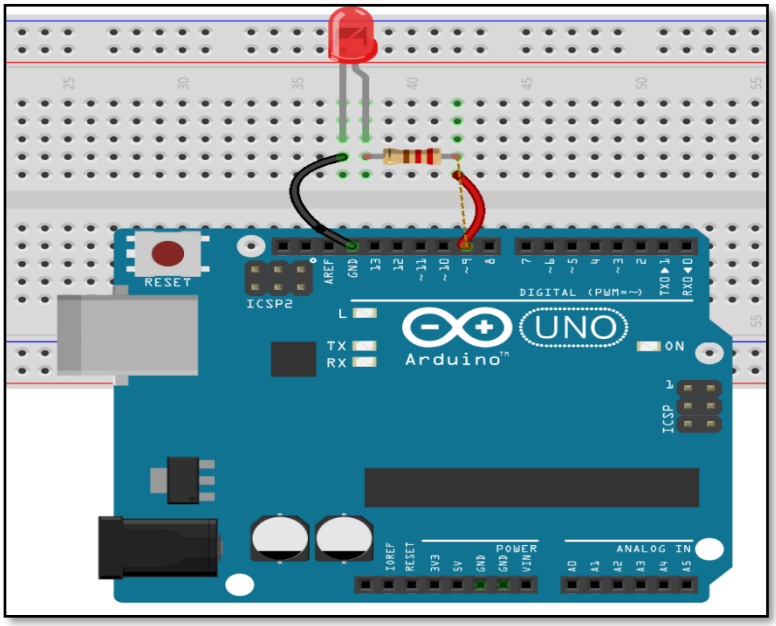
```
int ledPin = 9;       // LED connected to digital pin 9
int analogPin = 3;    // potentiometer connected to analog
pin 3
int val = 0;          // variable to store the read value

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
}
```

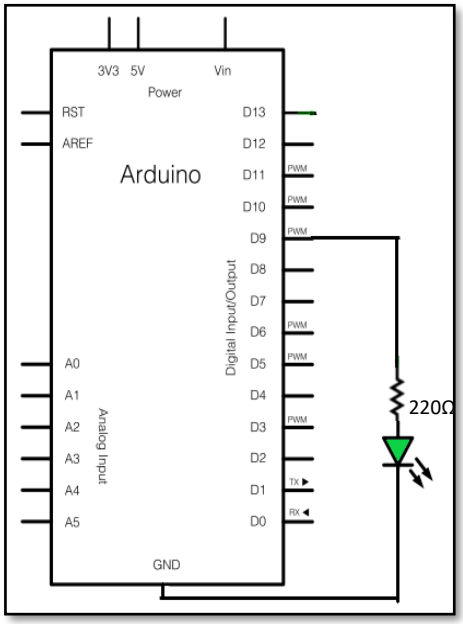
2 Projects

2.1 LED blink

Brief: Through this project we shall learn how to turn an LED on and off. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value for digitalWrite: 5V for HIGH, 0V (ground) for LOW.

Components: LED, RESISTOR 220Ω

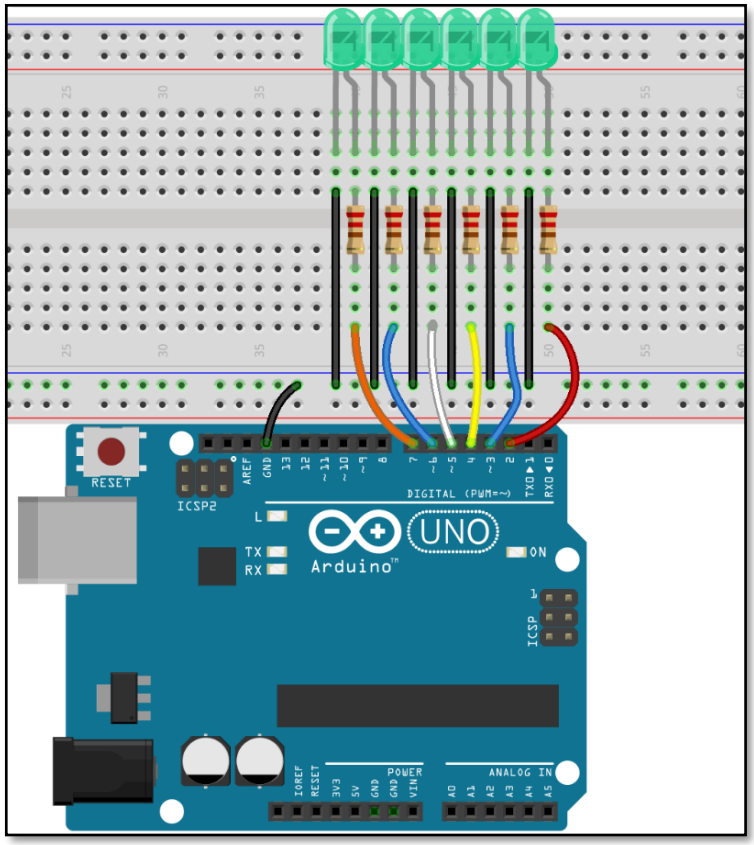


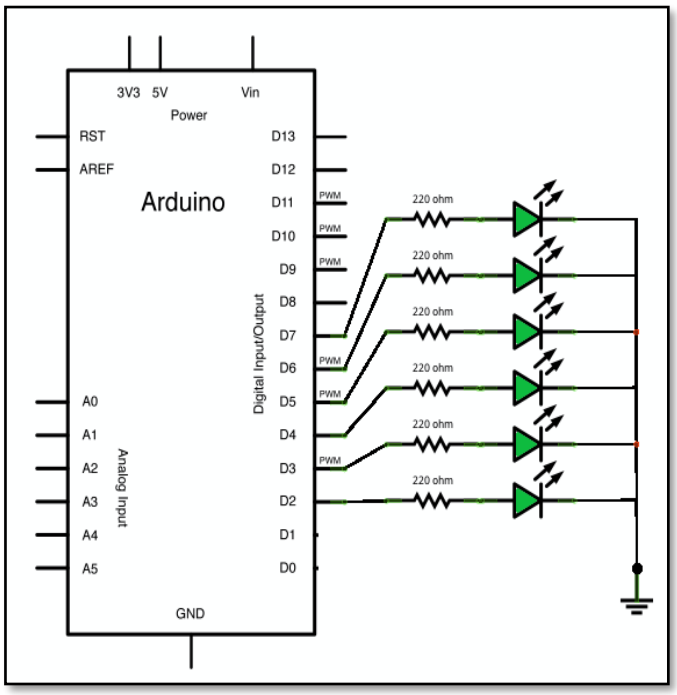


```
const int ledPin = 9;            //the Pin number of the on-board LED
void setup ()
{
pinMode (ledPin, OUTPUT);  //initialize the digital pin 9 as an output
}
void loop ()       //the loop routine runs repeatedly forever
{
digitalWrite(ledPin,HIGH);                  //turn the LED on
delay(500);                 //wait for half a second
digitalWrite(ledPin,LOW);                   //turn the LED off
delay(500);                  //wait for half a second
}
```

2.2 LED Trailing effect

Brief: Through this project we shall learn how to turn on the LED one by one using Arduino. We here use for loop to activate the LEDs connected to pin 2, 3,4,5,6,7 one after the other with a delay 500 i.e. half a second.
Components: LED, 220Ω resistors



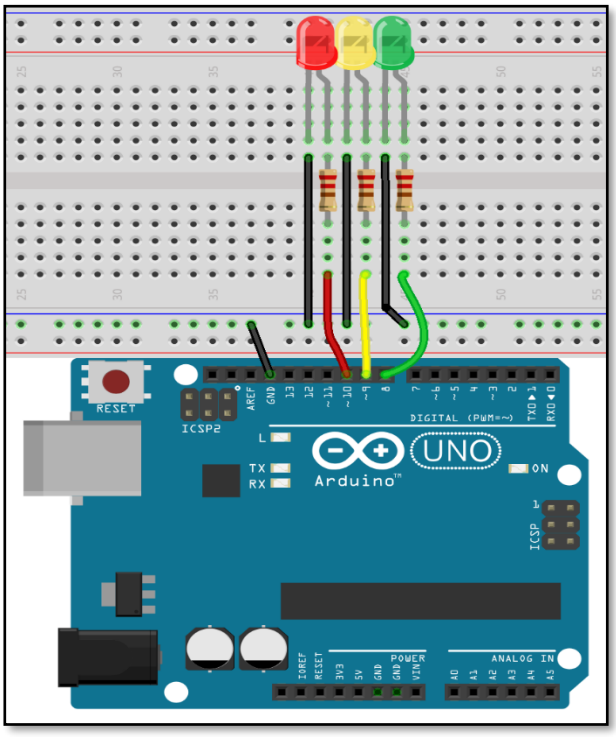


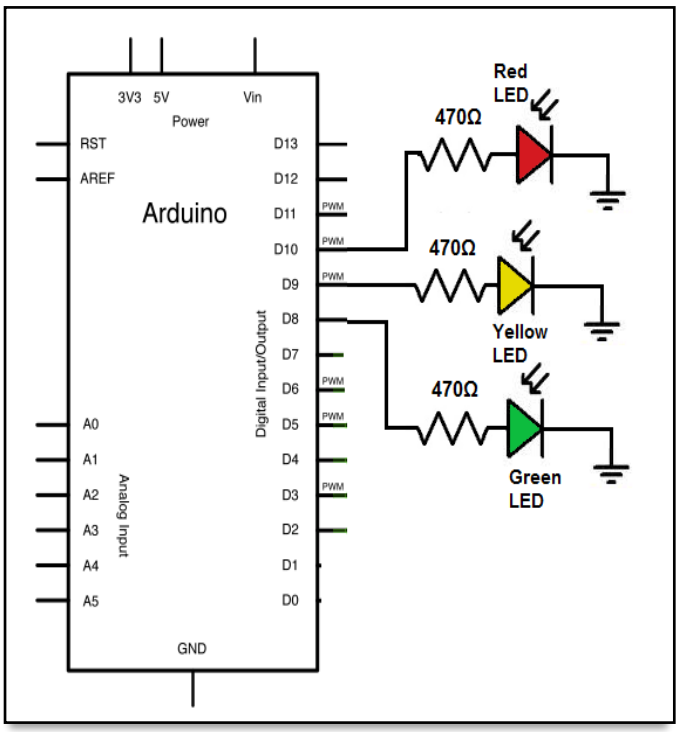
```
int timer=500;
void setup() {
  // initialize pin 2 to pin 7 as output pins
for (int number = 2; number < 8; number++) {
pinMode(number, OUTPUT);
  }
}
void loop() {
 // loop from the pin 2 to pin 7
for (int number = 2; number < 8; number++) {
digitalWrite(number, HIGH);   // turn on the pin
delay(timer);
digitalWrite(number, LOW); // turn off the pin
  }  // loop from pin 7 to pin 2
for (int number = 7; number >= 2; number--) {
digitalWrite(number, HIGH);    // turn on the pin
delay(timer);
digitalWrite(number, LOW);     // turn off the pin
  }
}
```

2.3 Traffic Light

Brief: Traffic light controller can be designed by giving definite amount of delay between switching of traffic light colors. In the code each light is turned on for one second while the other two colors are off and the cycle repeats with a delay if 1 second.

Components Required: LED (red, green, yellow), Resistor 220Ω





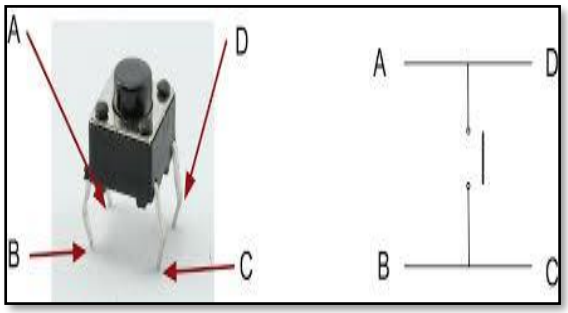
```
const int greenlight= 8; // assign greenlight to pin 8
const int yellowlight= 9; // assign greenlight to pin 9
const int redlight= 10; // assign greenlight to pin 10
void setup() {
pinMode (greenlight, OUTPUT);
pinMode (yellowlight, OUTPUT);
pinMode (redlight, OUTPUT);
}
void loop(){
digitalWrite (greenlight, HIGH); //turns greenlight on
digitalWrite (yellowlight, LOW); //turns yellowlight off
digitalWrite (redlight, LOW); //turns redlight off
delay(1000);
digitalWrite (greenlight, LOW); //turns greenlight  off
digitalWrite (yellowlight, HIGH); //turns yellowlight on
digitalWrite (redlight, LOW); //turns redlight off
delay(1000);
digitalWrite (greenlight, LOW);  //turns greenlight off
digitalWrite (yellowlight, LOW); //turns yellowlight off
digitalWrite (redlight, HIGH); //turns redlight on
delay(1000);
}
```

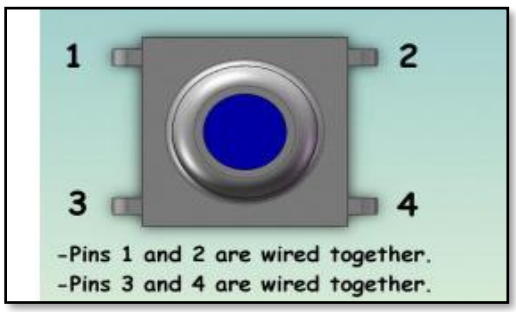
2.4 Controlling LED by Buttons

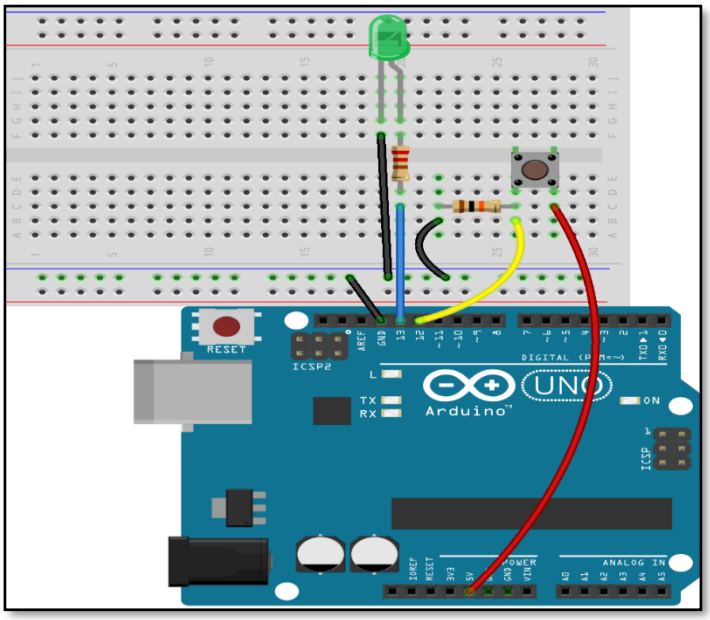
Brief: We use push button to ON and OFF LED. When button is pressed the circuit is closed which makes the LED to turn ON. In the code given below we read the value on the pin 12, if the button is pressed then the value at pin 12 will be HIGH then the LED is made to glow by setting the ledpin to HIGH.

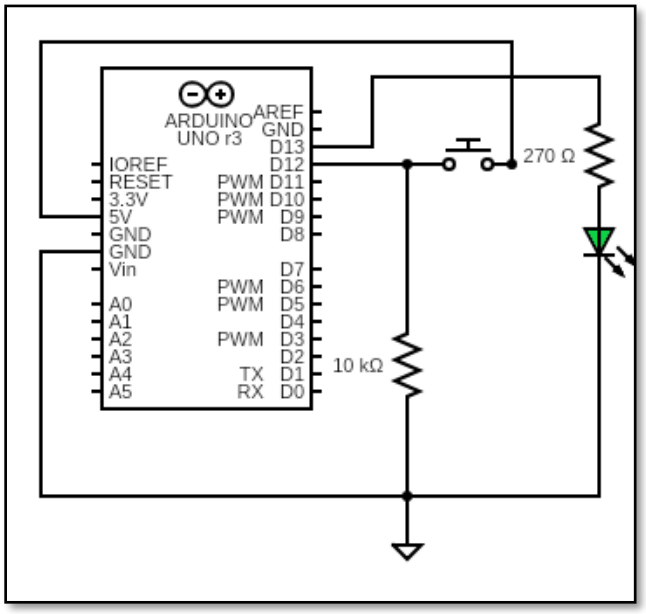
Components Required: Button switch, LED, Resistor 220Ω, 10KΩ, breadboard.
Push button works on the following principle

- Button not pressed = disconnected circuit
- Button pressed = connected circuit





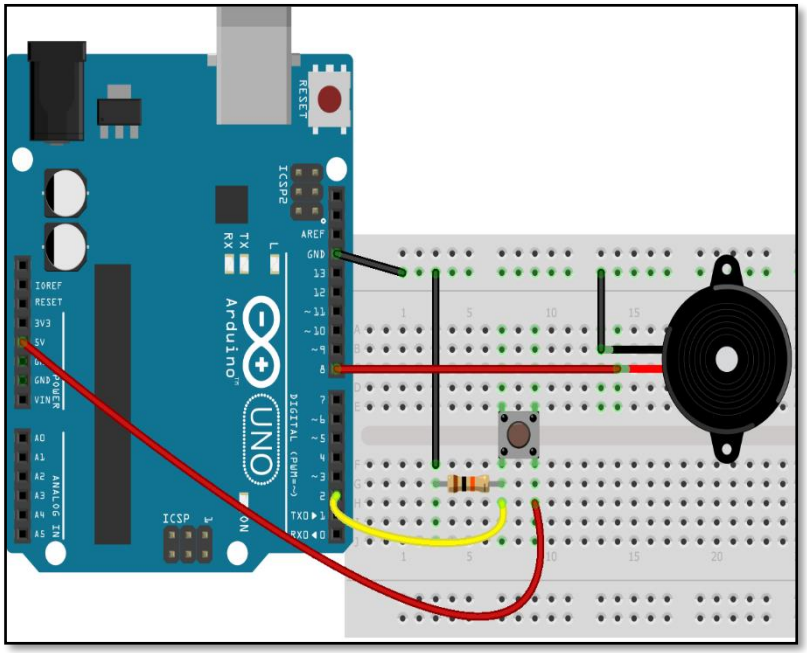


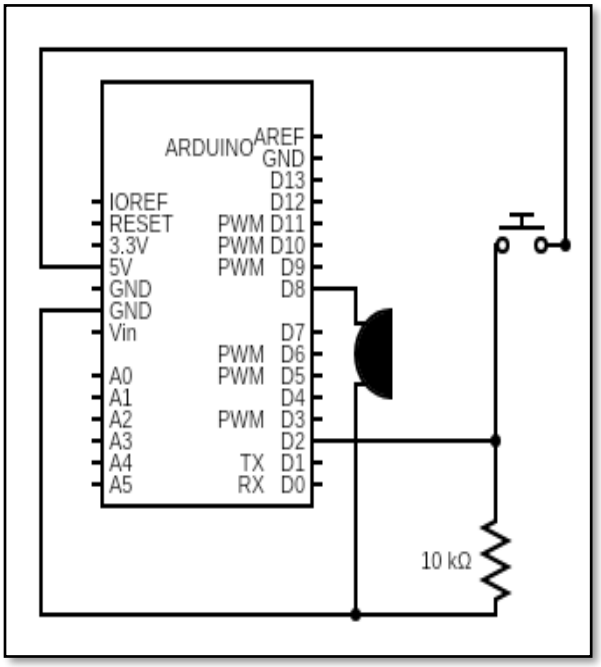



```
//LED is switched on and off using push button
// Written by ABRA ELECTRONICS
const int buttonPin = 12; //the button connected to pin 12
const int ledPin = 13; //the led connected to pin13
int buttonState = 0; // variable for reading the pushbutton status
void setup()
{
pinMode(buttonPin, INPUT); //initialize thebuttonPin as input
pinMode(ledPin, OUTPUT); //initialize the led pin as output
}
void loop ()
{   //read the state of the button value
buttonState = digitalRead (buttonPin);
 if (buttonState == HIGH)       // check if button is high (pressed)
          {
digitalWrite (ledPin, HIGH); //turn the led on
          }
else
          {
digitalWrite (ledPin, LOW); //turn the led off
          }
}
```

2.5 Doorbell using active buzzer

Brief: An **active buzzer** will generate a tone using an internal oscillator, so all that is needed is a DC voltage which we now provide through one of the digital pins on Arduino. Components: BUZ-120, Resistor 220Ω, Push button



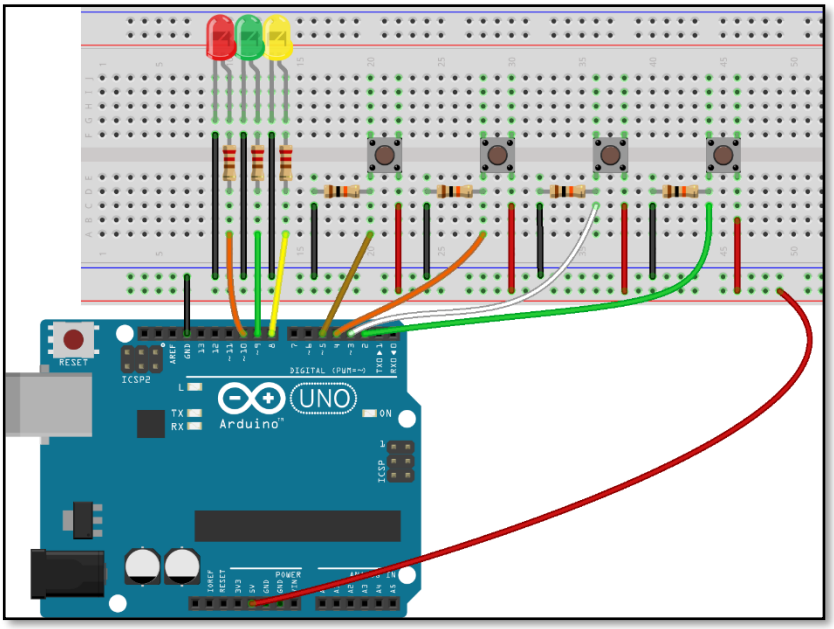


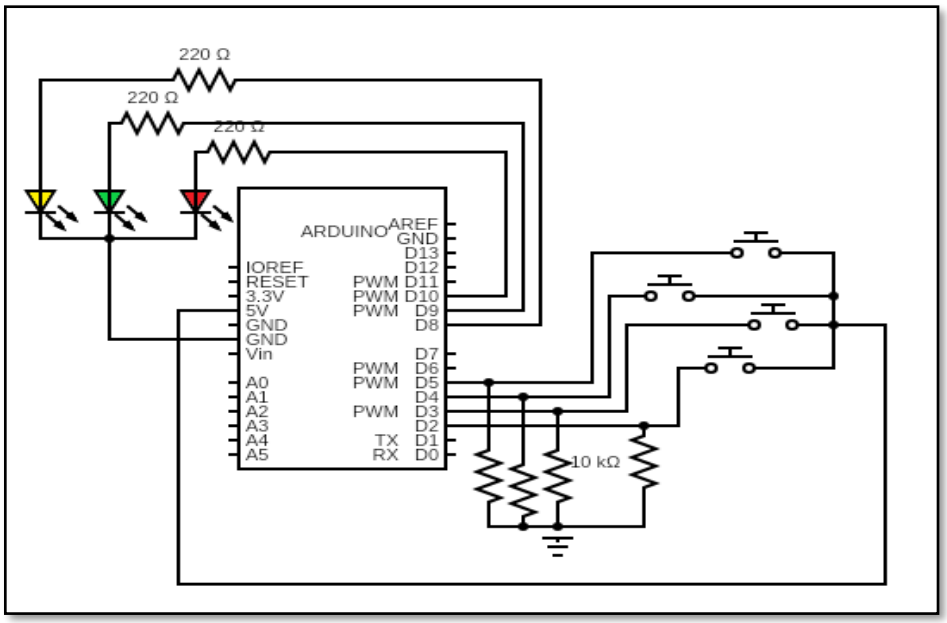

```
//Doorbell
//Turns buzzer on and off using the button.
// Written by ABRA ELECTRONICS
const int buttonPin = 2; //the button connect to pin2
const int buzzerPin = 8;  //the led connect to pin8
int buttonState = 0;  // variable for reading the pushbutton     status
void setup() {
pinMode(buttonPin, INPUT); //initialize the button pin as input
pinMode(buzzerPin, OUTPUT); //initialize the buzzer pin as output
      }
void loop() {
       //read the state of the button value
 buttonState = digitalRead(buttonPin);
             if (buttonState == HIGH ) { //and check if the button is pressed if it    is, the state will be HIGH
for (int i = 0; i < 25; i++) {
digitalWrite(buzzerPin, HIGH); //Activate the buzzer
delay(500);       //wait for half second
digitalWrite(buzzerPin, LOW); //Deactivate the buzzer
delay(500);       //wait for half second
         }
       }
     }
```

2.6 Responder Experiment

Brief: Through this experiment we learn how to control different colors of LED. We also use reset button to reset (clear all LED's to off state).

Components: Button switch (4), Red LED, Yellow LED, Green LED, Resistors 220Ω (3), 10KΩ (4)


RESET
ICSP2
UNO
Arduino
TX
RX
ON
ICSP
POWER
DIGITAL (PWM~)

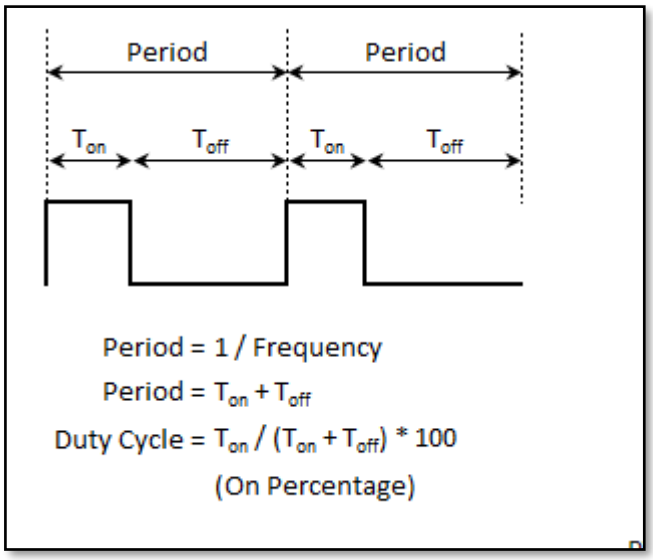

220 Ω
220 Ω
220 Ω
ARDUINO
AREF
GND
D13
D12
IOREF
RESET
3.3V
5V
GND
GND
Vin
PWM D11
PWM D10
PWM D9
D8
D7
PWM D6
PWM D5
D4
PWM D3
D2
TX D1
RX D0
A0
A1
A2
A3
A4
A5
10 kΩ

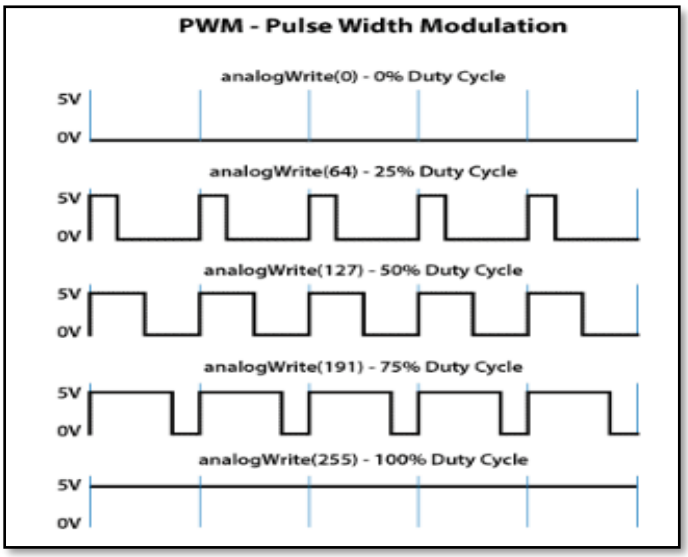
```
int redlight=10;    //assign red LED to pin 10
int yellowlight =8;  //assign yellow LED to pin 8
int greenlight =9;   //assign green LED to pin 9
int redbutton=5;    //assign red button to pin 5
int yellowbutton=4;  //assign yellow button  to pin 4
int greenbutton=3;   //assign green button to pin 3
int reset=2;   //assign reset button to pin 2
int red, yellow, green;
void setup() {
pinMode(redlight,OUTPUT); //set pin 10 (redlight) as output
pinMode(yellowlight ,OUTPUT); //set pin 8 (yellowlight) as output
pinMode(greenlight ,OUTPUT); //set pin 9 (greenlight) as output
pinMode(redbutton,INPUT);
pinMode(yellowbutton,INPUT);
pinMode(greenbutton,INPUT);
}
void loop()  {
red=digitalRead(redbutton); //read button value at pin 5
yellow=digitalRead(yellowbutton); //read button value at pin 4
green=digitalRead(greenbutton); //read button value at pin 3
if(red==HIGH)  OnRed(); //check if red button pressed then call OnRed function
if(yellow==HIGH) Onyellow(); //check if green pressed then call OnYellow function
if(green==HIGH)  Ongreen();//check if green pressed then call OnGreen function
}
void OnRed() { // function for red button pressed
while(digitalRead(reset)==0) {
digitalWrite(redlight, HIGH); //turn on red light
digitalWrite(greenlight , LOW);
digitalWrite(yellowlight , LOW);
 }
clear_lights();
}
void Onyellow() { // function for yellow pressed
while(digitalRead(reset)==0){
digitalWrite(redlight,LOW);
digitalWrite(greenlight ,LOW);
digitalWrite(yellowlight ,HIGH); //turn on yellow light
  }
clear_lights();
}
void Ongreen() { // function for green button pressed
  while(digitalRead(reset)==0)
  {
  digitalWrite(redlight,LOW);
  digitalWrite(greenlight ,HIGH); //turn on green light
  digitalWrite(yellowlight ,LOW);
  }
  clear_lights();
}
void clear_lights() // function for off all lights connected from pin 8 to 10
{
  for (int i = 8; i < 11; i++) {
digitalWrite(i, LOW);
}
}
```

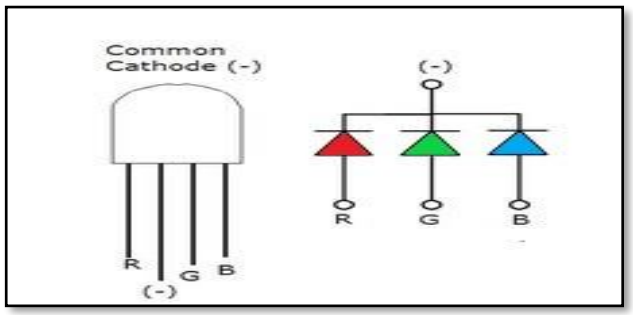
2.7 RGB LED

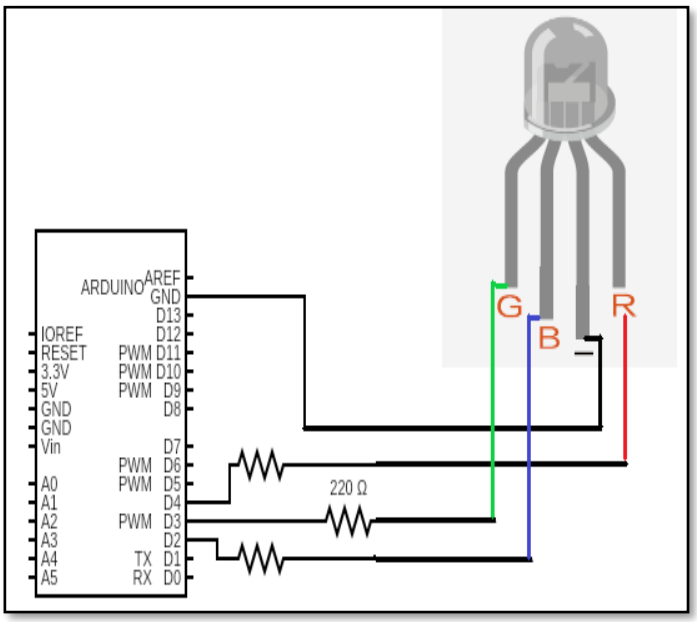
Brief: In this example we will learn how to vary the color of RGB led using Arduino. Varying the PWM values causes the change in color. Pulse Width Modulation (or PWM) is a technique for controlling power. We also use it here to control the brightness of each of the LEDs. Using PWM the amount of power delivered to a device can be controlled. Duty Cycle and Frequency concepts are used in PWM to control brightness of RGB LED. Duty cycle indicates the duration for which the pulse is HIGH over it period. In lay man terms this duty cycle is a value in percent of ON status compared to OFF status. From the figure below we can see the formula to calculate the duty cycle. It is measured in percentage and it indicates the voltage between OFF and ON levels (usually 0V and 5V).

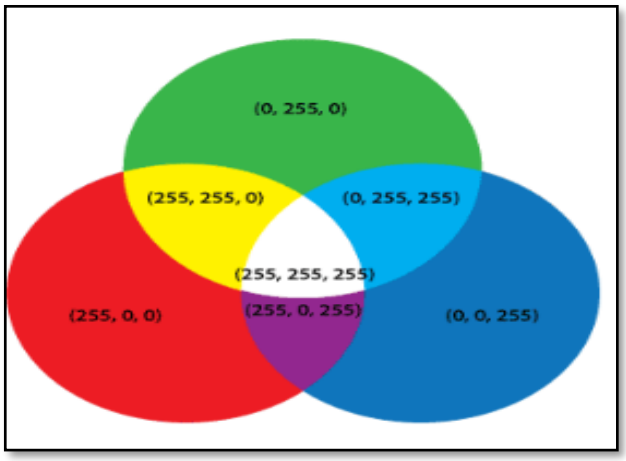
Components: Button switch (4), Red LED, Yellow LED, Green LED, Resistors 220Ω (3), 10KΩ (4),Breadboard

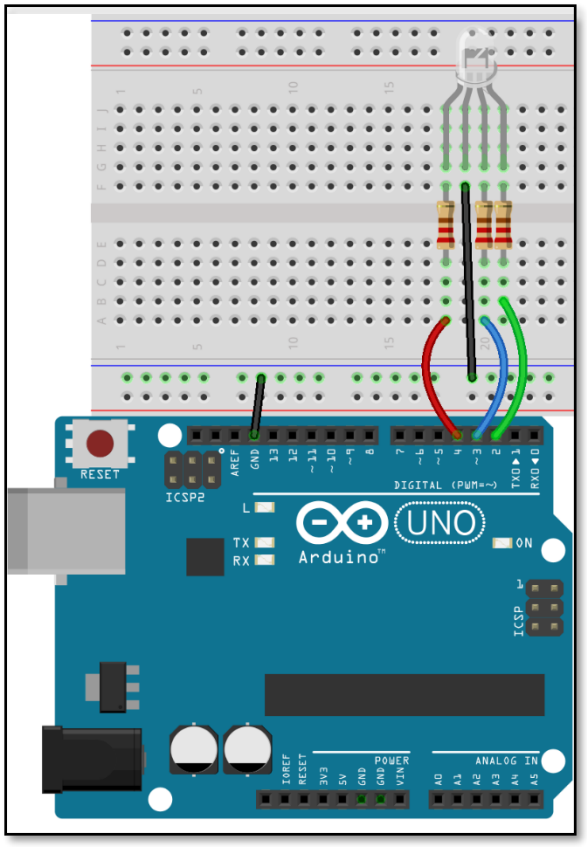












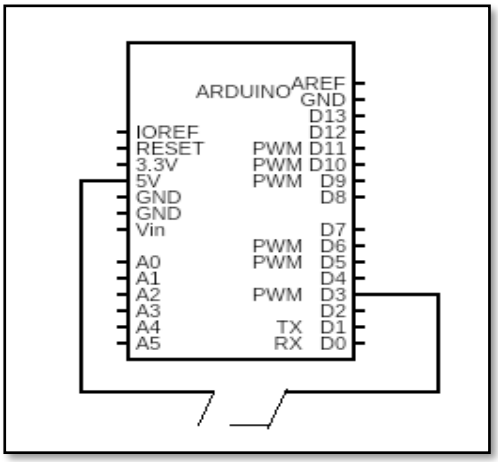
For color to stay longer you can increase the delay in the code

```
//RGB Led
// Written by ABRA ELECTRONICS
int blue_pin = 2;
int green_pin = 3;
int red_pin= 4;
void setup() {
  pinMode(red_pin, OUTPUT);
  pinMode(green_pin, OUTPUT);
  pinMode(blue_pin, OUTPUT);
}
void loop() {
  RGB_CODE(255, 0, 0); // Red colour
  delay(500); // delay by half a second
  RGB_CODE(0, 255, 0); // Green colour
  delay(500);
  RGB_CODE(0, 0, 255); // Blue colour
  delay(500);
RGB_CODE(255, 255, 0); // Yellow colour
  delay(500);
  RGB_CODE(255, 255, 255); // White colour
  delay(500);
RGB_CODE(255, 255, 125); // Raspberry colour
  delay(500);
  RGB_CODE(0, 255, 255); // Cyan colour
  delay(500);
  RGB_CODE(255, 0, 255); // Magenta colour
  delay(500);
}
void RGB_CODE(int red_code, int green_code,
int blue_code)
{ // function to read the pin values of RGB
  analogWrite(red_pin, red_code);
  analogWrite(green_pin, green_code);
  analogWrite(blue_pin, blue_code);
}
```

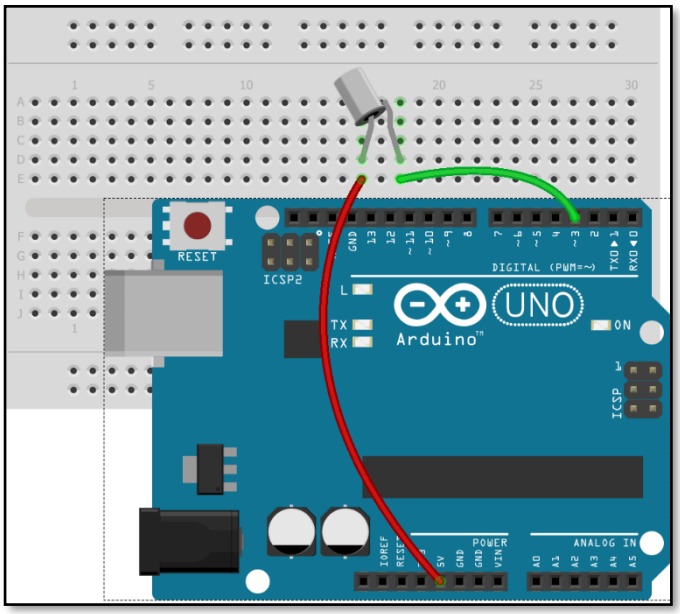
2.8 Tilt Switch (SENS-39)

Brief: Tilt switch is a switch which opens and closes an electrical circuit when it is tilted at certain angles. After connecting the circuit if the breadboard is tilted to certain angle the LED will glow. They are small, inexpensive, low-power, and easy-to-use.

Components: SENS-39



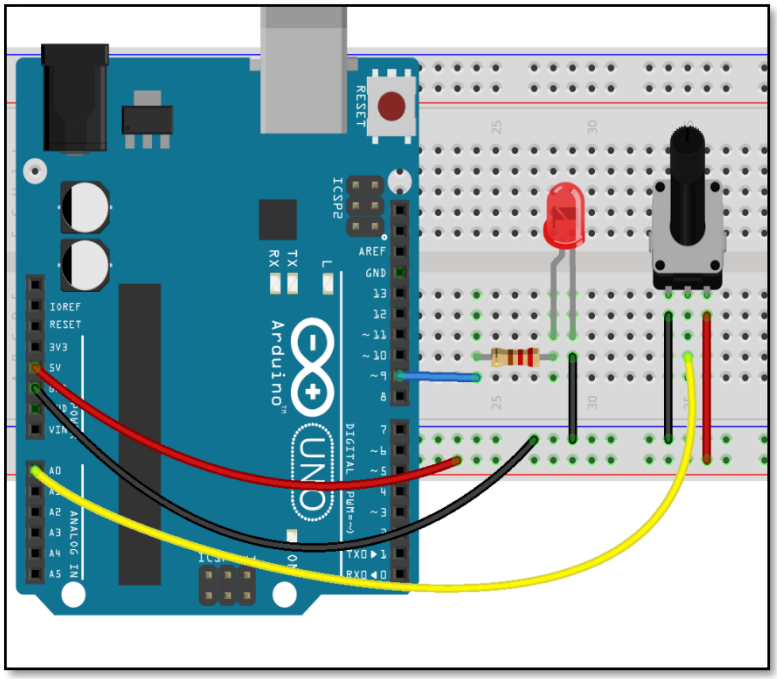

```
// Tilt switch
// Written by ABRA ELECTRONICS
int led_pin = 13; // Built in LED connected to pin 13
int tilt_switch = 3; // tilt switch connected to pin 3
int value;

void setup()
{
        pinMode(led_pin, OUTPUT);
        pinMode(tilt_switch, INPUT);
}
void loop()
{
value = digitalRead(tilt_switch); //Read the tilt value
        if(value == HIGH)
        {
                digitalWrite(led_pin, HIGH);
        }
        else
        {
                digitalWrite(led_pin, LOW);
        }
}
```

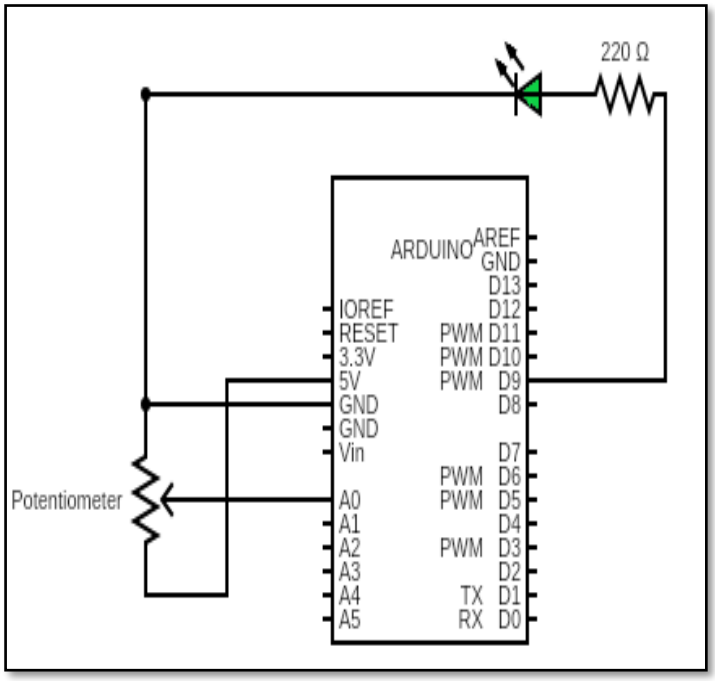


2.9 Controlling an LED by potentiometer

Brief: Brightness of the potentiometer is controlled using potentiometer by changing the value of the resistance.

Components: Breadboard, 220Ω resistor, LED.

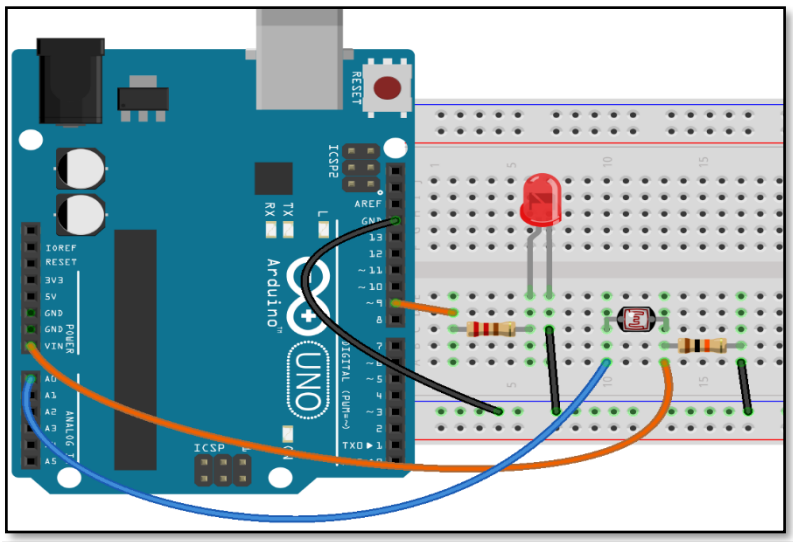


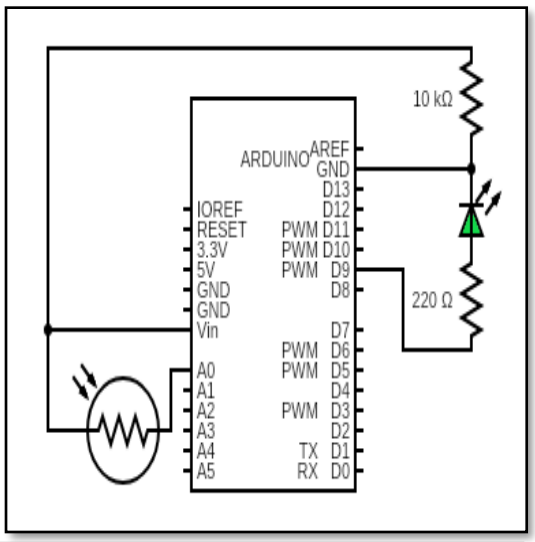



```
//          Controlling an LED by potentiometer
int Analoginput=A0;  //Analog pin A0 receives input controlled by potentiometer.
int out1=9;   //Pin 9 is output to which LED is connected.
void setup()
{
pinMode(out1, OUTPUT); //Pin 9 is output
}
void loop()
{
int recievedinput=analogRead(Analoginput); //Reading the voltage out by potentiometer
int ledbrightness=recievedinput/4;      //Dividing by 4 to bring it in range of 0 – 255
analogWrite(out1, ledbrightness);
}
```

2.10 Photoresistor (Photo-300)

Brief: A photoresistor or photocell is a light-controlled variable resistor. It works on the principle of photoconductivity. The resistance of a photoresistor decreases with increasing incident light intensity. When resistance decreases current flow through it.



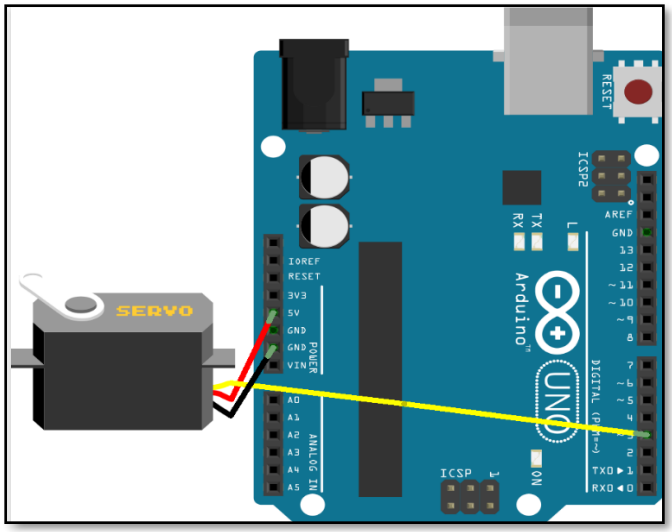


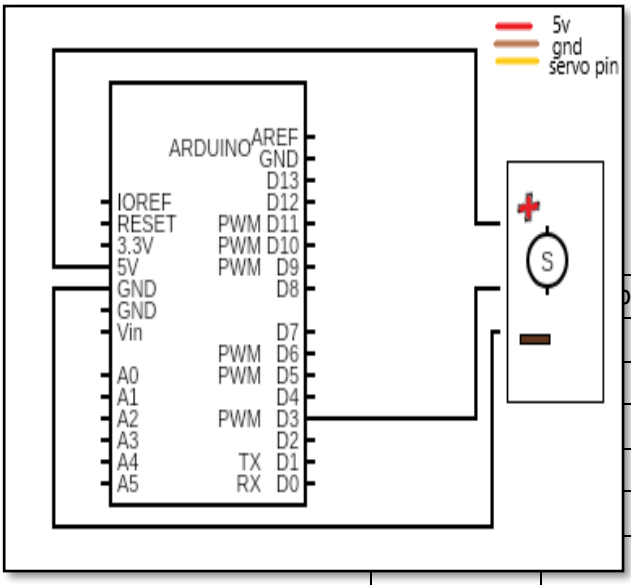

```
//Use a photoresistor to turn on an LED in the dark
const int photocell = A0; // Photoresistor at Arduino analog pin A0
const int ledPin=9;      // Led pin at Arduino pin 9
int digitalvalue; // to store digitalvalue from photoresistor (0-1023)
void setup(){
 pinMode(ledPin, OUTPUT);  // Set lepPin - 9 pin as an output
 pinMode(photocell, INPUT);
}
void loop(){
  digitalvalue = analogRead(photocell);

  if (digitalvalue > 30){
    digitalWrite(ledPin, LOW);  //Turn led off if there is enough light in room
  }
  else{
    digitalWrite(ledPin, HIGH); //Turn led on if it is dark
  }
}
```

2.11 Servo (SG-90)

Brief: A **servo motor** is an electrical device which can rotate an object with high precision. If you want to rotate an object at a specific angle, we use servo motor. Servo motor is controlled by PWM (Pulse with Modulation). In this project we can connect small servo motors directly to an Arduino to control the shaft position very precisely. Arduino sends PWM signal to the servo which then rotates by an angle depending upon the pulse width.

Components: Servo motor



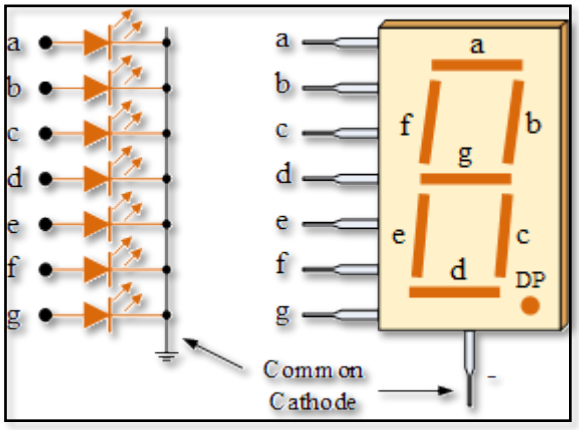


	pin
g	9
-	gnd

```
#include <Servo.h>  // Include the Servo library
int ser_pin = 4; //  Servo pin connected to pin 4
Servo ser_obj;  // instantiate servo object
int i=0;
void setup() {
  ser_obj.attach(ser_pin);
}
void loop(){
   while(ser_pin=HIGH){
  for (i = 0; i <= 360; i++) { //anti clockwise
   ser_obj.write(i);
   delay(10);
  }
  for ( i = 360; i>= 0; i--) { //clockwise
   ser_obj.write(i);
   delay(10);
  }
   }
}
```

2.12 1 digit 7-segment display (S-5101AS)

Brief: Interfacing seven segment display to Arduino. In the common cathode display, all the cathode connections of the LED segments connected to ground. The segments are illuminated by applying “HIGH”, or logic “1” signal via a current limiting resistor to forward bias the Anode terminals. For example, if you want to illuminate segment ‘a’ then pin7 on Arduino that is connected to segment ‘a’ is made HIGH



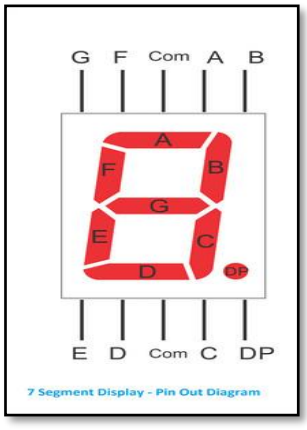
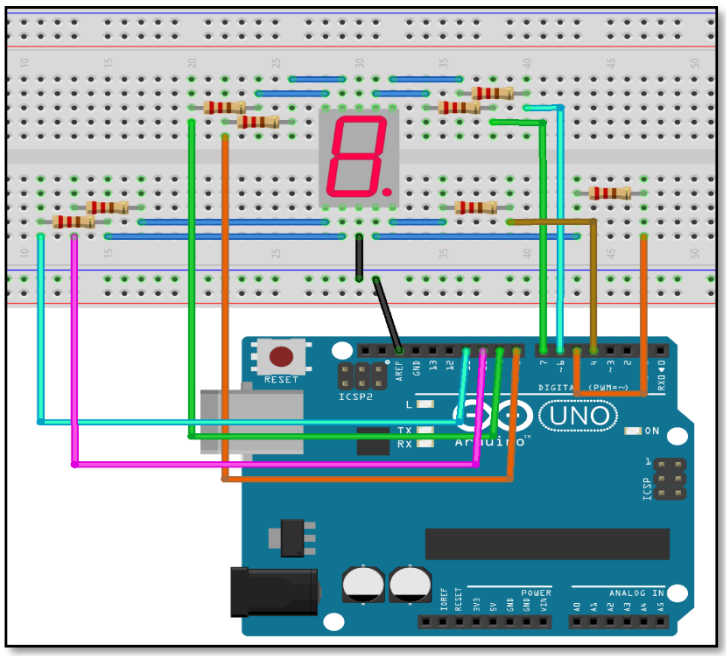
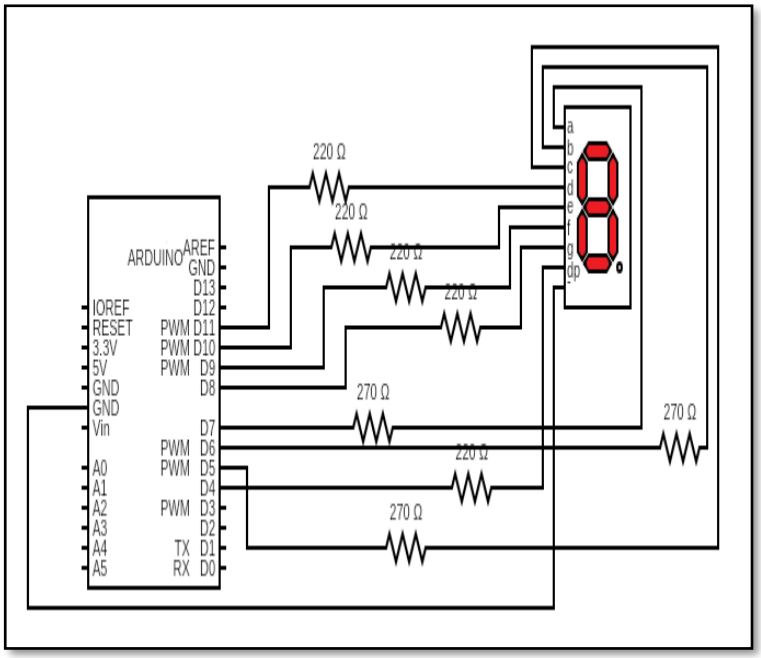


Table for Connections





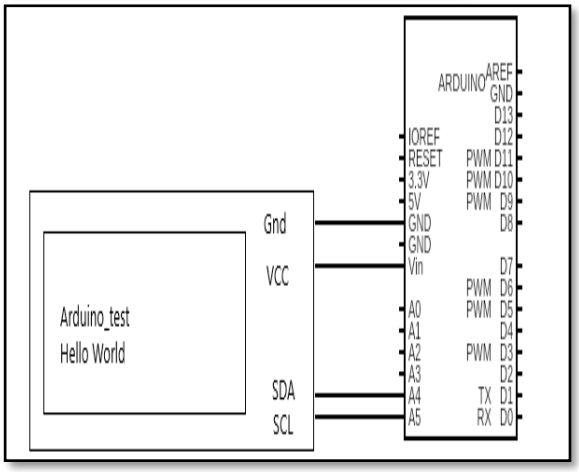

```
// Declare array for 7 bits and activate the required segment it is in order of (c,b,a,f,g,e,d)
int alpha[7][7] = { { 1,0,1,1,1,1,1 },// G
                    { 0,0,1,1,1,1,0 },   // F
                    { 0,0,1,1,1,1,1 },   // E
                    { 1,1,1,1,0,1,1 },   // D
                    { 0,0,1,1,0,1,1 },   // C
                    { 1,1,1,1,1,1,1 },   // B
                    { 1,1,1,1,1,1,0 }  };  // A
void print_data(int);
void setup() {
   int  i;
   for(i=5;i<=11;i++)
      pinMode(i,OUTPUT);
}
void loop() {
  for (int counter = 7; counter >0; counter--) {
   delay(1000);
   print_data(counter-1);
  }
  delay(1000);
}

// this functions writes values to the seven segment pins
void print_data(int number) {
  int out_pin=5;
  for (int j=0; j < 7; j++) {
   digitalWrite(out_pin, alpha[number][j]);
   out_pin++;
  }
}
```

2.13 LCD 16x2 with I2C (LCD-MOD-13)

Brief: LCD I2C uses I2C interface, so it has 4 pins NAMELYGND, VCC, SDA (I2C data signal), SCL (I2C clock signal).

Components: I2C LCD, Jumper wires. Note: If the LED does not display characters then try LCD contrast adjust.

```
#include <Wire.h> // Library for I2C communication
#include <LiquidCrystal_I2C.h> // install this library
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27,
16 column and 2 rows

void setup()
{
  lcd.init();// initialize the lcd
  lcd.backlight();
  lcd.setCursor(1,0);       // set cursor to   (0, 0)
  lcd.print("Arduino_test");       // print message at (0, 0)
  lcd.setCursor(0,1);       //Set to bottom row
  lcd.print("Hello World"); // print message at (0,1)
}
void loop()
{}
```



2.14 Thermistor (334-103)

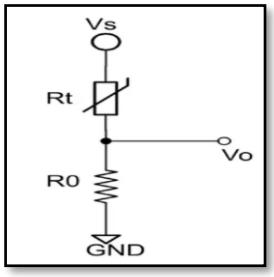
Brief: A thermistor is a type of resistor whose resistance is dependent on temperature. Thermistors are of two opposite types:

- With **NTC** thermistors, resistance ***decreases*** as temperature rises. An NTC is commonly used as a temperature sensor.
- With **PTC** thermistors, resistance ***increases*** as temperature rises.

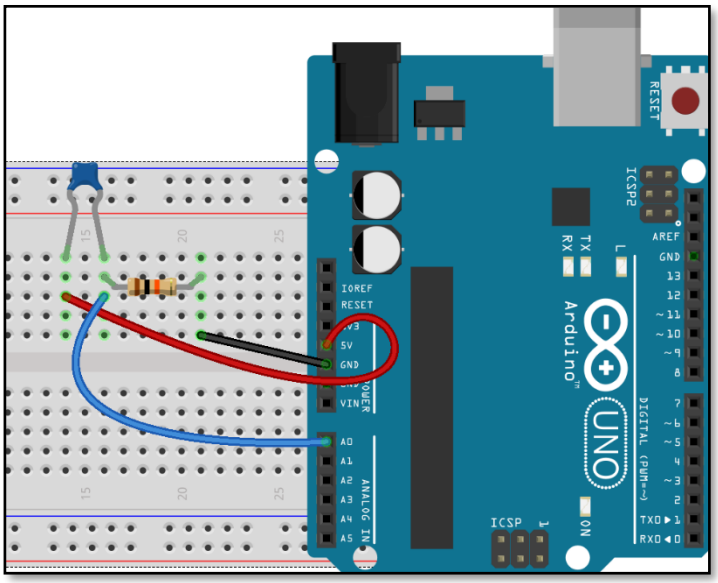
In this experiment we create a voltage divider between thermistor and 10kΩ resistor and perform the calculation.

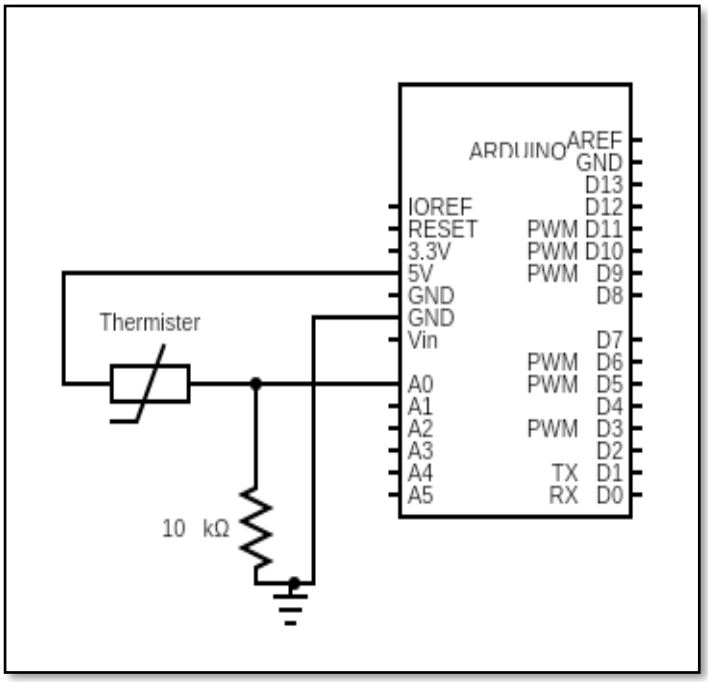
- **Vo = Vs * (R0 / (Rt + R0**
- **Rt = R0 * ((Vs / Vo) - 1)**
- **1/T = A + B*ln(R) + C*(ln(R))^3**

Components: 10K thermistor, 10kΩ resistor.



```
#include <math.h>
int thermistor = A0;
int Vo;
float R1 = 10000; / / resistor value (R1)
float logR2, R2, T, Tc, Tf;
float A = 1.009249522e-03, B = 2.378405444e-04, C =
2.019202697e-07; / / Value of constants

void setup() {
Serial.begin(9600);// set baud rate for serial
communication
}
// calculations
void loop() {

  Vo = analogRead(thermistor);
  R2 = R1 * (1023.0 / (float)Vo - 1.0);
  logR2 = log(R2);
  T = (1.0 / (A + B*logR2 + C*logR2*logR2*logR2));
  Tc = T - 273.15; //Kelvin to celcius conversion
  Tf = (Tc * 9.0)/ 5.0 + 32.0; //celcius to Fahrenheit

  Serial.print("Temperature measured: ");
  Serial.print(Tf);
  Serial.print(" Fahrenheit; ");
  Serial.print(Tc);
  Serial.println(" Celcius");
  delay(500);
}
```



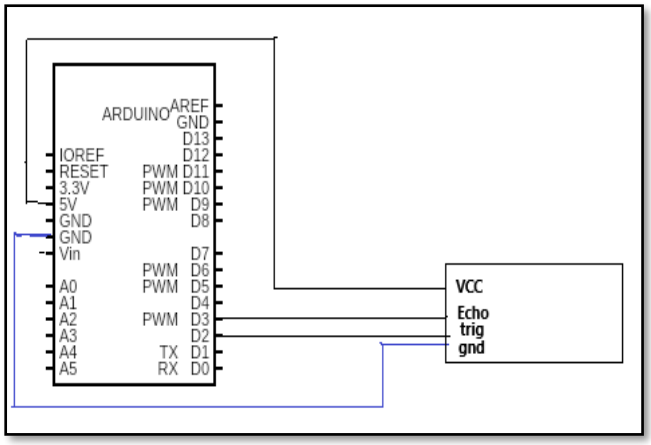


2.15 Ultrasonic sensor (HC-SR04)

Brief: Ultrasonic Sensor HC-SR04 is a sensor that can measure distance. It emits an ultrasound at 40000 Hz (40kHz) which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Using the time taken to travel and the speed of the sound distance is calculated.

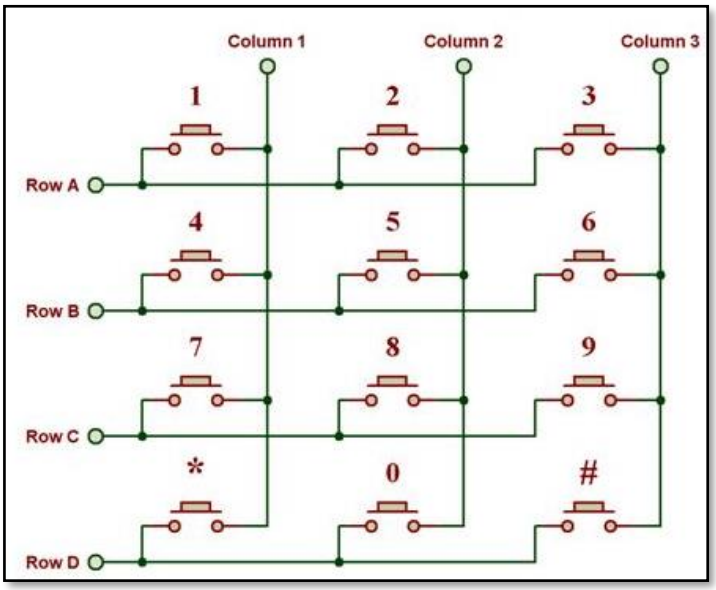
If x is the distance between point A and point B then 2x will be the distance travelled from A to B and the B to A. We know that speed of the ultrasound is about 340 meters per second.

- Distance = Speed*Time.
- Distance=2x.
- 2x= 340*Time Now time is given by pulseIn() function. x=340*Time/2




```
const int echo=3; //  D3 on Arduino to pin Echo of HC-SR04

const int trig=2 ;// D2 Arduino to pin Trig of HC-SR04
long travel_time; //  travel_time of sound wave
int distance; // variable for the distance measurement
void setup() {
  pinMode(trig, OUTPUT); // Set trig (pin 2) as an OUTPUT
  pinMode(echo, INPUT); // Set echo (pin 3) as an INPUT
  Serial.begin(9600); // set baud rate tor communication
}
void loop() {
  // Clear the trig condition
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);  // activate trif for 20 microseconds
  delayMicroseconds(20);
  digitalWrite(trig, LOW);
  // Reads a pulse (either HIGH or LOW) on a pin. For example, if
  //value is HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go
LOW and stops timing
  travel_time = pulseIn(echo, HIGH);
  // Calculate the distance
  distance = travel_time * 0.034 / 2; // One way travel time= (to and fro time/2)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance measured is: ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

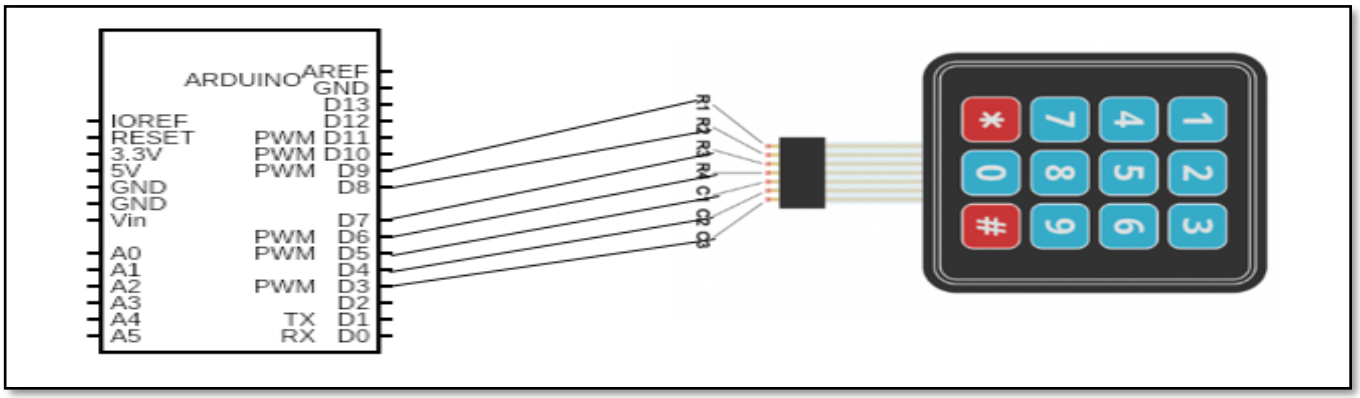
2.16 4x3 Keypad (419-ADA)



Brief: 4x3 keypad has 4 rows and 3 columns. If no key has been pressed, then all columns will remain HIGH. Pressing a button shorts one of the row lines to one of the column lines, allowing current to flow between them. For example, when key '2' is pressed, column 2 and row 1 are shorted so column 2 and row 1 are LOW.

Components: 4x3 Keypad, Jumper wires

Note: while compiling the code if there is an error regarding Keypad.h such as library does not exist then make sure to add keypad library by ***sketch->include library->manage library.*** In type select Arduino search for keypad and install keypad for matrix type library.

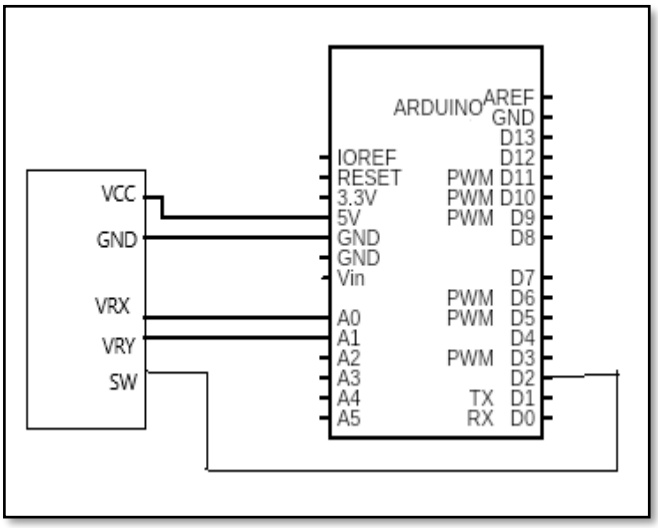



```
#include <Keypad.h>  //include keypad library
// declare rows and coloumns
const byte row = 4; //four rows
const byte col = 3; //three columns
char keys[row][col] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte row_pin[row] = {9, 8, 7, 6}; //connect to the row pinouts of the keypad
byte col_pin[col] = {5, 4, 3}; //connect to the column pinouts of the keypad
//Create an object of keypad (instantiation)
Keypad keypad = Keypad( makeKeymap(keys), row_pin, col_pin, row, col );
void setup(){
  Serial.begin(9600); // set baudrate
}
void loop(){
  char key_pressed = keypad.getKey();// Read the key pressed
  if (key_pressed){              // check if button is pressed
    Serial.print("Key Pressed is : ");
    Serial.println(key_pressed);
  }
}
```

2.17 Joystick PS2 (AM-JOYSTICK)

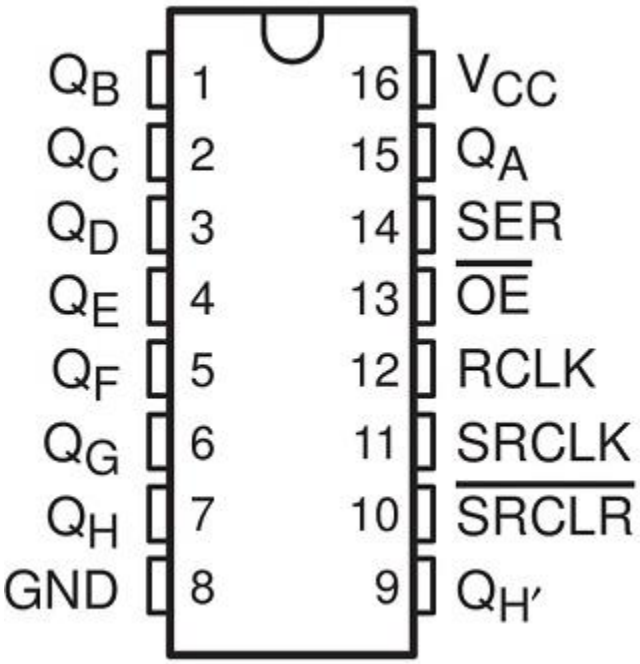
Brief: PS2 controllers have two **analog** joysticks. Analog joysticks are basically potentiometers, so they return analog values. Here two **potentiometers** are positioned at right angles to each other below the joystick. Current flows continuously through joysticks and the amount of current is determined by the amount of resistance. Resistance is increased or decreased based on the position of the joystick. The goal of the joystick is to communicate motion in 2D (X axis and Y axis) to Arduino. Therefore, two independent 10K potentiometers (one per axis) is used in joystick.

Components: Joystick, Wires



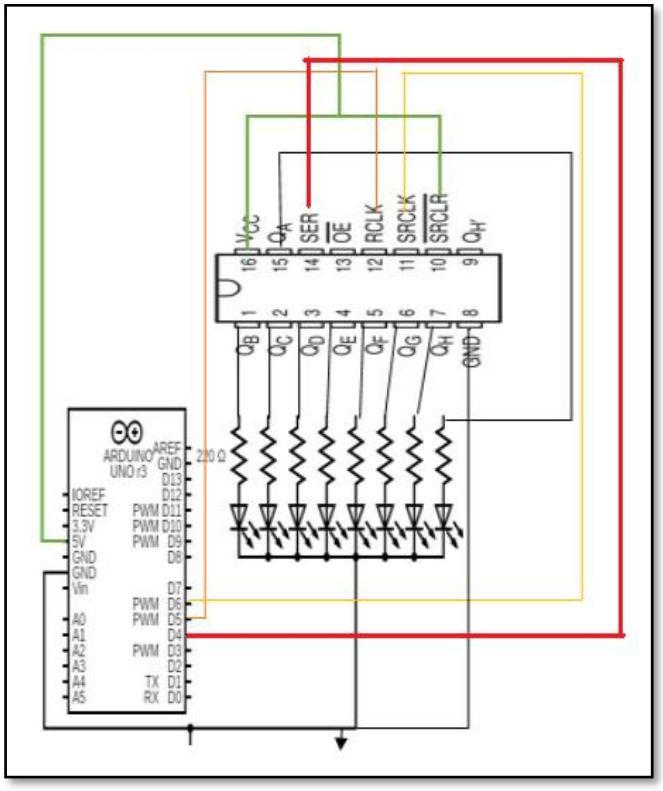

```
// JOYSTICK
const int SW = 2; // digital pin connected to switch
const int VRx = A0; // analog pin connected to X output
const int VRy =A1; // analog pin connected to Y output
int mapX=0;
int mapY=0;
int x=0;
int y=0;
void setup() {
  pinMode(SW, INPUT);
  digitalWrite(SW, LOW);
  Serial.begin(9000);
}
void loop() {
  Serial.print("Joystick  ");
  Serial.print(digitalRead(SW));
  Serial.print("\n");
  Serial.print("X-axis: ");
  x=analogRead(VRx);
   mapX = map(x, 0, 1023, -512, 512);
  Serial.print(x); // print the value of x
  Serial.print("\n");
  Serial.print("Y-axis: ");
   y=analogRead(VRy);
  mapY = map(y, 0, 1023, -512, 512);
  Serial.print(y); // print the value of y
  Serial.print("\n\n");
  delay(500);
}
```

2.18 Interfacing Shift Register (74HC595)



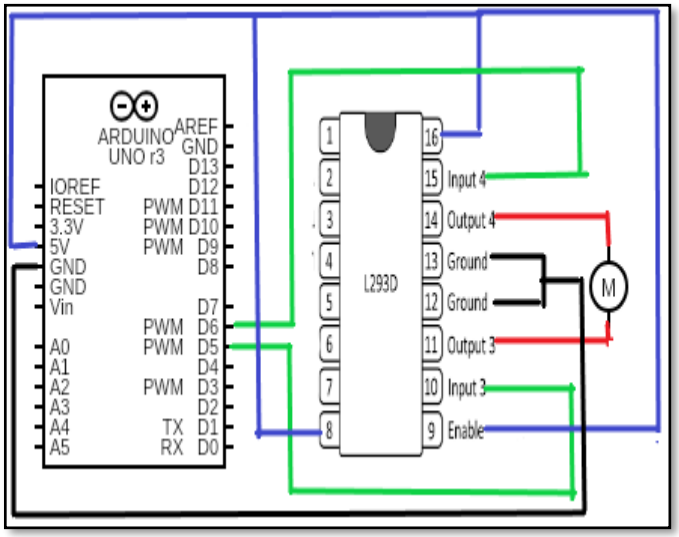
- GND should be connected to the ground of Arduino.
- VCC is the power supply for 74HC595 shift register which we connect the 5V pin on the Arduino.
- SER (Serial Input) pin is used to feed data into the shift register a bit at a time.
- SRCLK (Shift Register Clock) is the clock for the shift register. Bits are shifted on the rising edge of the clock.
- RCLK (Register Clock / Latch when logic HIGH, the contents of Shift Register are copied into the Storage/Latch Register; which ultimately shows up at the output.
- SRCLR (Shift Register Clear) pin allows us to reset the entire Shift Register, making all its bits 0, at once. This is a negative logic pin, we need to set the SRCLR pin LOW for reset. When reset not required, this pin should be HIGH.
- QA–QH (Output Enable) are the output pins connected to output like LEDs.

```
int latchPin = 5; //to RCLK
int clockPin = 6; //to SRCLK
int dataPin = 4; //to SER
void setup()
{
   Serial.begin(9600);
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop()
{
//count from 1 to 265 and display binary values on
leds
  for ( int i = 0; i < 256;i++)
  {
  digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, MSBFIRST, i);
   Serial.print(i);
   digitalWrite(latchPin, HIGH);
    delay(500);
  }
}
```



2.19 Interfacing L293D Motor Driver (L293D)

Brief: LD93D Motor driver can run two motors at a time on either direction. In this experiment we use LD93D to run a single DC motor. It works on the principle of H-bridge which allows the voltage to be flown in either direction.

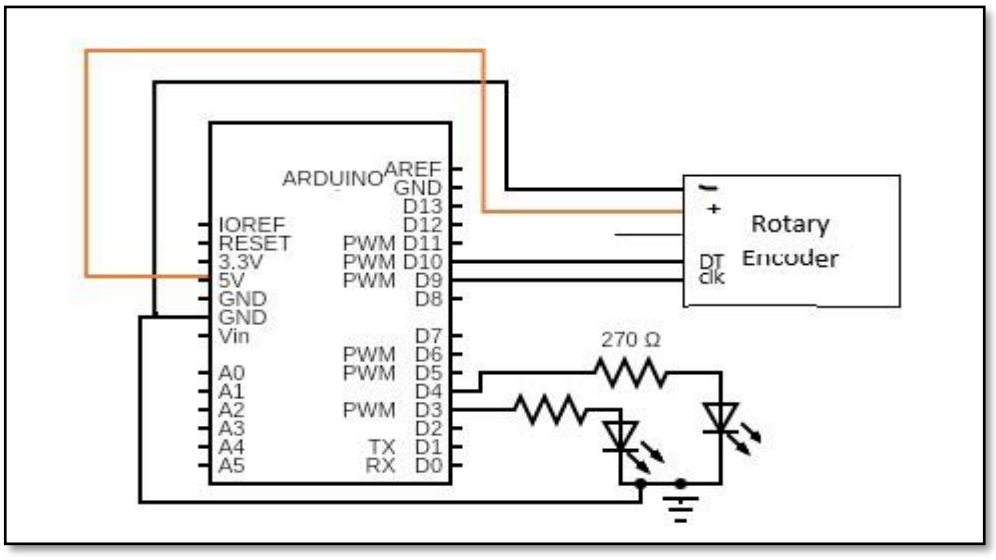



```
// Interfacing L293D Motor Driver
const int out_pin1 = 5;  // to Pin 14 of L293
clockwise
const int out_pin2  = 6;  //Pin 11 of L293 anti
clockwise
void setup(){
   //Set output pins
   pinMode(out_pin1, OUTPUT);
   pinMode(out_pin2, OUTPUT);
}
void loop(){
 while(1){ //clockwise rotation
  digitalWrite(out_pin1, HIGH);
   digitalWrite(out_pin2, LOW);
   delay(5000);
  // stop motor
   digitalWrite(out_pin1, LOW);
   digitalWrite(out_pin2, LOW);
    delay(500);
  //counterclock wise rotation
   digitalWrite(out_pin1, LOW);
   digitalWrite(out_pin2, HIGH);
   delay(5000);
}
}
```

2.20 Rotary Encoder (AM-127)

Brief: A rotary encoder is a position sensor used to determine the angular position of a rotating shaft.

We have used 2 LEDs one to specify clockwise direction and the other one for anti-clockwise direction.



```
// Rotary encoder
const int DT=10; // DT pin of Rotary Encoder
const int CLK=9;// CLK pin of Rotary Encoder
const int out1=3;//red led
const int out2=4;//green led
 int Position = 0;
 int current_position;
 int previous_position;
 void setup() {
  pinMode (CLK,INPUT);
  pinMode (DT,INPUT);
  pinMode (out1,OUTPUT);
  pinMode (out2,OUTPUT);
  Serial.begin (9600);// set baud rate
    previous_position = digitalRead(CLK); // read current value of CLK
 }
 void loop() {
  current_position = digitalRead(CLK);
    if (current_position != previous_position)
    {
       if (digitalRead(DT) != current_position)
       {
     Position ++;
     digitalWrite(out1, HIGH); // On led if its clock wise
      digitalWrite(out1, LOW);
    }
    else {
     Position --;
     digitalWrite(out2, HIGH); // On led if its anti-clock wise
      digitalWrite(out2, LOW);
    }
    Serial.print("Position: ");
    Serial.println(Position);
  }
  previous_position = current_position; // update previous state of CLK with the current state
 }
```

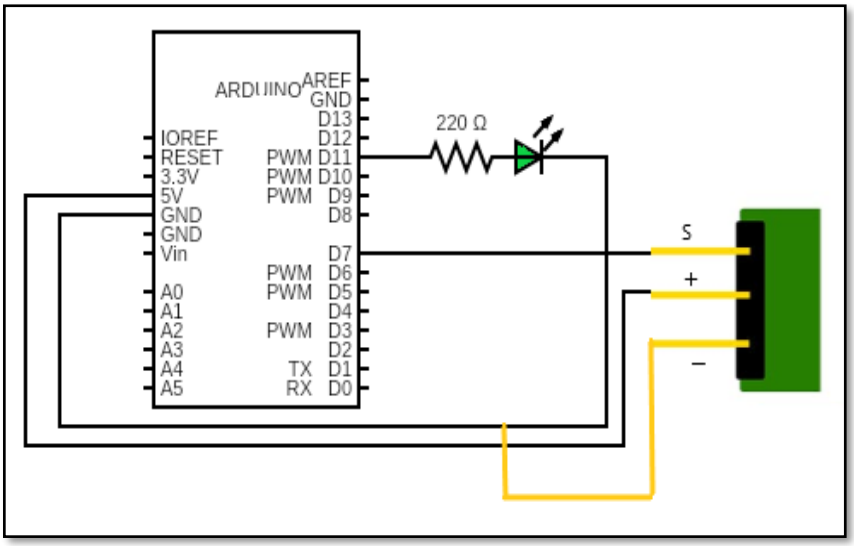
2.21 Remote control (WL-110)

Brief: In this project we learn how to switch on and off LED using remote control. An IR remote called a transmitter uses light to carry signals from the remote to the device it controls. It emits pulses infrared light that correspond to specific binary codes. These codes represent operations, such as power on, volume down, or channel up. The controlled device (also called the receiver) decodes the infrared pulses of light into binary code that its internal microprocessor understands. Once the signal is decoded, the microprocessor executes the commands. IR remotes require line-of-sight, which means you should not block the path between the transmitter and receiver.

Different type of remotes has different values for the keys. The table given below gives the code value of the keys on the remote that we will be using.

Components: LED, Resistor 220Ω, WL-100 (Remote control)

Code value	Keys
FFA25D	1
FF629D	2
FFE21D	3
FF22DD	4
FF02FD	5
FFC23D	6
FFE01F	7
FFA857	8
FF906F	9
FF6897	*
FF9867	0
FFB04F	#
FF18E7	Up
FF4AB5	Down
FF10EF	Left
FF5AA5	Right



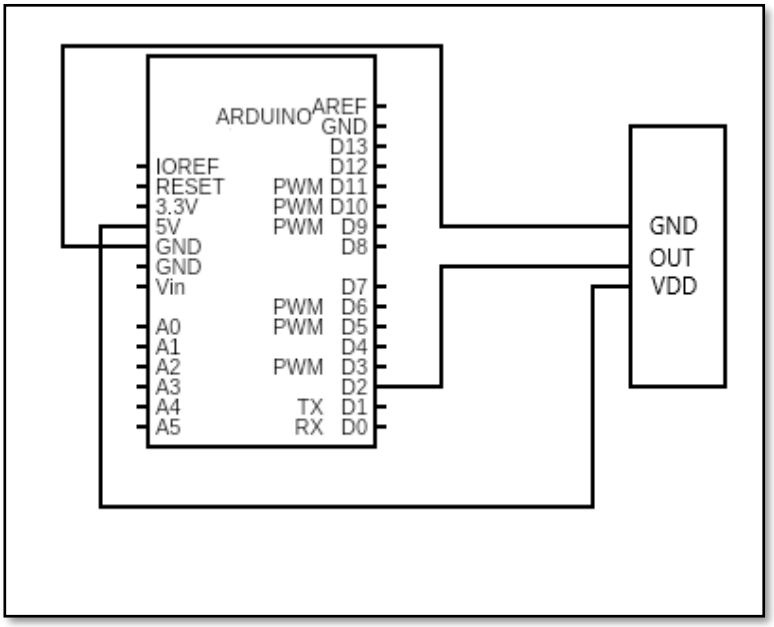

```
#include <IRremote.h>

const int RECV_PIN = 7;
IRrecv irrecv(RECV_PIN);  // create an object
decode_results results;
const int greenPin = 11;

void setup(){
  irrecv.enableIRIn();
  irrecv.blink13(true);
  pinMode(greenPin, OUTPUT);
}
void loop(){
   if (irrecv.decode(&results)){
      switch(results.value){
        case 0xFFA25D: //Keypad button "1" turn on LED
        digitalWrite(greenPin, HIGH);
        }
      switch(results.value){
        case 0xFF629D: //Keypad button "2" turn off LED
        digitalWrite(greenPin, LOW);
        }
      irrecv.resume();
   }
}
```

2.22 PIR sensor (SENS-PIR)

Brief: A passive infrared **sensor** (**PIR sensor**) is an electronic **sensor** that measures infrared (IR) light radiating from objects in its field of view. They are most often used in **PIR**-based **motion** detectors. **PIR sensors** are commonly used in security alarms and automatic lighting **applications**. Once the circuit is built and uploaded into Arduino try moving objects in front of PIR sensor which causes LED to glow, output can also be seen on serial monitor.

Components: SENS-PIR

```
int ledPin = 13;  //builtin LED
int PIR_pin = 2;
int PIR_state = LOW;
int state = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(PIR_pin, INPUT);
  Serial.begin(9600);
}
void loop(){
  state = digitalRead(PIR_pin);
  if (state == HIGH) {       //motion detected turn
on LED
    digitalWrite(ledPin, HIGH);
     Serial.println("Motion detected");
    }
  else {
    digitalWrite(ledPin, LOW); // turn LED off
     Serial.println("No Motion");
    }
}
```

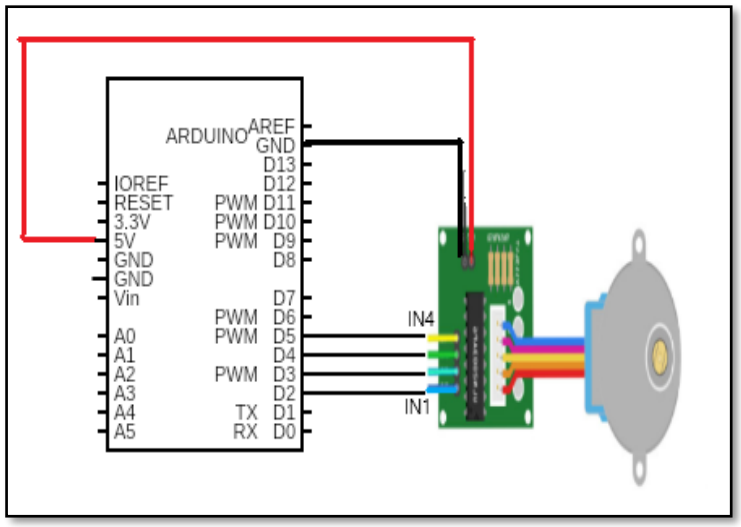


2.23 Stepper motor with driver (MOT-28BYJ-48)

Brief: The 28BYJ-48 stepper motor is a stepper motor, which converts electrical pulses into discrete mechanical rotation. The 28BYJ-48 stepper motor consumes high current and hence, we will need to use a driver IC ULN2003 to control the motor with a microcontroller like the Arduino.

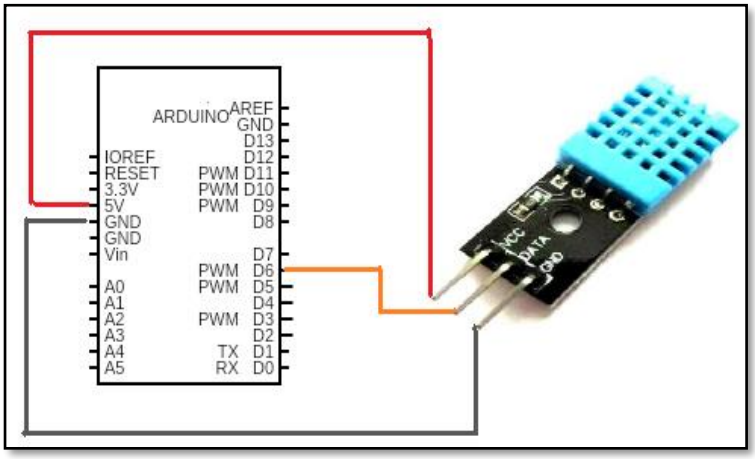
```
#include <Stepper.h> // include stepper library
const int stepsPerRevolution = 1048;  // number of
steps per rotation:
// Pin 2 to IN1 on the ULN2003 driver
// Pin 3 to IN2 on the ULN2003 driver
// Pin 4 to IN3 on the ULN2003 driver
// Pin 5 to IN4 on the ULN2003 driver
// Create stepper object called 'myStepper', note the
pin order:
Stepper myStepper = Stepper(stepsPerRevolution, 2,
4, 3, 5);
void setup() {
  myStepper.setSpeed(10);   // Set the speed to 10
rpm:
  Serial.begin(9600);
}
void loop() {
  Serial.println("clockwise");   // Clockwise rotation
  myStepper.step(stepsPerRevolution);
  delay(500);
  Serial.println("Anticlockwise"); // Anti-Clockwise
rotation
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

Components: MOT-28BYJ-48, Jumper wires



2.24 Humidity sensor (SENS-DHT11-BB)

Brief: In this project we interface a humidity sensor to measure temperature and humidity. DHT11 consist of a humidity sensing component, a NTC temperature sensor (or thermistor) and an IC. It has a humidity sensing component which has two electrodes with moisture holding substrate between them. When the humidity changes, the conductivity of the substrate changes or the resistance between these electrodes' changes. This change in resistance is measured and processed by the IC which makes it data to be read by a microcontroller.



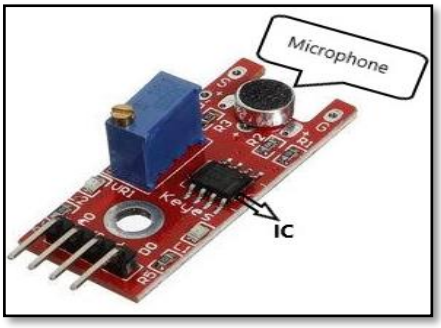
Components: Humidity sensor (SENS-DHT11-BB), Jumper wires

```
#include "DHT.h"
#define DHTTYPE DHT11
int dht_pin = 6;
DHT dht(dht_pin, DHTTYPE);// crteate object fot DHT
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  float heat_index_F = dht.computeHeatIndex(f, h);
  float heat_index_C = dht.computeHeatIndex(t, h, false);
  if (isnan(h) || isnan(t) || isnan(f)) { //Returns whether h or t or f is a NaN (Not-A-Number) value
    Serial.println("Not able to read");
    return;
  }
  Serial.print("Humidity is  ");
  Serial.print(h);
  Serial.print(" \n");
  Serial.print("Temperature is");
  Serial.print("\t");
  Serial.print(t);
  Serial.print(" Degree Celcius");
  Serial.print("\t");
  Serial.print(f);
  Serial.print(" Degree Fahrenheit");
  Serial.print("\t");
  Serial.print(" \n");
  Serial.print("Heat index is");
  Serial.print("\t");
  Serial.print(heat_index_C);
  Serial.print(" Degree Celcius");
  Serial.print("\t");
  Serial.print(heat_index_F);
  Serial.println(" Degree Fahrenheit");
}
```

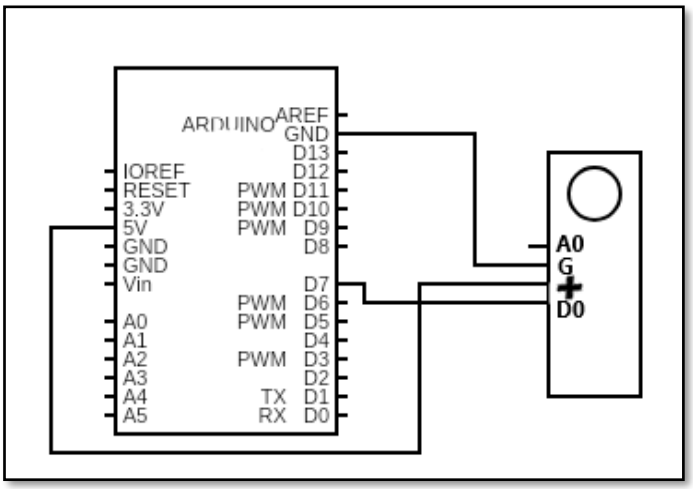
2.25 Sound Detector

Brief: Here we use SENS-42 to detect the sound of clap. When sound is detected a message will be displayed on the serial monitor. The microphone in the Sound sensor module detects the sound. This sound is fed into the LM393 IC. This IC is a voltage comparator that compares threshold voltage with the output from the microphone.

Components: SENS-42, Jumper wires



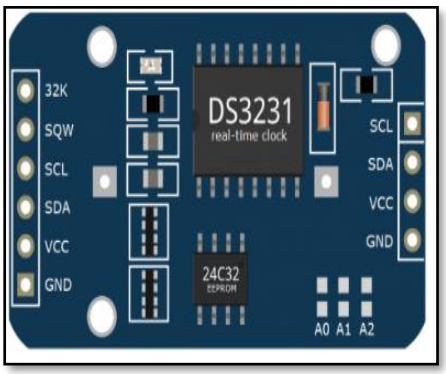

```
#define sensor_Pin 7

void setup() {
  pinMode(sensor_Pin, INPUT);  // Set
sensor pin as an INPUT
  Serial.begin(9600);
}

void loop() {
  // Read Sound sensor
  int sensorData =
digitalRead(sensor_Pin);
  // If pin goes LOW, sound is detected
  if (sensorData == LOW) {

     Serial.println("Clap detected!");
   }
}
```



2.26 RTC Module (ARD-DS-3231)

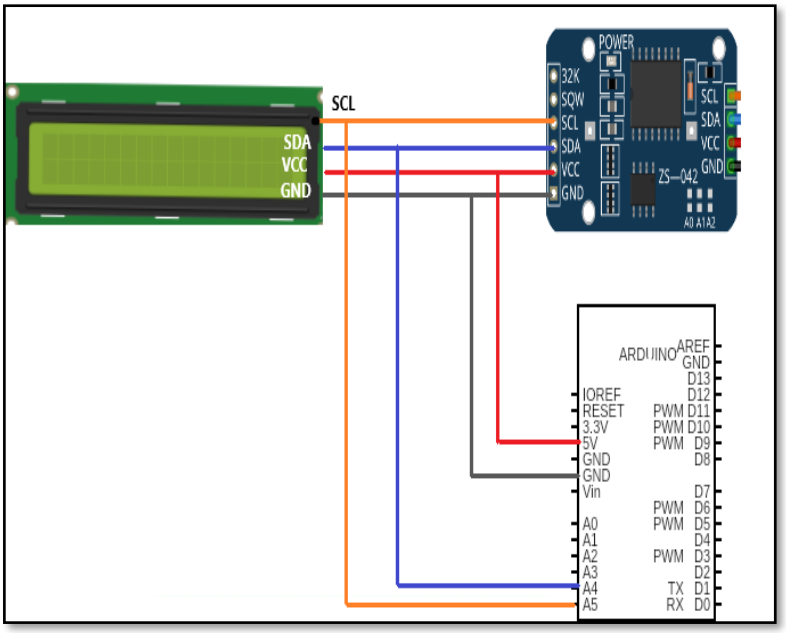
Brief: We use RTC module to display the current time, date, and day.

Components: I2C LCD, ARD-DS3231



32K – outputs from the DS3231 chip a very accurate 32KHz oscillator
SQW – outputs a square-wave signal from the DS3231 chip. The frequency of the square-wave can be changed between 1Hz, 4kHz, 8kHz, or 32kHz programmatically. this pin can also be used programmed as an interrupt output.
SCL – input pin for I2C Serial Clock
SDA – input/output pin for I2C Serial Data
VCC – power source input pin for the module; can be any voltage from +3.3V to +5.5V DC
GND – Ground pin connection

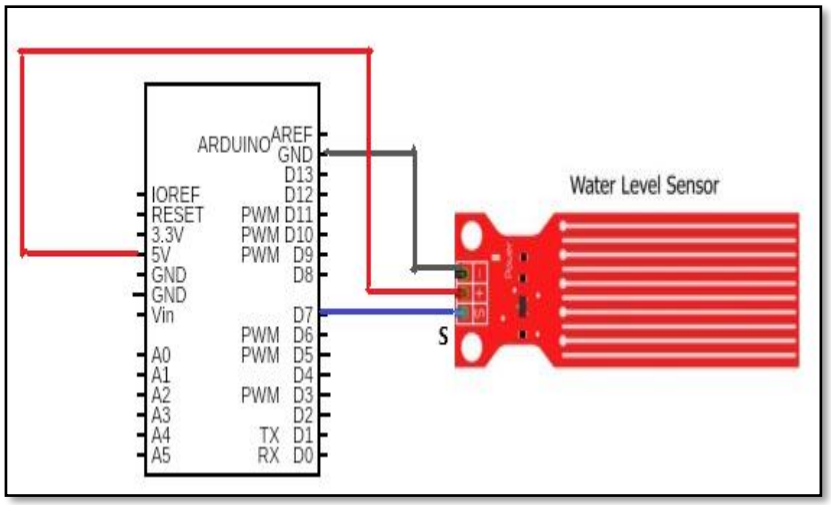
```
#include "RTClib.h"
#include <LiquidCrystal_I2C.h> // install this library
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address
0x27, 16 column and 2 rows
DateTime now;
RTC_DS3231 rtc;
//enum for daysOfTheWeek
char daysOfTheWeek[7][12] = {"Sun", "Mon", "Tue",
"Wed", "Thu", "Fri", "Sat"};
void setup ()
{
  //Serial.begin(9600);
  lcd.init();// initialize the lcd
  lcd.backlight();// to switch on the backlight of LCD
}
void loop () {
    DateTime now = rtc.now();
  lcd.setCursor(0,0);
  lcd.print(now.day());
  lcd.print('/');
  lcd.print(now.month());
  lcd.print('/');
  lcd.print(now.year());
  lcd.setCursor(11,0); // set first row and 11th
column
lcd.print(daysOfTheWeek[now.dayOfTheWeek()]);
  lcd.setCursor(1,1);
  lcd.print("Time:");
  lcd.print(now.twelveHour());
  lcd.print(':');
  lcd.print(now.minute());
  lcd.print(':');
  lcd.print(now.second());
  lcd.print("    ");
}
```



2.27 Water level sensor (SENS-78)

Brief: Water level sensor work based on variable resistance of the series of exposed parallel conductors (just like a potentiometer) whose resistance change according to the water level. The resistance is inversely proportional to the height of the water.

Components: SENS-78, Jumper wires.



```
#define sensor_switch 7

#define sensor_input A0
// Value for storing water level
int depth = 0;

void setup() {
   Serial.begin(9600);
//set pin 7 as the switch
  pinMode(sensor_switch, OUTPUT);
//switch off sensor
  digitalWrite(sensor_switch, LOW);
}
void loop() {
  int water_level = Sensor_level();
  Serial.print("Water water_level is ");
  Serial.println(water_level);
  delay(100);
}
int Sensor_level() {
  digitalWrite(sensor_switch, HIGH);  // Turn the sensor ON
  delay(10);
  depth = analogRead(sensor_input);    // Read the analog value form sensor
  digitalWrite(sensor_switch, LOW);   // Turn the sensor OFF
  return depth;           // send depth of water water_level
}
```