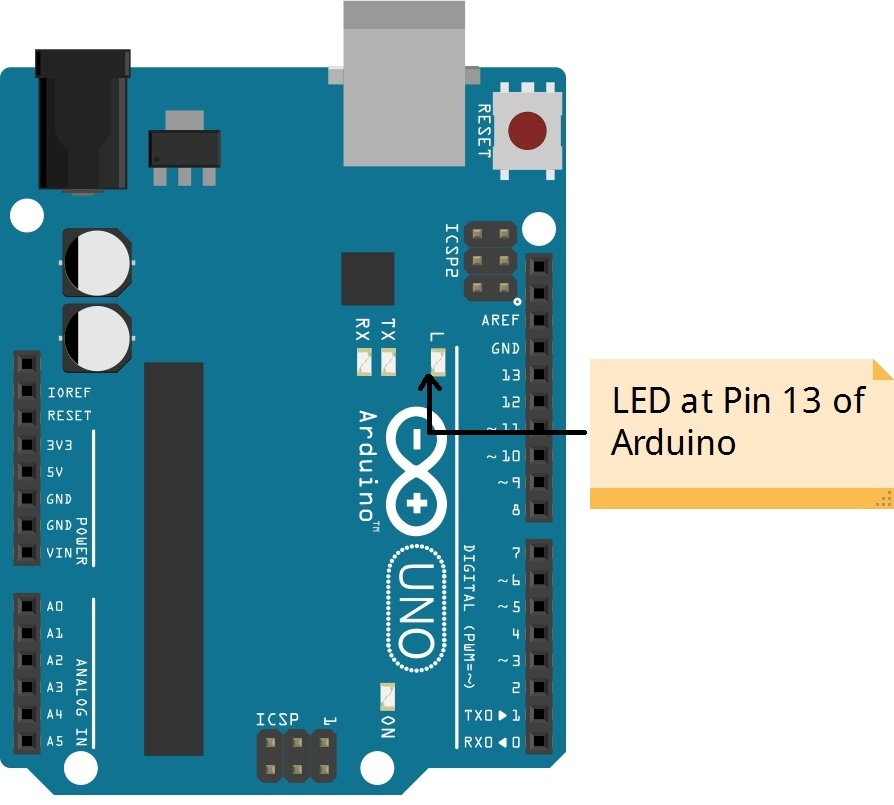
# Experiment 1

## LED Blinking :

In Arduino the very interesting and basic experiment is LED Blinking. Arduino board consists of one led connected to pin 13. In this we will turn ON and OFF Led after some specified time delay with the help of Arduino. To interface LED with Arduino via programming we have to follow some programming rules and formats.

## Circuit:



While programming Arduino one thing should be remembered that interface device should be declared its working Mode i.e. either INPUT or OUTPUT.

## Program :

int led = 13;

// every arduino board has an LED connected to pin 13 by default

void setup()

{

pinMode (led,OUTPUT); // this makes the pin no 13 work as output.

}

void loop()

{

// put your main code here, to run repeatedly:

digitalWrite(led,HIGH);// Turn ON LED

delay(100);

digitalWrite(led,LOW); // Turn OFF LED

delay(100);

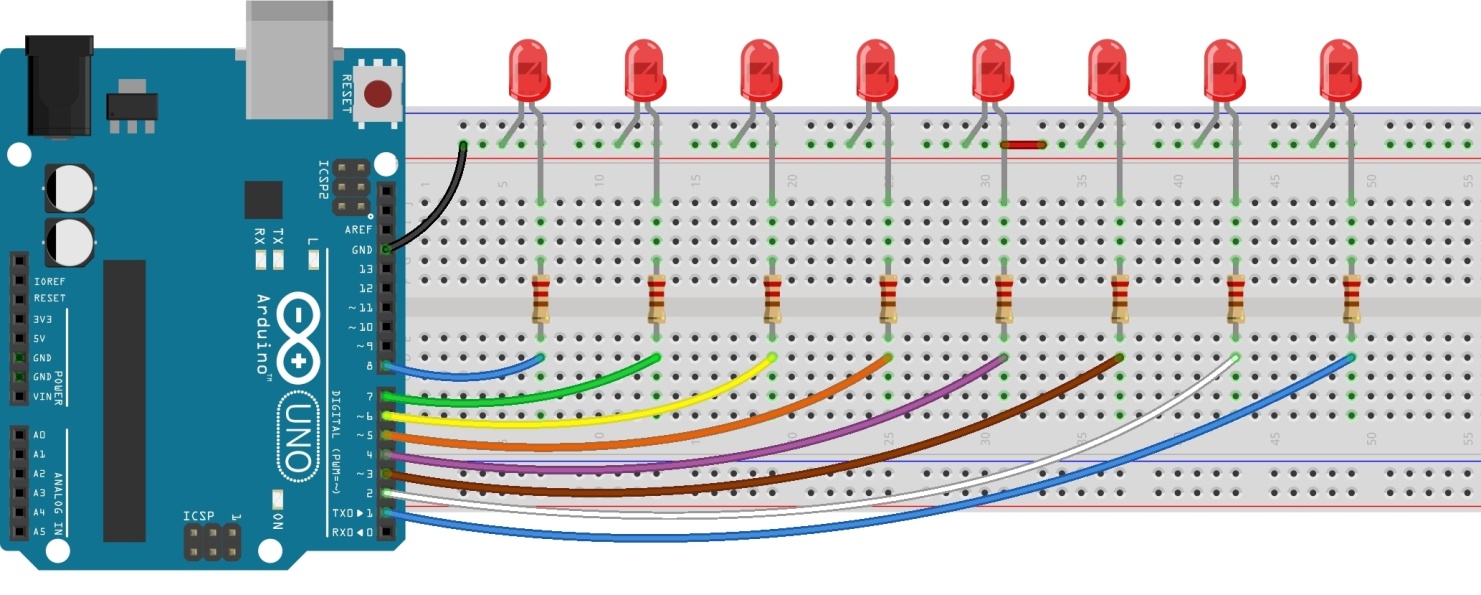
}

# Experiment 2

## Chaser :

This program is used to blink multiple LEDs one by one. It is commonly known as Chaser. Here we are using eight LEDs. Each LED will blink one after another. Eight LEDs are mounted on breadboard along with resistors. Anode of each led is given to Arduino pin through resistor of 220 Ohm. Cathode of each LED is connected to Ground.

## Circuit:



Same chaser logic can be performed by more than one method in first multiple write functions are used in single program. In another method For loop is used for chaser programming which reduces size of program.

### Pin Connections to Arduino:

* Connect Anode(+) terminal of all eight LEDs to pins varying from 1 to 8 through 1K resistor
* Connect cathode(-) of all LED to Ground(0V)

## Program:

int led1 = 1;

int led2 = 2;

int led3 = 3;

int led4 = 4;

int led5 = 5;

int led6 = 6;

int led7 = 7;

int led8 = 8;

void setup()

{

pinMode(1,OUTPUT);

pinMode(2,OUTPUT);

pinMode(3,OUTPUT);

pinMode(4,OUTPUT);

pinMode(5,OUTPUT);

pinMode(6,OUTPUT);

pinMode(7,OUTPUT);

pinMode(8,OUTPUT);

}

void loop()

{

digitalWrite(led1,HIGH); delay(100);

digitalWrite(led1,LOW); delay(100);

digitalWrite(led2,HIGH); delay(100);

digitalWrite(led2,LOW); delay(100);

digitalWrite(led3,HIGH); delay(100);

digitalWrite(led3,LOW); delay(100);

digitalWrite(led4,HIGH); delay(100);

digitalWrite(led4,LOW); delay(100);

digitalWrite(led5,HIGH); delay(100);

digitalWrite(led5,LOW); delay(100);

digitalWrite(led6,HIGH); delay(100);

digitalWrite(led6,LOW); delay(100);

digitalWrite(led7,HIGH); delay(100);

digitalWrite(led7,LOW); delay(100);

digitalWrite(led8,HIGH); delay(100);

digitalWrite(led8,LOW); delay(100);

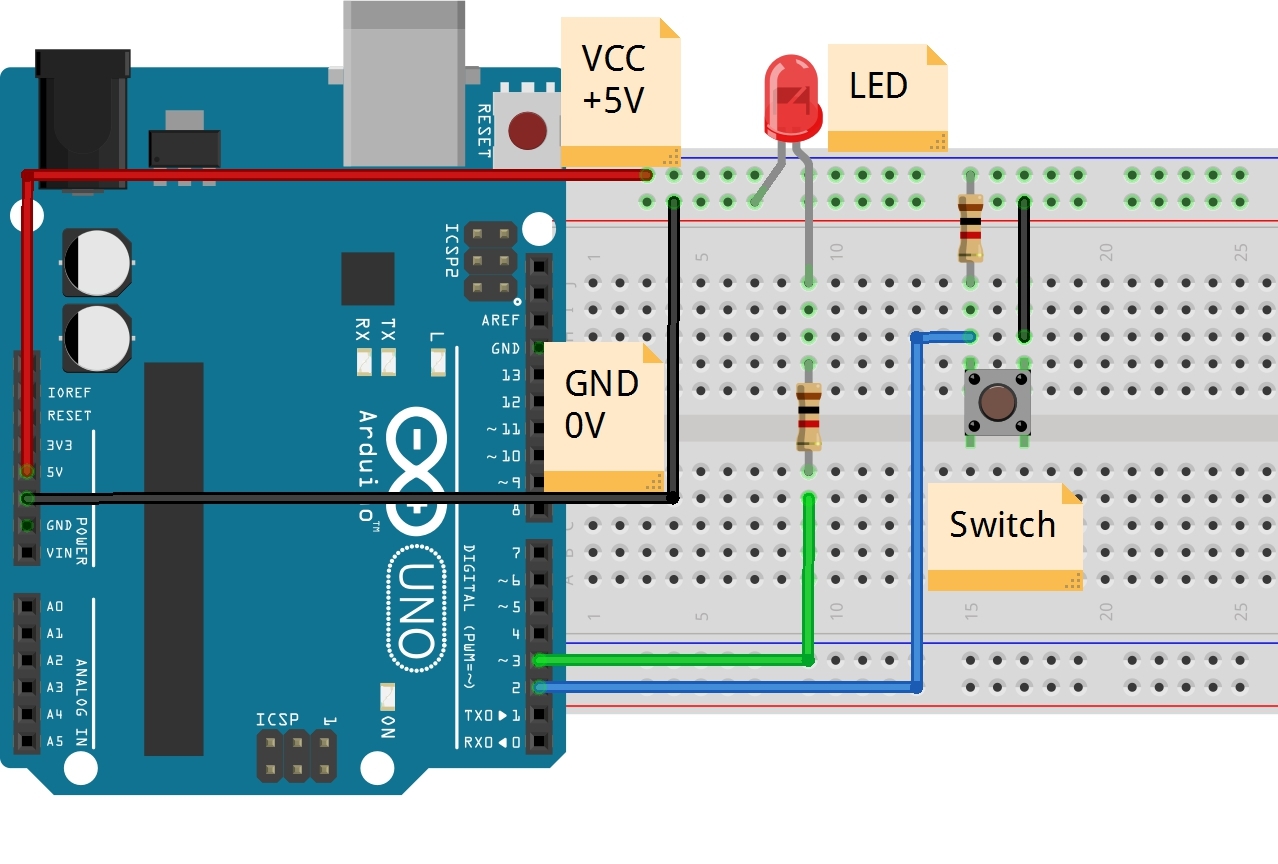
}

# Experiment 3

## Single Input :

In previous experiments we turned ON / OFF LEDs with the help of programming only. Here along with programming we will use one switch as INPUT. This will turn ON /OFF LED according to the program.

## Circuit:



In this experiment we will require one LED and one Switch. Here Switch is used as INPUT and LED is used as OUTPUT. Whenever switch is pressed LED should turn ON and when switch is released it should turn OFF. Switch is connected to pin 2 of Arduino and LEd is connected to pin 3 of Arduino.

## Led Connection:

Anode: To pin 3 of Arduino

Cathode: To Ground

## Switch connection:

Connect Pin2 of Arduino to one of the terminal of switch and through 1K resistor connect same pin to VCC

**Program:**

# const int buttonPin = 2; // the number of the pushbutton pin

# const int ledPin = 3; // the number of the LED pin

# // variables will change:

# int buttonState = 0; // variable for reading the pushbutton status

# void setup()

# {

# pinMode(ledPin, OUTPUT);

# pinMode(buttonPin, INPUT);

# }

# void loop()

# {

# // read the state of the pushbutton value:

# buttonState = digitalRead(buttonPin);

# // check if the pushbutton is pressed.

# // if it is, the buttonState is HIGH:

# if (buttonState == LOW)

# {

# // turn LED on:

# digitalWrite(ledPin, HIGH);

# }

# 

# else

# {

# // turn LED off:

# digitalWrite(ledPin, LOW);

# }

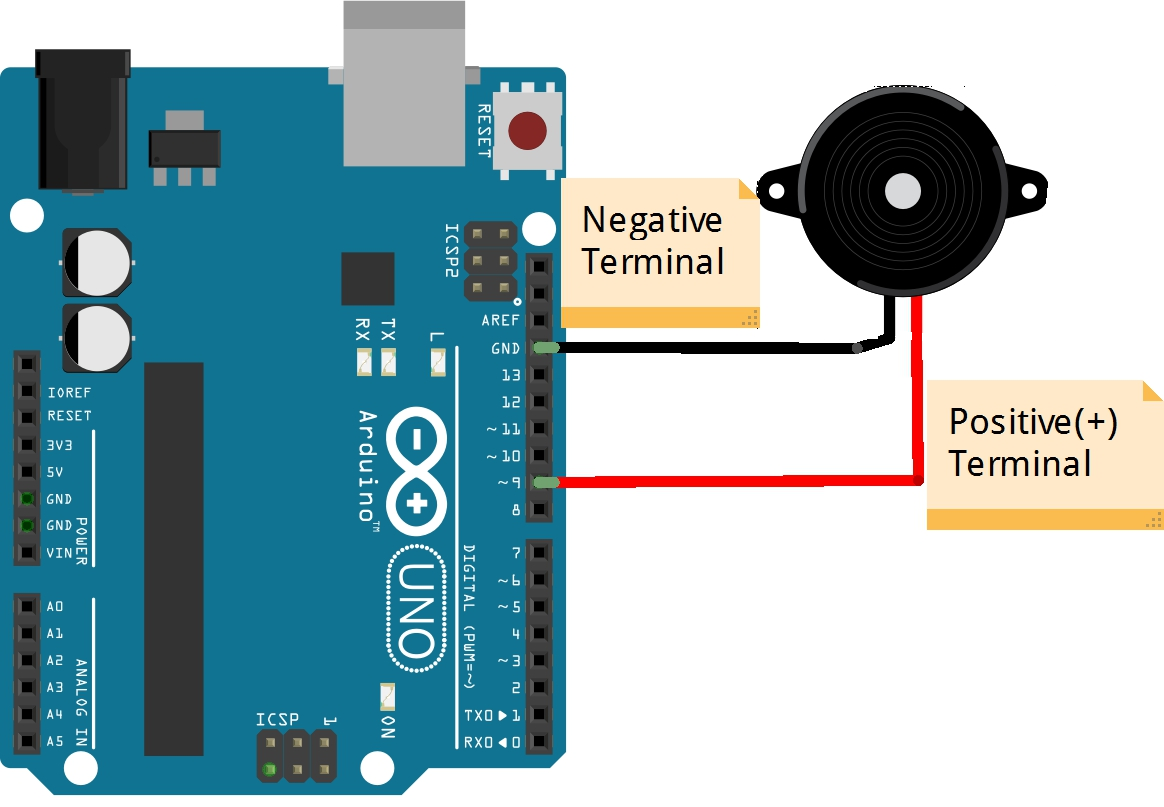
}

# Experiment 4

## Buzzer:

## A buzzer or beeper is an audio signaling device,which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.

## Circuit:



While interfacing Buzzer with Arduino we can do several experiments on sound melody. We can turn On and OFF buzzer using Blink logic. We can generate different melody signals also.

### Buzzer Connections:

* Positive Terminal of Buzzer: Pin 9 of Arduino
* Negative Terminal of Buzzer: Ground

## Program:

# void setup()

# {

# }

# void loop() {

# // turn off tone function for pin 9:

# noTone(9);

# // play a 440Hz frequency on pin 6 for 2000 ms:

# tone(9, 440, 200);

# delay(2000);

# noTone(9); // now tone is stopped

# // turn off tone function for pin 6:

# delay(2000);

# // now we'll use a for loop to give wide range of

# // freuquencies to buzzer for small durations

# // this will go longer as we'll hear sounds of each

# // frequencies for 500 msecs

# for(int i = 20; i < 2000; i++) //

# {

# tone(9,i);

# delay(5);

# noTone(9);

# }

# noTone(9);

# delay(1000); // we stop here

# }

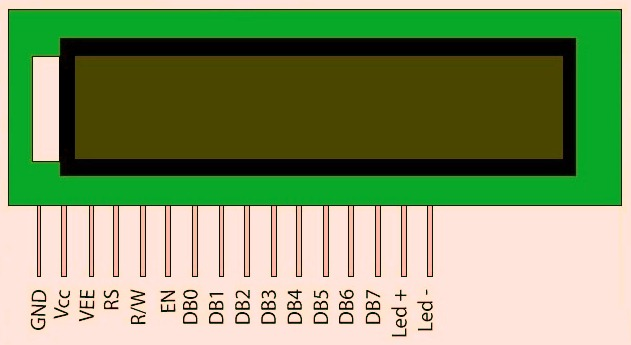
# Experiment 5

## Liquid Crystal Display:

Liquid Crystal Display is used to display information in its rows and columns. The information is in the form of letters, Numbers, special characters or graphics. Depending upon type it displays Alphabets ,numbers or Graphics.

In this experiment we are using 16x2 Alpha numeric LCD which can display letters (A-Z, a-z), Numbers(0-9),and some special Characters . It has 16 pins. Each pin performs different Functions.

### Pin Diagram:



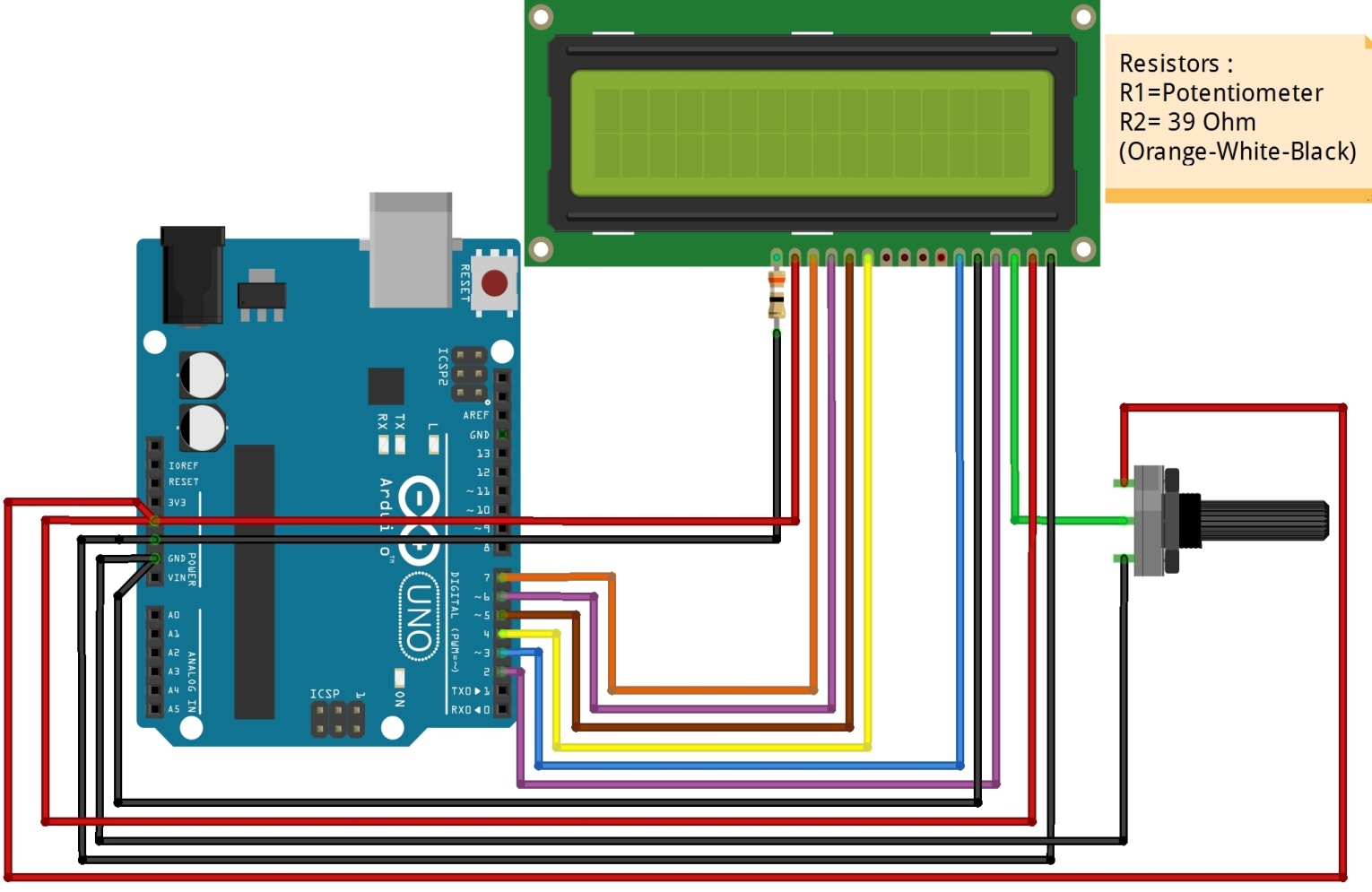
### Pin Description:

|  |  |  |
| --- | --- | --- |
| **Pin** | **Symbol** | **Description** |
| 1 | VSS | Ground |
| 2 | VCC | +5V |
| 3 | VEE | Power Supply to control Contrast |
| 4 | RS | RS=0, Select Command Register  RS=1, Select Data Register |
| 5 | R/W | R/W=0, For Write R/W=1, For Read |
| 6 | EN | Enable |
| 7 | D0 | Data Pin 0 |
| 8 | D1 | Data Pin 1 |
| 9 | D2 | Data Pin 2 |
| 10 | D3 | Data Pin 3 |
| 11 | D4 | Data Pin 4 |
| 12 | D5 | Data Pin 5 |
| 13 | D6 | Data Pin 6 |
| 14 | D7 | Data Pin 7 |

While interfacing LCD with Arduino connect above mentioned pin to proper pin.In this experiment we are displaying information on LCD Screen using Arduino programming. For Data transfer use upper 4 bit pins.This will save some Arduino pins and will reduce complexity of programming also.

Pin 15 and 16 pin of Alphanumeric LCD are used for backlight control Connect pin 15 to +5V and pin 16 to Ground through 39 Ohm Resistance.

## Circuit:



### 16x2 LCD Connections:

* Pin 1: VSS : To Ground
* Pin 2: VCC : +5v
* Pin 3: VEE : Connect middle wire of potentiometer to VEE.
* Pin 4: RS : To Digital Pin 2 of Arduino
* Pin 5: R/W : To Ground
* Pin 6: EN : To Digital Pin 3 of Arduino
* Pin 11: D4 : To digital pin 4 of Arduino
* Pin 12: D5 : To digital pin 5of Arduino
* Pin 13: D6 : To digital pin 6 of Arduino
* Pin 14: D7 : To digital pin 7 of Arduino

## Program:

#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(2,3,4,5,6,7);

void setup()

{

// set up the LCD's number of columns and rows:

lcd.begin(16, 2);

// Print a message to the LCD.

lcd.print("e-Vidya");

}

void loop()

{

// set the cursor to column 0, line 1

// (note: line 1 is the second row, since counting begins with 0):

// so lcd.setCursor works as lcd.setCursor(coloumn no, Row no);

// where coloumn no can be 0-15 and row no can be 0-1

lcd.setCursor(0, 1);

// print the number of seconds since program started

lcd.print(millis() / 1000);

}

# Experiment 6

## DC motor and L293 Driver:

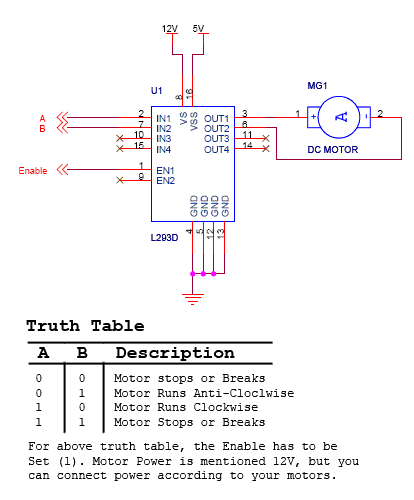
Whenever a robotics hobbyist talk about making a robot, the first thing comes to his mind is making the robot move on the ground. And there are always two options in front of the designer whether to use a DC motor or a stepper motor. When it comes to speed, weight, size, cost... DC motors are always preferred over stepper motors.

There are many things which you can do with your DC motor when interfaced with a microcontroller. For example you can control the speed of motor, you can control the direction of rotation.Usually H-bridge is preffered way of interfacing a DC motor.

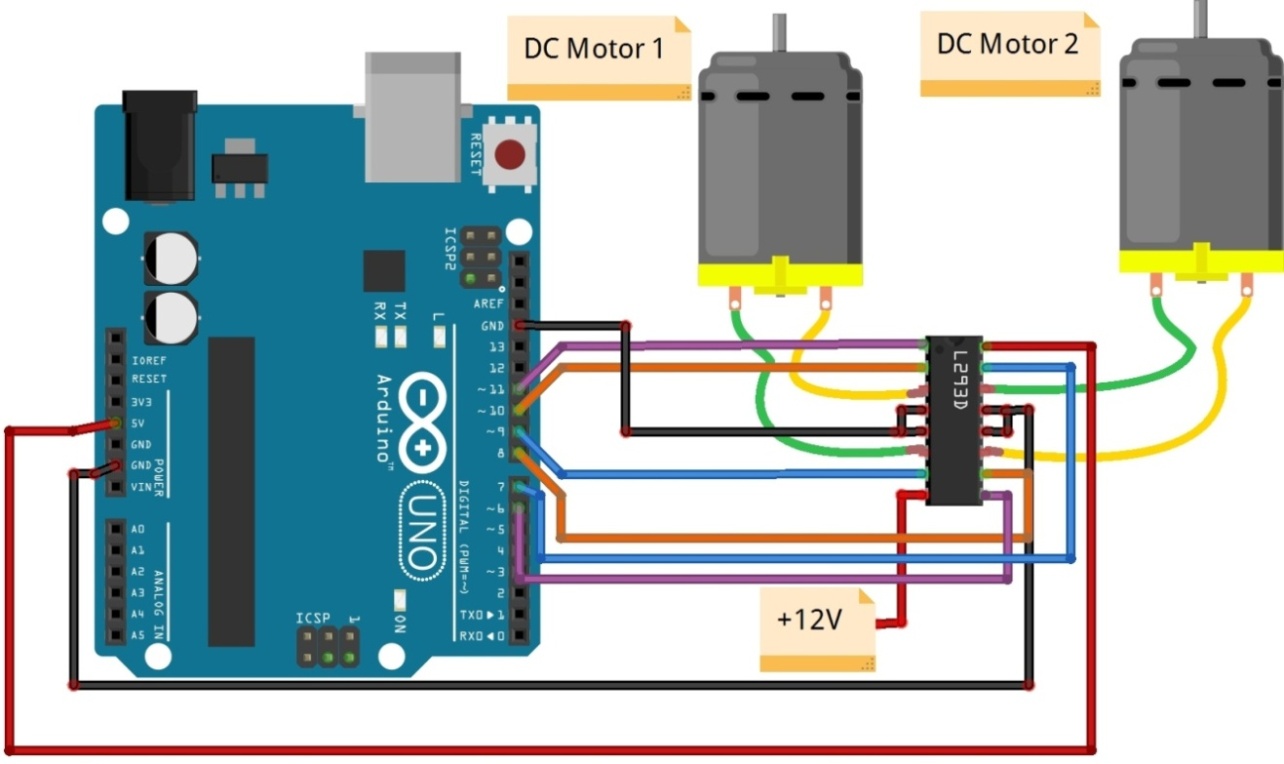
These days many IC manufacturers have H-bridge motor drivers available in the market like L293D is most used H-Bridge driver IC. l293D is a dual H-Bridge motor driver, So with one IC we can interface two DC motors which can be controlled in both clockwise and counter clockwise direction.

L293D has output current of 600mA and peak output current of 1.2A per channel. Moreover for protection of circuit from back EMF ouput diodes are included within the IC. The output supply (VCC2) has a wide range from 4.5V to 36V, which has made L293D a best choice for DC motor driver.

## L293D and DC Motor Interfacing:



## Circuit:



## L293D Connections:

* Pin 1 (EN1) : To digital Pin 11 of Arduino
* Pin 2(IN1) : To digital Pin 10 of Arduino
* Pin 3 : To DC Motor 1
* Pin 4 & Pin 5 : Ground
* Pin 6 : To DC Motor 1
* Pin 7 :To DC Motor 1
* Pin 8 : To +12 V
* Pin 9(EN2) : To digital Pin 6 of Arduino
* Pin 10(IN3) : To digital Pin 8 of Arduino
* Pin 11 : To DC Motor 2
* Pin 12 & Pin 13 : Ground
* Pin 14 : To DC Motor 2
* Pin 15(IN4) : To digital Pin 7 of Arduino
* Pin 16 :+5V

# Experiment 7

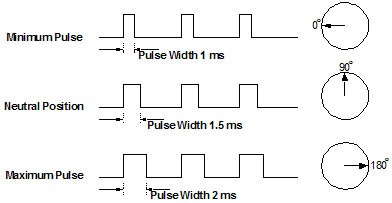
## Servo Motor:

Servos are [DC motors](http://www.societyofrobots.com/actuators_dcmotors.shtml) with built in gearing and feedback control loop circuitry. Most servo motors can rotate about 90 to 180 degrees. Some rotate through a full 360 degrees or more. However, servos are unable to continually rotate, meaning they can't be used for driving wheels (unless [modified](http://www.societyofrobots.com/actuators_modifyservo.shtml)), but their precision positioning makes them ideal for [robot arms](http://www.societyofrobots.com/robot_arm_tutorial.shtml). Since servos are fully self contained, the velocity and angle control loops are very easy to impliment, while prices remain very affordable.

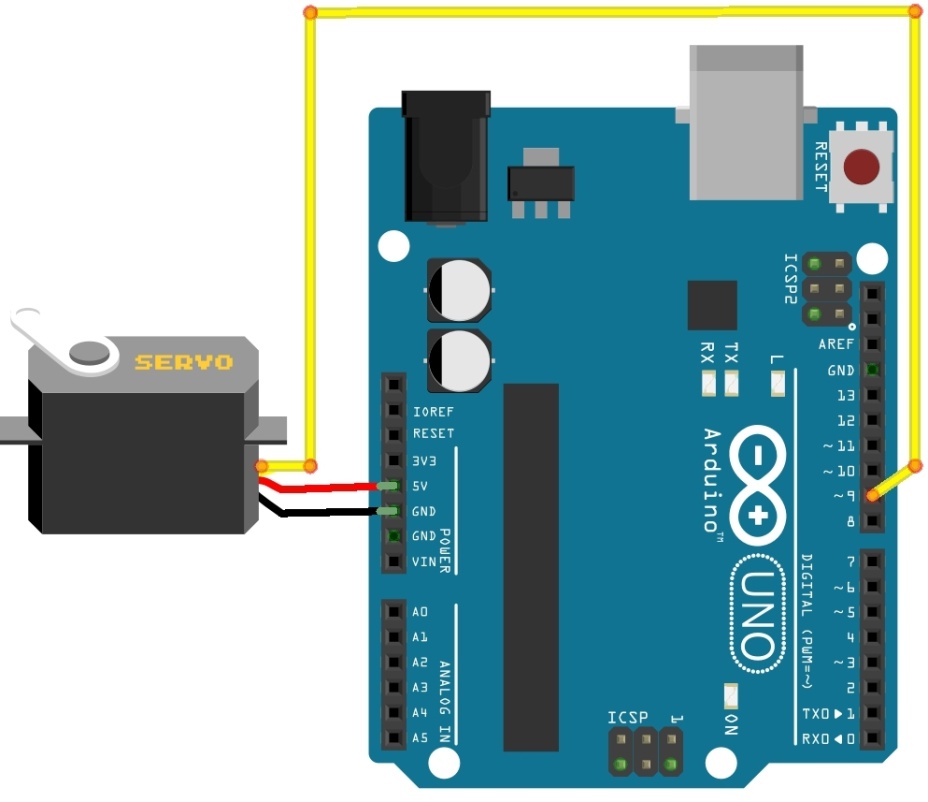
To use a servo, simply connect the black wire to ground, the red to a 4.8-6V source, and the yellow/white wire to a signal generator (such as from your [microcontroller](http://www.societyofrobots.com/microcontroller_tutorial.shtml)). Vary the square wave pulse width from 1-2ms and your servo is now position/velocity controlled. Servo Wiring All servos have three wires:   
 1.Black or Brown is for ground.

2.Red is for power(4.8-6V)  
 3.Yellow, Orange, or White is the signal wire (3-5V).

The standard **time vs. angle** is represented in this chart:



## Circuit:



In this experiment we are going to control the angle or we can called it as position of servo motor. To achieve this we have to use library and functions of Servo motor defined in Arduino System. Here we are rotating sevo in between 0 to 180 angle. For that we are using PWM pin 9 of Arduino.

### Servo Motor Connections:

Servo Motor:

* Black Wire : To Ground
* Red Wire : +5V
* OUTPUT(yellow/green/orange/white/brown): To digital Pin 9 of Arduino

## Program:

#include <Servo.h>

Servo myservo;

int pos = 0;

void setup()

{

myservo.attach(9);

}

void loop()

{

for(pos = 0; pos <= 180; pos += 1)

{

myservo.write(pos);

delay(15);

}

for(pos = 180; pos>=0; pos-=1)

{

myservo.write(pos);

delay(15);

}

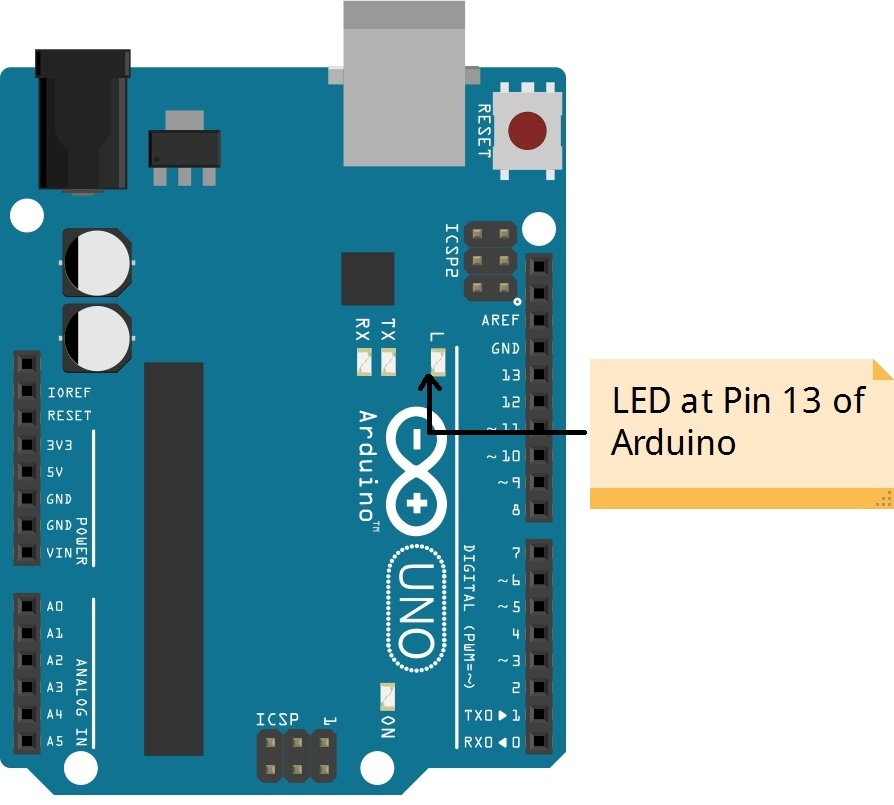
}

# Experiment 8

## Serial Communication:

It is used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): **Serial**. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB.

## Circuit:



In this project PC based Device control takes place. With Arduino platform as mediator we are going to perform this task. Serial communication is used inside this program. When we enter one letter in COM port window which is normally used for serial communication purpose it will turn ON the LED. To turn off LED again we will enter different letter.

## Program:

int led1 = 13;

int rx;

void setup()

{

Serial.begin(9600);

pinMode(led1,OUTPUT);

}

void loop()

{

if(Serial.available() > 0)

{

rx = Serial.read();

if(rx == 'a' || rx == 'A')

{

digitalWrite(led1,HIGH);

}

if(rx == 'w' || rx == 'W')

{

digitalWrite(led1,LOW);

}

}

}

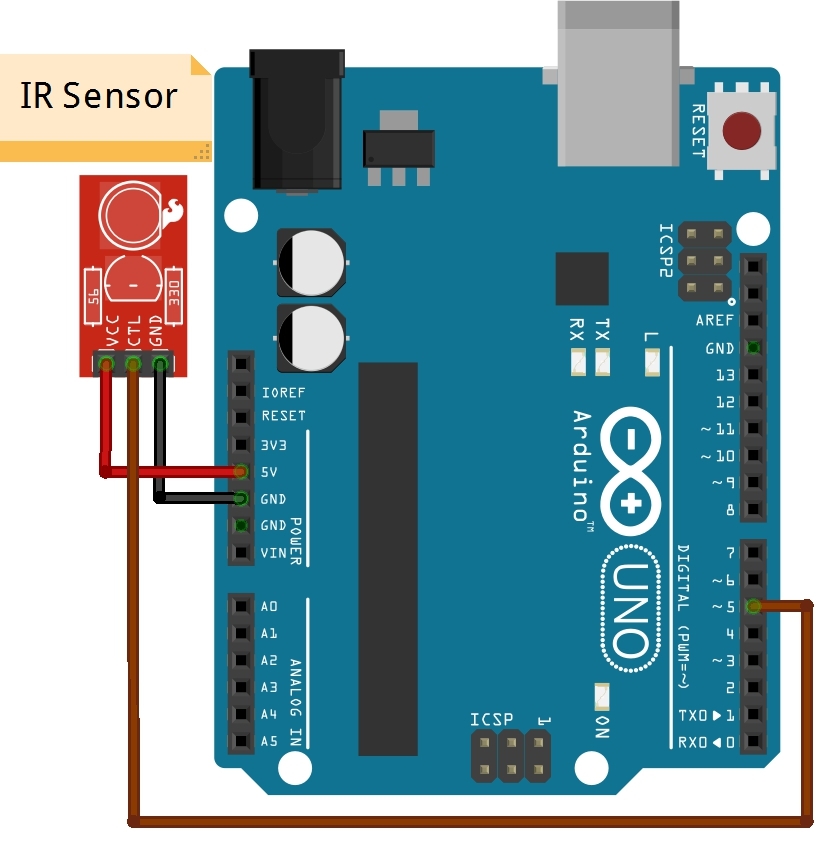
# Experiment 9

## IR Sensor:

IR Sensors work by using a specific light sensor to detect a select light wavelength in the Infra-Red (IR) spectrum. By using an LED which produces light at the same wavelength as what the sensor is looking for, you can look at the intensity of the received light.

When an object is close to the sensor, the light from the LED bounces off the object and into the light sensor. This results in a large jump in the intensity, which we already know can be detected using a threshold.

## Circuit:



In this experiment using IR sensor we have done Automatic Light Controller. IR Sensor is used to detect object(Example :Person). It will count number of interrupts. When count is ‘0’ it will turn of LED. If count is greater than 0 it will turn On the LED.

### IR Connections:

* VCC of Sensor : +5V
* Gnd of Sensor: Ground
* Output :To Digital pin 5 of Arduino

## Program:

int count=0;

int IR;

void setup()

{

// put your setup code here, to run once:

pinMode(8,INPUT);

pinMode(13,OUTPUT);

Serial.begin(9600);

}

void loop()

{

// put your main code here, to run repeatedly:

IR=digitalRead(8);

if (IR==LOW)

{

count++;

Serial.println(count);

delay(100);

if(count >0)

{

digitalWrite(13,HIGH);

}

else

{

digitalWrite(13,LOW);

}

}

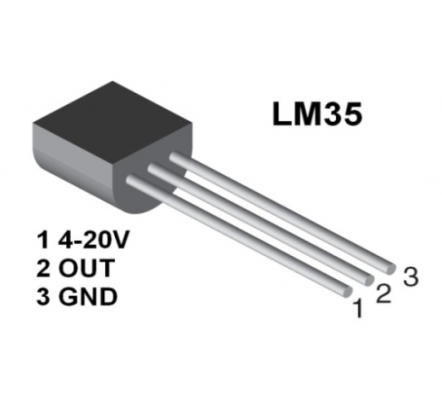
}

# Experiment 10

## LM 35:

The LM35 is an integrated circuit sensor that can be used to measure temperature with an electrical output proportional to the temperature (in **o**C).The scale factor is 10mV/**o**C.

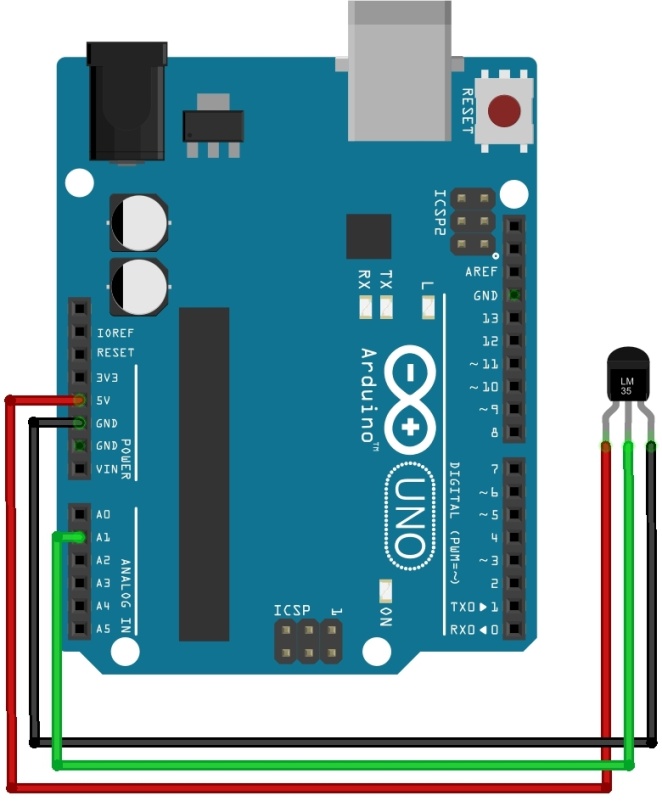
## Pin Description:



In this experiment we are going to display temperature in serial window. To achieve this we are using one electronics circuit which is commonly known as temperature sensor LM35. LM35 senses the temperature generates its response at its pin 2. This output is analog in nature. To converts analog into digital one connect output pin of LM35 to ADC pin of Arduino A1.

Use proper conversion formula for LM35 and display all values in Serial window using Serial Communication.

## Circuit:



## LM 35 Connections:

* Pin 1: VS - +5V
* Pin 2 : OUT –Analog pin A1 of Arduino
* Pin 3: Ground

## Program:

int temp;

int rx;

void setup()

{  
pinMode(A1,INPUT);

}

void loop()

{

rx = analogRead(A1);

temp = (rx \* 4.88) / 10; // conversion to temp

Serial.println(temp); // Display Values

delay(300);

}

# Experiment 11

## Ultrasonic Sensor:

The Ultrasonic module has two Transducers ,one for  Transmit & the other for Receive. Ultrasound is a high frequency sound of  frequency 40 KHz (we can’t hear this frequency, but animals can).

## Principle of operation:

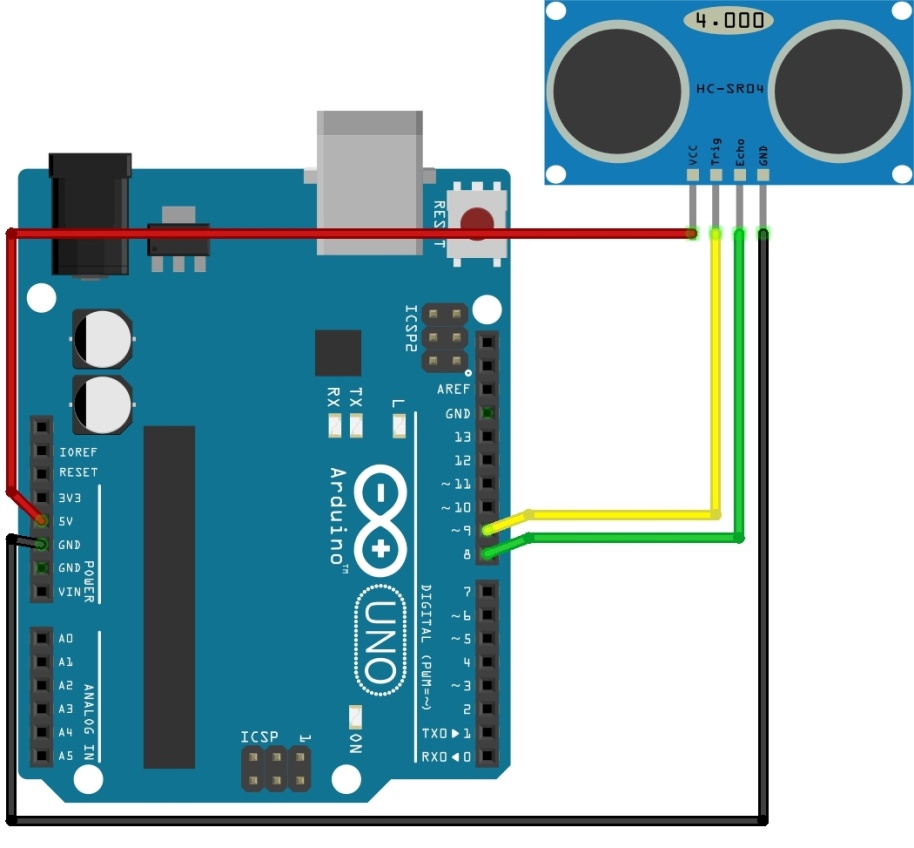
Generate a  short 10uS pulse to the **Trigger input** to start the ranging. The Ultrasonic Module  will send out an 8 cycle burst of ultrasound at 40khz **and raise its echo line high.** It then listens for an echo, and **as soon as it detects one it lowers the echo line again.** The echo line is therefore a pulse whose width is proportional to the distance to the object. By timing the pulse it is possible to calculate the range in inches/centimeters .

If nothing is detected then the module will lower its echo line anyway after about 30mS. The module provides an echo pulse proportional to distance. If the width of the pulse is measured in uS, then dividing by 58 will give you the distance in cm, or dividing by 148 will give the distance in inches.

The module can be triggered as fast as every 50ms, or 20 times each second. You should wait 50ms before the next trigger, even if the SRF05 detects a close object and the echo pulse is shorter. This is to ensure the ultrasonic "beep" has faded away and will not cause a false echo on the next ranging.

The sensor can detect objects within **3cm to 3m range.** In this experiment we are going to measure as well as display the distance between obstacle and Sensor using Arduino and Serial Communication.

## Circuit:



## HC SR-05 Connections:

* VCC : Ground
* Pin 2: VS - +5V
* Pin 7: Trigger Pin
* Pin 8: Echo Pin

## Program:

int trigPin = 7;

int echoPin = 8;

void setup()

{

Serial.begin(9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop()

{

long duration, inches, cm;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

inches = microsecondsToInches(duration);

cm = microsecondsToCentimeters(duration);

Serial.print(inches);

Serial.print("in, ");

Serial.print(cm);

Serial.print("cm");

Serial.println();

delay(100);

}

long microsecondsToInches(long microseconds)

{

return microseconds / 74 / 2;

}

long microsecondsToCentimeters(long microseconds)

{

return microseconds / 29 / 2;

}

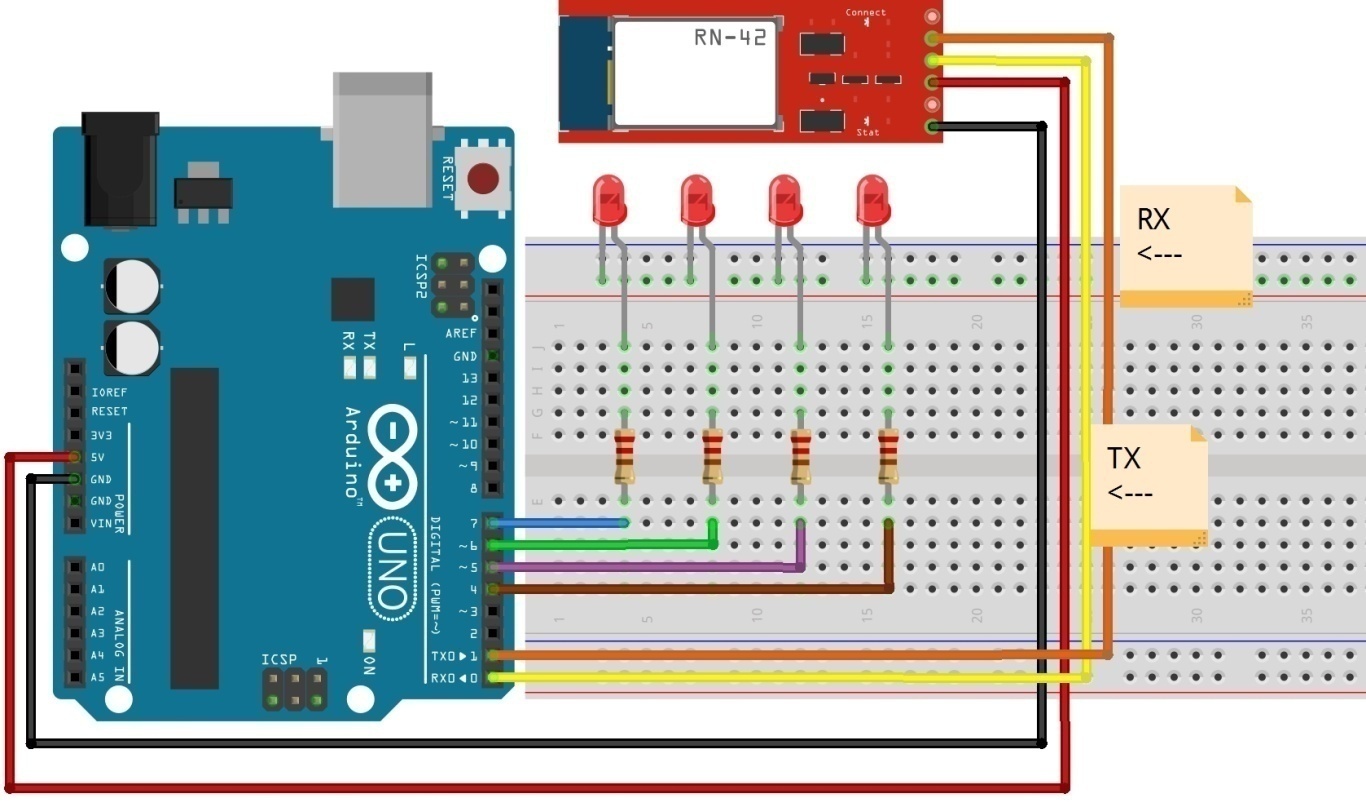
# Experiment 12

## Bluetooth Module:

Bluetooth is a [wireless](https://en.wikipedia.org/wiki/Wireless) technology standard for exchanging data over short distances (using short-wavelength [UHF](https://en.wikipedia.org/wiki/UHF) [radio waves](https://en.wikipedia.org/wiki/Radio_waves) in the [ISM band](https://en.wikipedia.org/wiki/ISM_band) from 2.4 to 2.485 GHz) from fixed and mobile devices.

In this we are using HC-05 is a class-2 bluetooth module with Serial Port Profile, which can configure as either Master or slave. You can use it simply for a serial port replacement to establish connection between MCU, PC to your embedded project and etc. These modems work as a serial (RX/TX) pipe. Any serial stream from 9600 to 115200bps can be passed.

## Circuit:



In this experiment we are turning ON/OFF devices using Commands send by Bluetooth.

## Bluetooth Connections:

* Ground : To Ground pin of Arduino
* VCC :+5V
* TX :To Pin 0 of Arduino
* RX :To pin Pin 1of Arduino
* Connect anode of four LEDs to Digital pins of Arduino(4 to 7) through Resistor
* Connect cathode of all LEDs to Ground

## Program:

void setup()

{

Serial.begin(9600); //Initialise Serial Communication

pinMode(4,OUTPUT);

pinMode(5,OUTPUT);

pinMode(6,OUTPUT);

pinMode(7,OUTPUT);

}

void loop()

{

// here we'll check if some data is received or not

if(Serial.available()>0)

{

int rxd=Serial.read();

if(rxd=='U')

{

digitalWrite(4,HIGH);

}

if(rxd=='D')

{

digitalWrite(5,HIGH);

}

if(rxd=='L')

{

digitalWrite(6,HIGH);

}

if(rxd=='R')

{

digitalWrite(7,HIGH);

}

if(rxd=='C')

{

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,LOW);

digitalWrite(7,LOW);

}

}

}