NIE

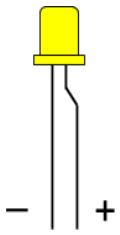
The National Institute of Engineering

IoT Lab Manual

Prepared By
K R Sumana
Assistant Professor
Department of MCA, NIE, Mysuru

Interfacing LED with Arduino

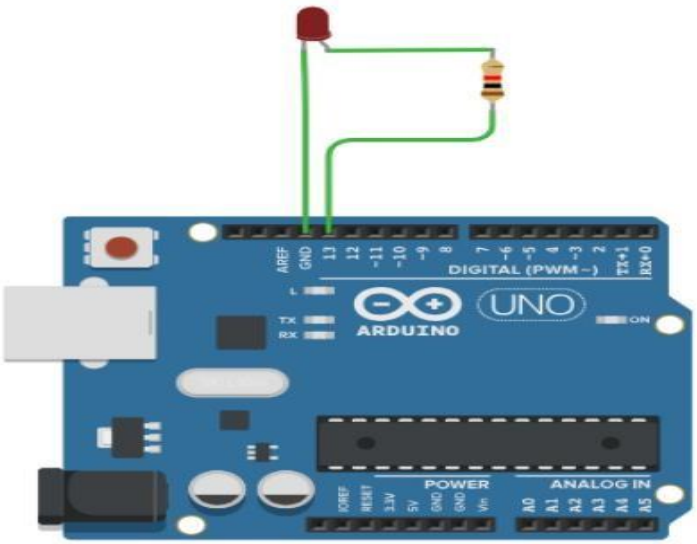
LEDs (light-emitting diodes) are small, bright, power-efficient lights commonly used in electronic products. An LED light is a polarized part, meaning it has to be connected to a circuit in a certain way to work properly. Specifically, each LED has a positive leg and a negative leg. These can be identified visually by length: the negative leg has been made shorter.



Hardware Requirements

- Arduino UNO
- LEDs
- Resistor-220ohms
- Jump wires

Circuit



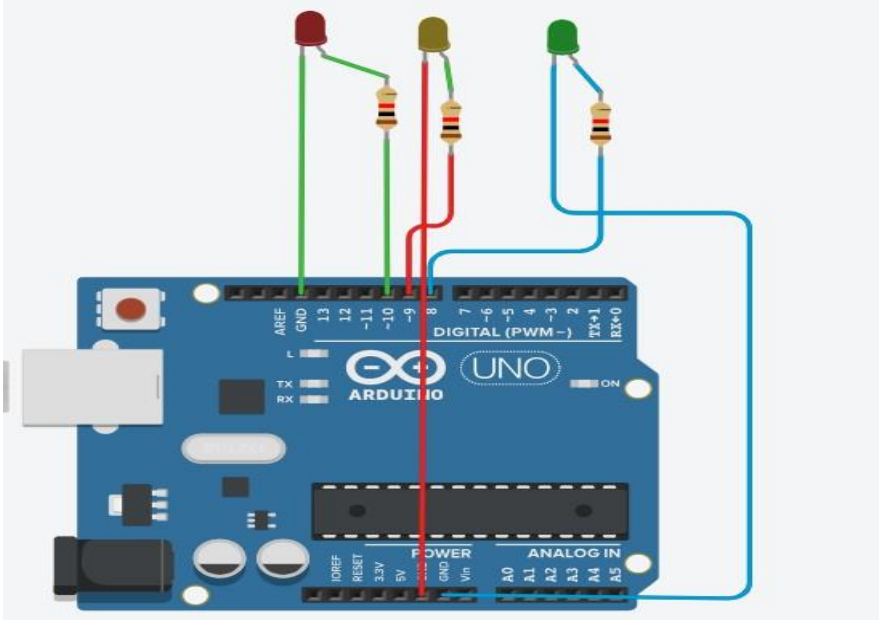
Code

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  // turn the LED off by making the voltage LOW
```

```
  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Traffic light with LED interfacing using Arduino UNO

Circuit

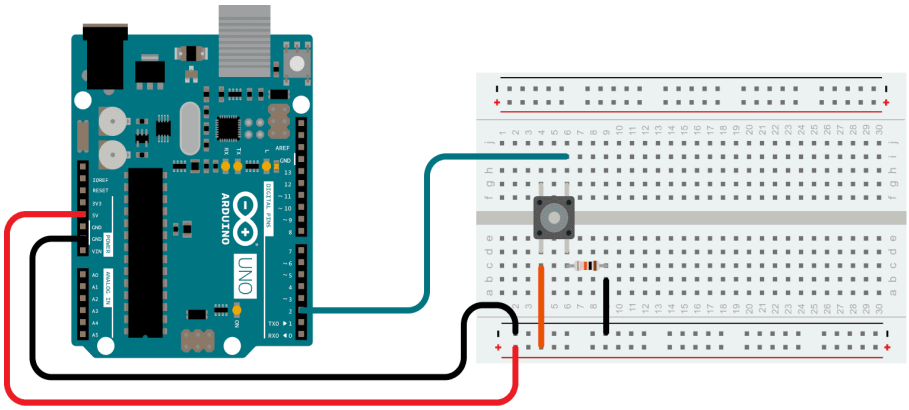


Code

```
void setup()
{
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop()
{
  // turn the RED-LED on (HIGH is the voltage level)
  digitalWrite(10, HIGH);
  delay(2000); // Wait for 1000 millisecond(s)
  // turn the RED-LED off by making the voltage LOW
  digitalWrite(10, LOW);
  delay(1000); // Wait for 1000 millisecond(s)

  // turn the YELLOW-LED on (HIGH is the voltage level)
  digitalWrite(9, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  // turn the YELLOW-LED off by making the voltage LOW
  digitalWrite(9, LOW);
  delay(1000); // Wait for 1000 millisecond(s)

  // turn the GREEN-LED on (HIGH is the voltage level)
```

```
  digitalWrite(8, HIGH);
  delay(2000); // Wait for 1000 millisecond(s)
  // turn the GREEN-LED off by making the voltage LOW
  digitalWrite(8, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

LED with pushbutton using Arduino UNO

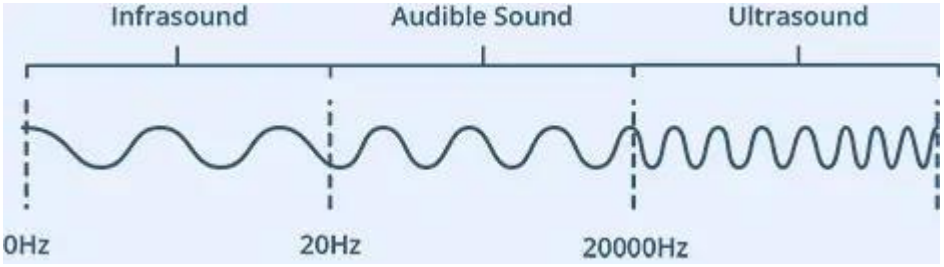


Code

```
const int buttonPin = 2;  // the number of the pushbutton pin
const int ledPin = 13;    // the number of the LED pin

// variables will change:
int buttonState = 0;  // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  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 == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

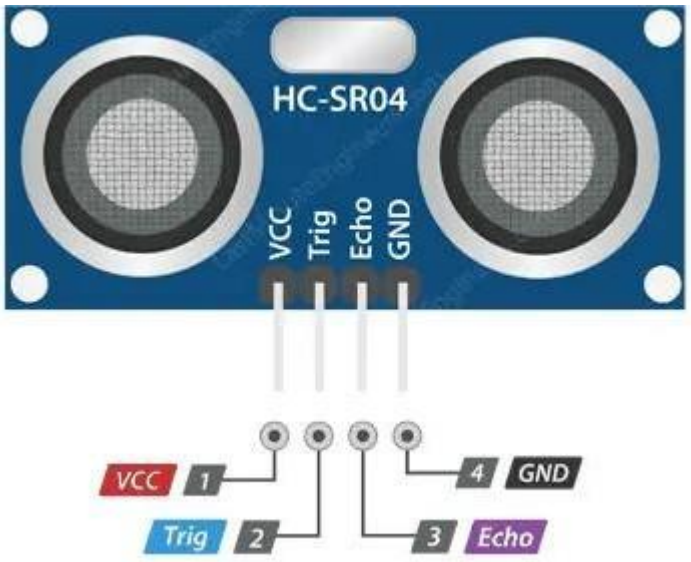
Interfacing Ultrasonic Sensor With Arduino

The HC-SR04 Ultrasonic Distance Sensor that can report the range of objects up to 13 feet away.



Ultrasound is a high-pitched sound wave whose frequency exceeds the audible range of human hearing.

Humans can hear sound waves that vibrate in the range of about 20 times a second (a deep rumbling noise) to 20,000 times a second (a high-pitched whistle). However, ultrasound has a frequency of more than 20,000 Hz and is therefore inaudible to humans.



It detects the distance of the closest object in front of the sensor (from 3 cm up to 400 cm). It works by sending out a burst of ultrasound and listening for the echo when it bounces off of an object. It *pings* the obstacles with ultrasound.

The Arduino board sends a short pulse to trigger the detection, then listens for a pulse on the same pin using the **pulseIn()** function. The duration of this second pulse is equal to the time taken by the ultrasound to travel to the object and back to the sensor. Using the speed of sound, this time can be converted to distance.

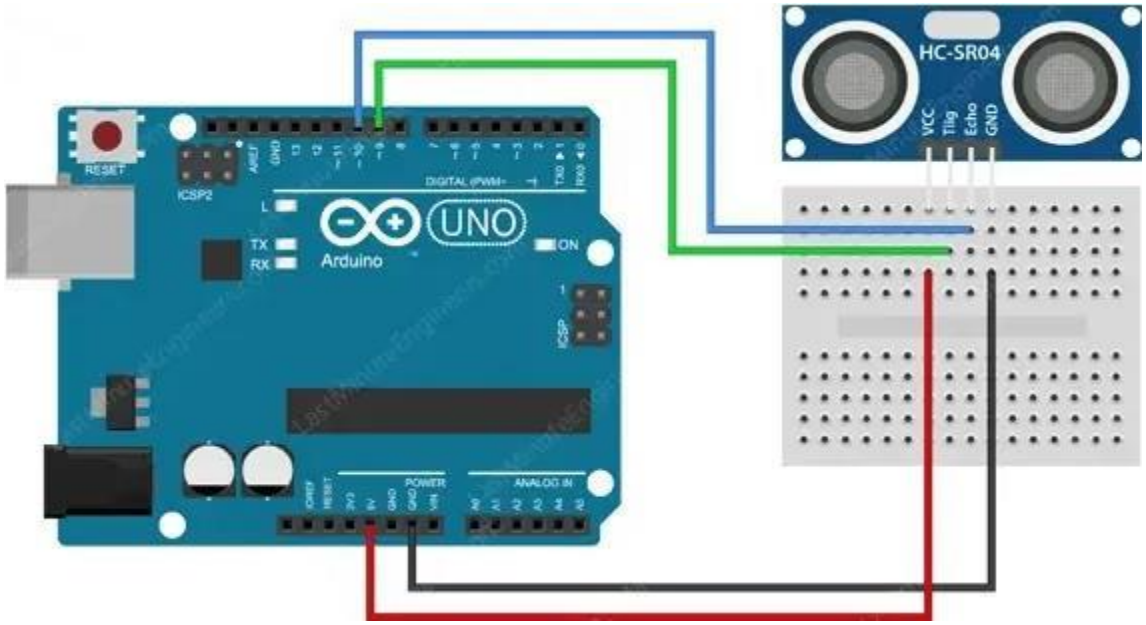
Hardware Required

- Arduino Board
- Ultrasonic Sensor (HC-SR04)

- Jump wires

Circuit

The 5V pin of the S HC-SR04 is connected to the 5V pin on the board, the GND pin is connected to the GND pin, the Triger/Ping pin is connected to digital pin 7 and Echo pin to digital pin 8 on the board.



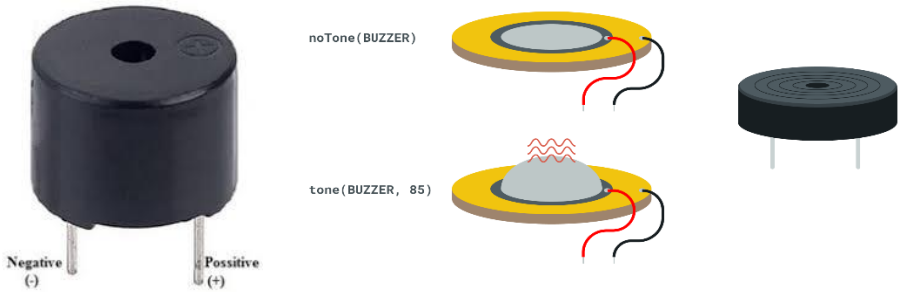
Code

```
 /*The circuit:
 - +Vcc attached to +5V
 - GND attached to ground
 - TRIG attached to digital pin 7
 - ECHO attached to digital pin 8*/

// this constant won't change. It's the pin number of the sensor's output:
const int pingPin = 7;
const int echoPin = 8;

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  pinMode(trigPin,OUTPUT);
  pinMode(echoPin,INPUT);
}

void loop() {
  // establish variables for duration of the ping, and the distance result
  // in inches and centimeters:
  long duration, inches, cm;
```

```
  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // The same pin is used to read the signal from the PING))): a HIGH pulse
  // whose duration is the time (in microseconds) from the sending of the ping
  // to the reception of its echo off of an object.

  duration = pulseIn(echoPin, HIGH);

  // convert the time into a distance
  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) {
  // According to Parallax's datasheet for the PING))), there are 73.746
  // microseconds per inch (i.e. sound travels at 1130 feet per second).
  // This gives the distance travelled by the ping, outbound and return,
  // so we divide by 2 to get the distance of the obstacle.

  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the object we
  // take half of the distance travelled.
  return microseconds / 29 / 2;
}
```

Interfacing Buzzer with Arduino

The piezo, also known as the buzzer, is a component that is used for generating sound. It is a digital component that can be connected to digital outputs, and emits a tone when the output is HIGH. Alternatively, it can be connected to an analog pulse-width modulation output to generate various tones and effects. It operates at both 3.3V and 5V with a sound output of 85 decibels. This module can be used to provide sound feedback to your application



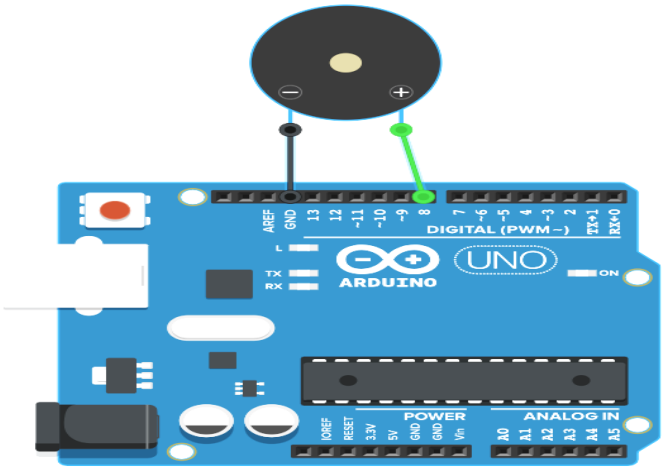
Inside the buzzer there is a membrane, like a drumhead, that vibrates due to electric current. When tone(BUZZER, 85) is executed, the current passes through the membrane and deforms it. For noTone(BUZZER) the current stops and the membrane goes back to its original shape. When current is turned on and off very quickly, the membrane vibrates back and forth, creating sound waves that are audible. By changing how fast the current is turned on and off, we can control the speed of the vibration. Faster vibrations create higher pitches while slower vibrations create lower pitches.

Hardware Required

- Arduino Board
- Buzzer/Piezo sensor
- Jump wires

Circuit

Negative terminal is connected to GND and Positive terminal to digital pin 8.



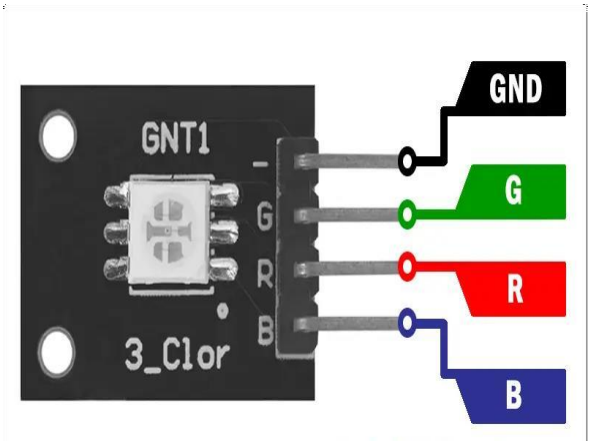
Code

```
//Simple Buzzer

#define BUZZER 18

void setup()
{
  // put your setup code here, to run once:
  pinMode(BUZZER, OUTPUT);
}

void loop()
{
  tone(BUZZER, 85); //Set the voltage to high and makes a noise
  delay(1000);//Waits for 1000 milliseconds
  noTone(BUZZER);//Sets the voltage to low and makes no noise
  delay(1000);//Waits for 1000 milliseconds
}


//Melody note with buzzer

#define NOTE_C4  262

#define NOTE_G3  196

#define NOTE_A3  220

#define NOTE_B3  247

#define NOTE_C4  262

// notes in the melody:

int melody[] = {

  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4

};


// note durations: 4 = quarter note, 8 = eighth note, etc.:

int noteDurations[] = {

  4, 8, 8, 4, 4, 4, 4, 4

};

void setup() {

  // iterate over the notes of the melody:

  for (int thisNote = 0; thisNote < 8; thisNote++) {
```

```
    // to calculate the note duration, take one second divided by the note type.
   // e.g. quarter note =   1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // The note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
          `
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

Interfacing RGB Full Color LED with Arduino

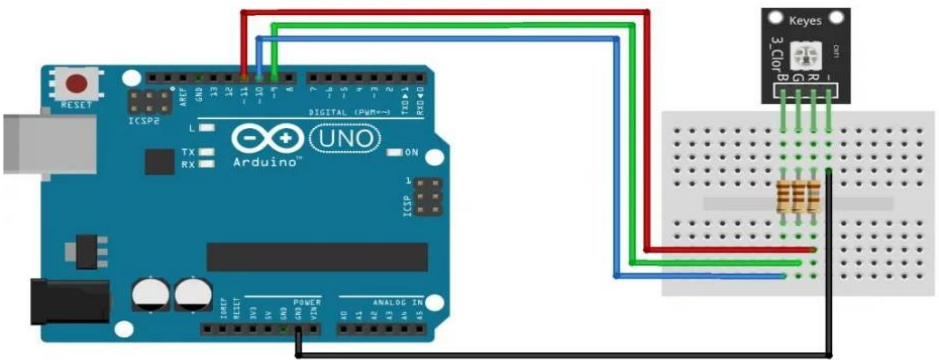
The KY-009 RGB Full Color LED module emits a range of colors by mixing red, green, and blue light. Each color is adjusted by using PWM. Compatible with popular electronics platforms like Arduino, Raspberry Pi and ESP32.



Hardware Requirements

- Arduino Board
- RGB Full Color LED (KY-009)
- Resistors: 180ohms, two resistors of 110ohms
- Jump wires

KY-009	Breadboard	Arduino
R	180Ω resistor	Pin 9
G	110Ω resistor	Pin 10
B	110Ω resistor	Pin 11
–		GND



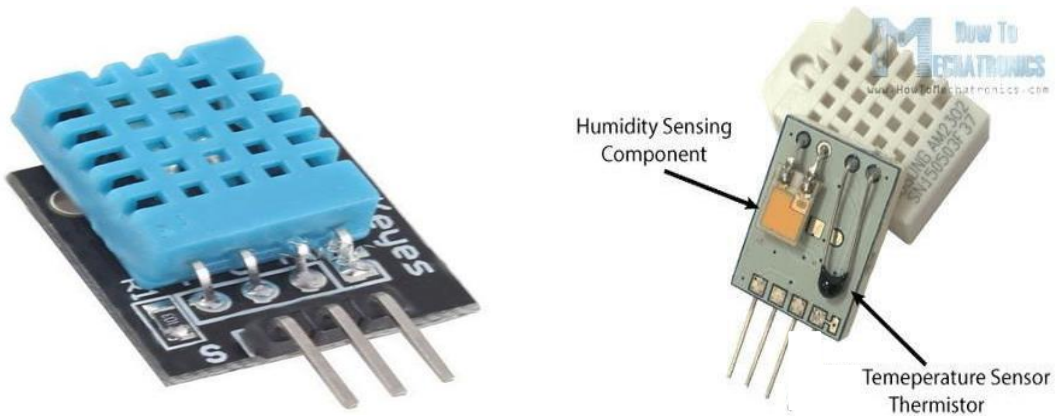
Code

```
int redpin = 11; //select the pin for the red LED
int bluepin =10; // select the pin for the  blue LED
int greenpin = 9;// select the pin for the green LED
int val;
void setup() {
  pinMode(redpin, OUTPUT);
  pinMode(bluepin, OUTPUT);
  pinMode(greenpin, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  for(val = 255; val > 0; val--)
  {
    analogWrite(redpin, val);  //set PWM value for red
    analogWrite(bluepin, 255 - val); //set PWM value for blue
    analogWrite(greenpin, 128 - val); //set PWM value for green
    Serial.println(val); //print current value
    delay(1);
  }
for(val = 0; val < 255; val++)
  {
    analogWrite(redpin, val);
    analogWrite(bluepin, 255 - val);
    analogWrite(greenpin, 128 - val);
    Serial.println(val);
    delay(1);
  }
}
```

Temperature and Humidity Sensor

Humidity is the concentration of water vapor present in the air. Water vapor, the gaseous state of water, is generally invisible to the human eye. Humidity indicates the likelihood for precipitation, dew, or fog to be present. **Temperature** is a physical quantity that quantitatively expresses the attribute of hotness or coldness.

Temperature and humidity sensor is a device that measures the temperature and humidity of the surrounding environment. It can convert the measured values into electrical signals that can be easily read and analysed by other devices. These sensors are commonly used in various applications, such as weather monitoring, indoor climate control, and food storage. There are many types of temperature and humidity sensors available in the market. One such example is the **DHT11 sensor**, which is a commonly used temperature and humidity sensor.



DHT11 Sensor – Temperature and Humidity Sensor

This sensor contains Humidity sensing component and also temperature sensing component which helps in measuring the humidity and temperature. Humidity sensors are devices that measure moisture and air temperature in an environment and convert the data into a corresponding electrical signal. Typically, humidity sensors contain a humidity sensing element and a thermistor, which is used to measure temperature.

Components Required:

- Temperature and Humidity Sensor DHT11
- Arduino Uno
- Jump Wires

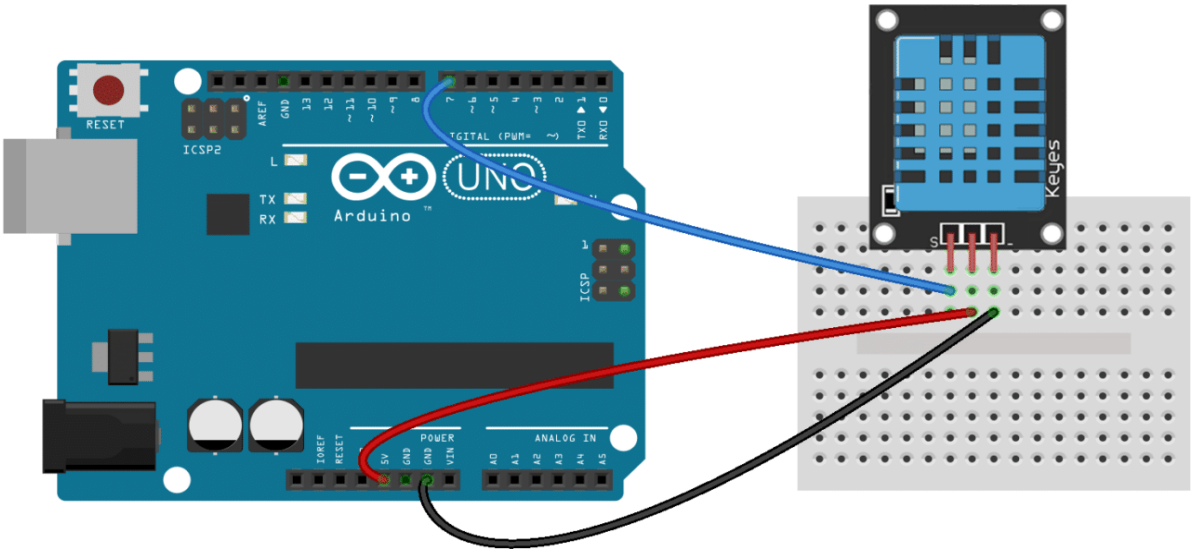
Circuit Diagram:

The left most pin the Positive pin which is connected to pin 7. The centre pin connected to 5V pin and right most pin is connected to Gnd of the Arduino board.

S -> output pin (7)

Centre Pin -> 5V

Right Pin -> GND



Code:

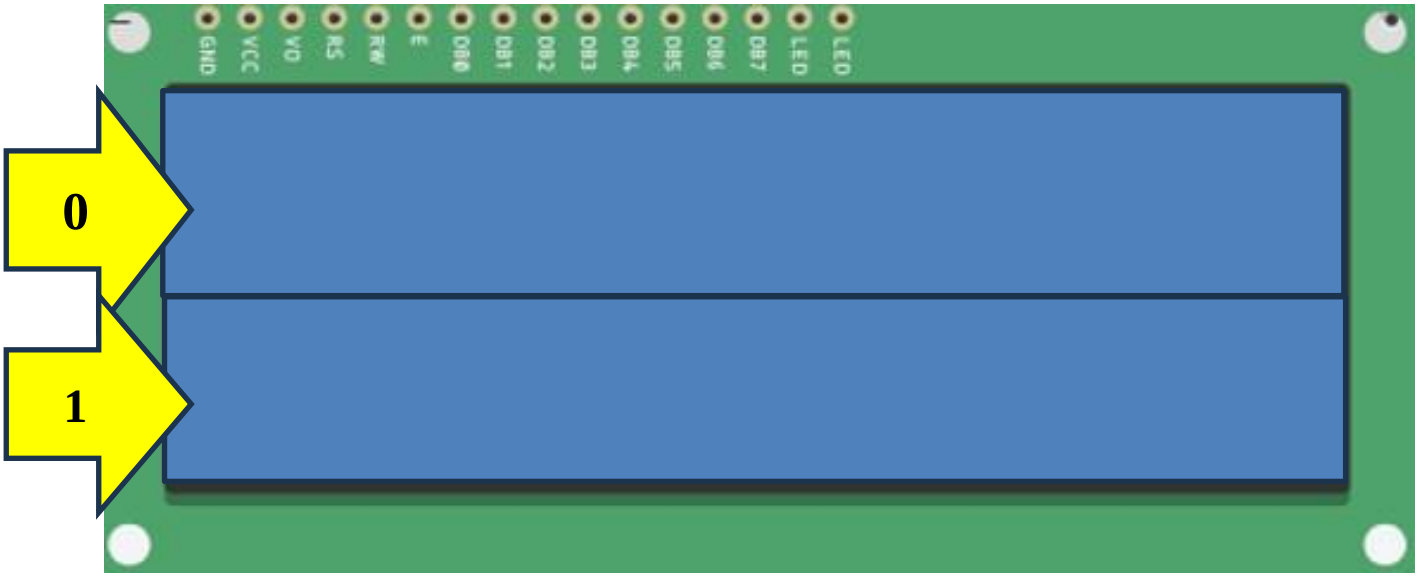
Note: Before executing the code dhtlib library has to be installed. The output has to be seen in the serial monitor with the baud rate of 9600.

```
#include <dht.h>

#define DHT11_PIN 7

dht DHT;


void setup(){

Serial.begin(9600);

}

void loop(){

DHT.read11(DHT11_PIN);

Serial.print("Temperature = ");

Serial.println(DHT.temperature);

Serial.print("Humidity = ");

Serial.println(DHT.humidity);

delay(1000);

}
```

LCD Interfacing with Arduino UNO

Liquid Crystal Display (LCD) and the Internet of Things (IoT) are related in that LCDs are often used as display devices for IoT devices and systems. For example, LCD screens can be used to display real-time data from IoT sensors to provide a user interface for controlling IoT devices.

There are 16 pins in the LCD display. Each pin has its own functionality which helps to display the required data.



Liquid Crystal Display

VCC pin is connected to 5V of the Arduino.

GND pin is connected to GND of the Arduino.

V0 pin is used to adjust the contrast of the LCD display which can also be used to potentiometer to adjust the contrast of LCD.

RS [Register Select] pin has 2 modes namely: command mode and data mode.

- **Command mode** is used to give commands to the LCD. To select the command mode RS pin is connected to GND
- **Data mode** is used to display the data required using LCD. To select data mode RS pin can be connected to any digital pin of Arduino.

R/W pin is used to enable read or write mode in LCD.

- **Write mode** can be enabled by connecting it to GND which is used to display the data.
- **Read mode** can be enabled by connecting R/W pin to 5V.

E pin is used to enable the data in LCD.

Data pins [D0-D7] acts as data bus using which the data is received by the LCD.

BLA(Anode)/LED

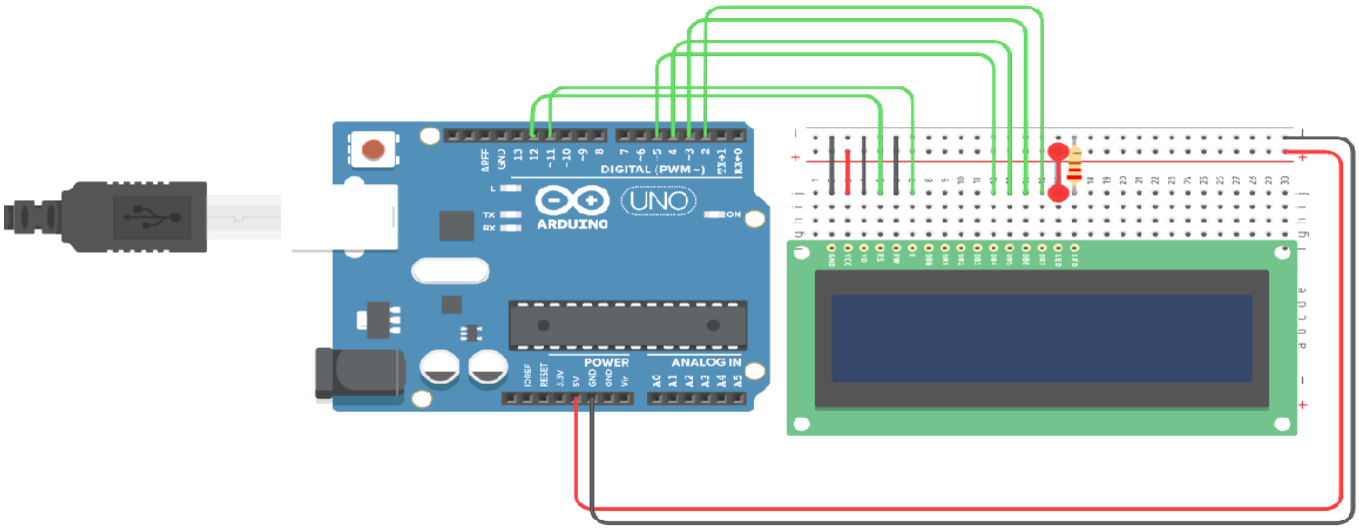
BLK(Cathode)/LED pins are used to power the LCD backlight.

- BLA pin needs to be connected to VCC.
- BLK pin needs to be connected to GND.

Components Required

- Arduino Board
- LCD
- 1k ohm resistor
- Jump wires
- breadboard

Circuit diagram



- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- LCD R/W pin to GND
- LCD VSS pin to GND
- LCD VCC pin to 5V
- LCD BLA/LED to 5V
- LCD BLK/LED to GND through resistor

Code

```
// include the library code:

#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

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

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

 lcd.begin(16, 2); //  lcd is  the name of the object of the class LiquidCrystal.

                   //16 is the number of columns and 2 is the number of rows.

}


void loop()
{
  lcd.setCursor(0, 0);//zero th coloumn, zero th row
```

```
  lcd.print("Hello" );
 delay(3000);
  lcd.setCursor(0, 1); //zero th coloumn, 1st row
  lcd.print("Good Morning");
}
```

7 - SEGMENT DISPLAY

A seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information. The numerical digits 0 to 9 are the most common characters displayed on seven-segment displays.

The seven segments are arranged as a rectangle, with two vertical segments on each side and one horizontal segment each at the top, middle, and bottom. The individual segments are referred to by the letters "a" to "g", and an optional decimal point (an "eighth segment", referred to as DP) is sometimes used for the display of non-integer numbers.

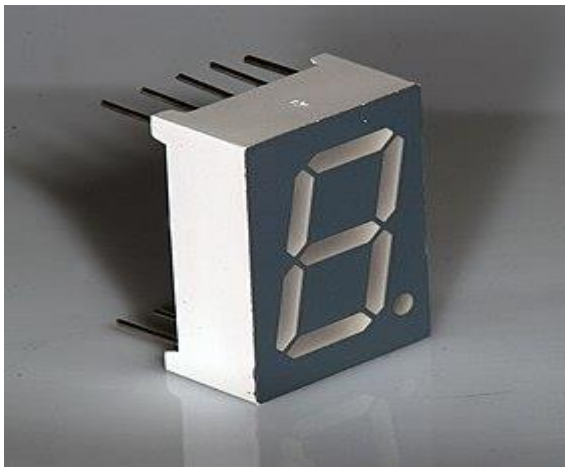
The *7-segment display*, also written as “seven segment display”, consists of seven LEDs (hence its name) arranged in a rectangular fashion as shown. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed.

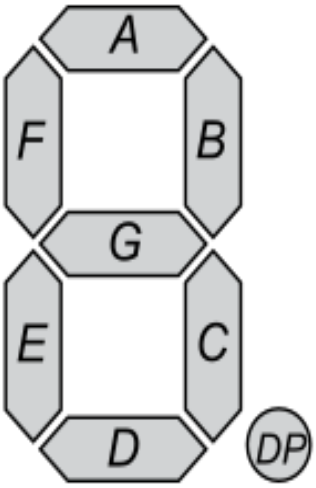
An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point, (DP) when two or more 7-segment displays are connected together to display numbers greater than ten.

The displays common pin is generally used to identify which type of 7-segment display it is. As each LED has two connecting pins, one called the “Anode” and the other called the “Cathode”, there are therefore two types of LED 7-segment display called: **Common Cathode** (CC) and **Common Anode** (CA).

The most common patterns used for each of these are:

0 1 2 3 4 5 6 7 8 9



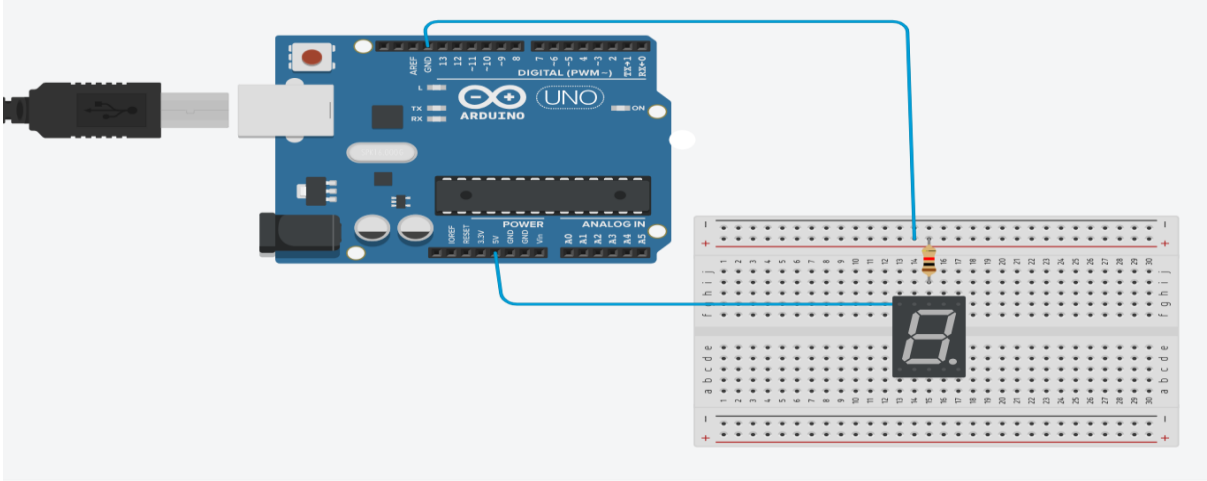


Components Required:

- Arduino Uno
- 7 – Segment LED Display
- Jump Wires
- Bread Board.
- Resistors

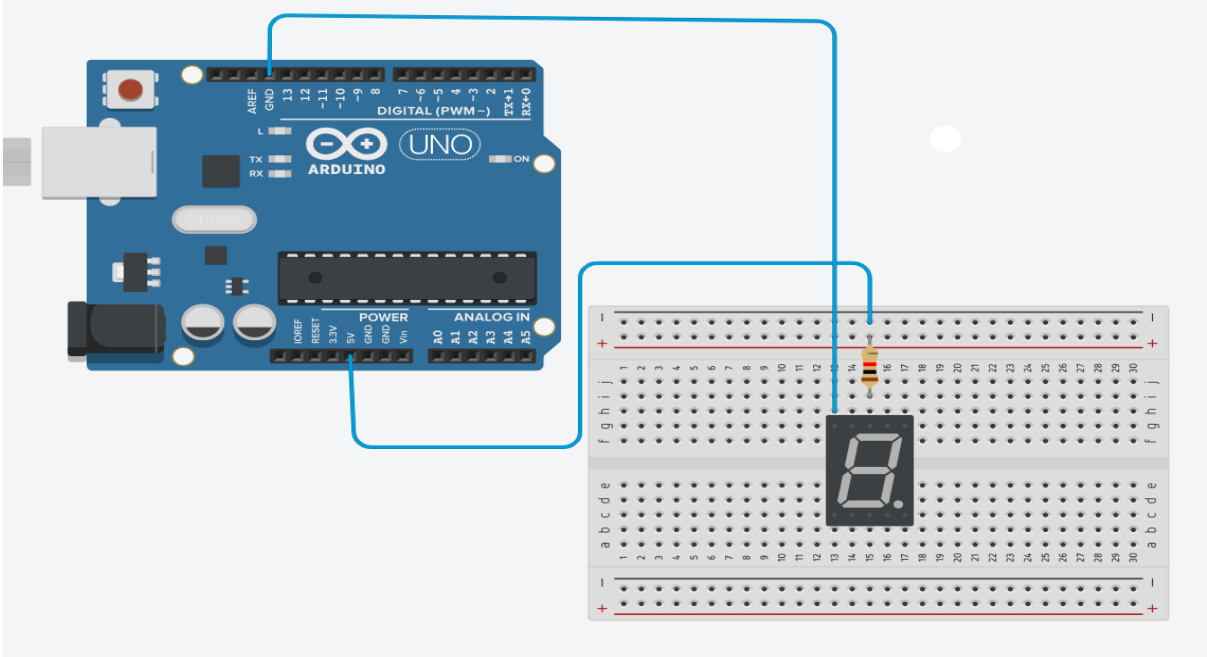
Connections Required to Identify the Common Cathode and Common Anode in 7 segment LED display

Identification of 7 Segment LED as Cathode or Anode:



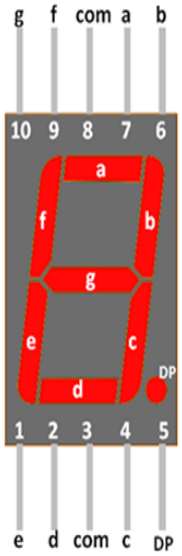
Here the Common Pin of the 7 segment LED Display is connected to GND through the resistor. Any pin other than the COM pin is connected to 5V for power supply. If anyone of the LED Segment glows then it's not common Cathode.

If the LED doesn't glow then Connections has to be changed as follows to check if its common anode:

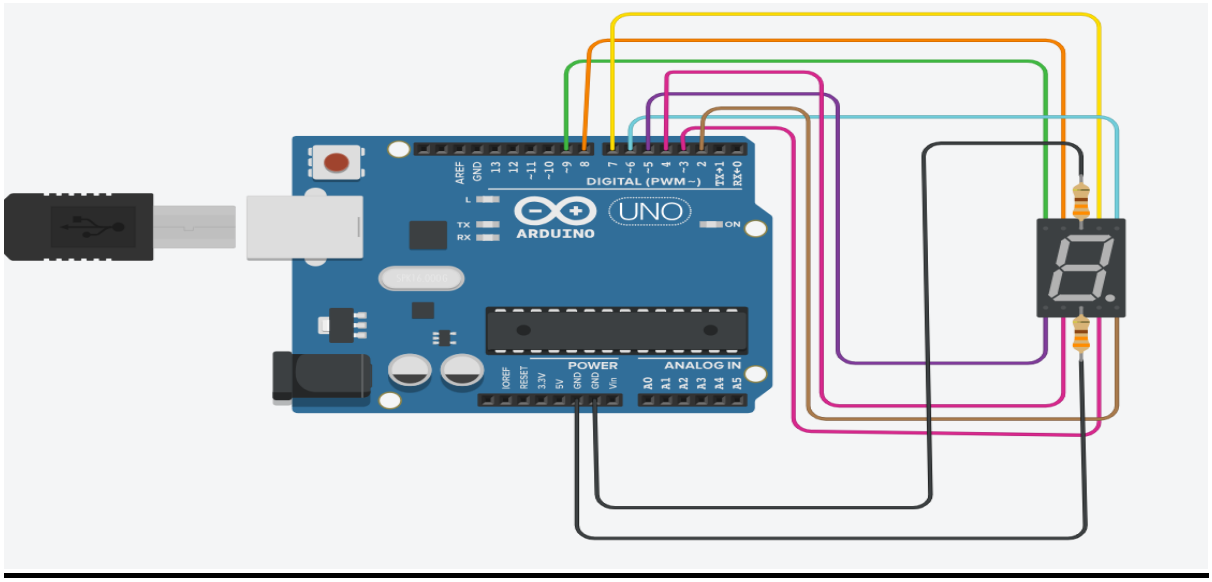


Here the common pin is connected to 5V pin through a resistor. Any other pin other than common pin is connected to GND. If any one of the LED segment glows then its common anode.

7 – Segment LED Display Connection for Cathode:



- G is connected to Pin 9
- F is connected to Pin 8
- A is connected to Pin 7
- B is connected to Pin 6
- E is connected to Pin 5
- D is connected to Pin 4
- C is connected to Pin 3
- DP is connected to Pin 2
- Centre Pins (COM) are Connected to GND.



Code:

```
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
}

void loop()
{
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
```

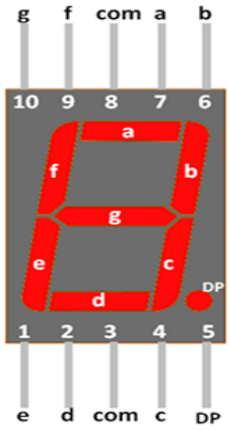
```
 digitalWrite(9, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
 digitalWrite(2, LOW);
 digitalWrite(3, HIGH);
 digitalWrite(4, LOW);
 digitalWrite(5, LOW);
 digitalWrite(6, HIGH);
 digitalWrite(7, LOW);
 digitalWrite(8, LOW);
 digitalWrite(9, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
 digitalWrite(2, LOW);
 digitalWrite(3, LOW);
 digitalWrite(4, HIGH);
 digitalWrite(5, HIGH);
 digitalWrite(6, HIGH);
 digitalWrite(7, HIGH);
 digitalWrite(8, LOW);
 digitalWrite(9, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(2, LOW);
 digitalWrite(3, HIGH);
 digitalWrite(4, HIGH);
```

```
 digitalWrite(5, LOW);
 digitalWrite(6, HIGH);
 digitalWrite(7, HIGH);
 digitalWrite(8, LOW);
 digitalWrite(9, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(2, LOW);
 digitalWrite(3, HIGH);
 digitalWrite(4, LOW);
 digitalWrite(5, LOW);
 digitalWrite(6, HIGH);
 digitalWrite(7, LOW);
 digitalWrite(8, HIGH);
 digitalWrite(9, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(2, LOW);
 digitalWrite(3, HIGH);
 digitalWrite(4, HIGH);
 digitalWrite(5, LOW);
 digitalWrite(6, LOW);
 digitalWrite(7, HIGH);
 digitalWrite(8, HIGH);
 digitalWrite(9, HIGH);
```

```
 delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(2, LOW);
 digitalWrite(3, HIGH);
 digitalWrite(4, HIGH);
 digitalWrite(5, HIGH);
 digitalWrite(6, LOW);
 digitalWrite(7, HIGH);
 digitalWrite(8, HIGH);
 digitalWrite(9, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(2, LOW);
 digitalWrite(3, HIGH);
 digitalWrite(4, LOW);
 digitalWrite(5, LOW);
 digitalWrite(6, HIGH);
 digitalWrite(7, HIGH);
 digitalWrite(8, LOW);
 digitalWrite(9, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(2, LOW);
 digitalWrite(3, HIGH);
 digitalWrite(4, HIGH);
 digitalWrite(5, HIGH);
```

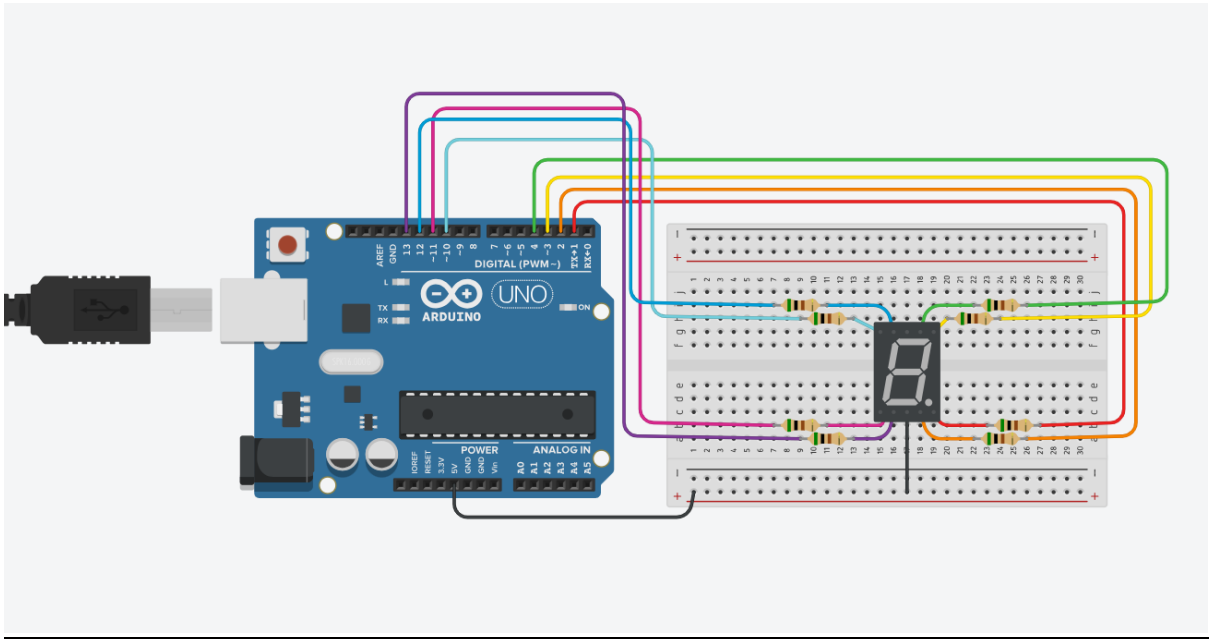
```
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

7 segment LED for Anode



Connections for Anode:

- G is connected to Pin 10
- F is connected to Pin 12
- A is connected to Pin 4
- B is connected to Pin 3
- E is connected to Pin 11
- D is connected to Pin 13
- C is connected to Pin 2
- DP is connected to Pin 1
- Centre Pins (COM) are Connected to 5V



Code:

```
unsigned const int A = 4;
unsigned const int B = 3;
unsigned const int C = 2;
unsigned const int D = 13;
unsigned const int E = 11;
unsigned const int F = 12;
unsigned const int G = 10;
unsigned const int H = 1;
void setup(void)
{
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(H, OUTPUT);
}
//My Functions

void zero(void) {
```

```
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
}

void one(void) {
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
}

void two(void) {
  digitalWrite(A, LOW);
```

```
  digitalWrite(B, LOW);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
}

void three(void) {
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
}

void four(void) {
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
```

```
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
}

void five(void) {
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
}

void six(void) {
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
```

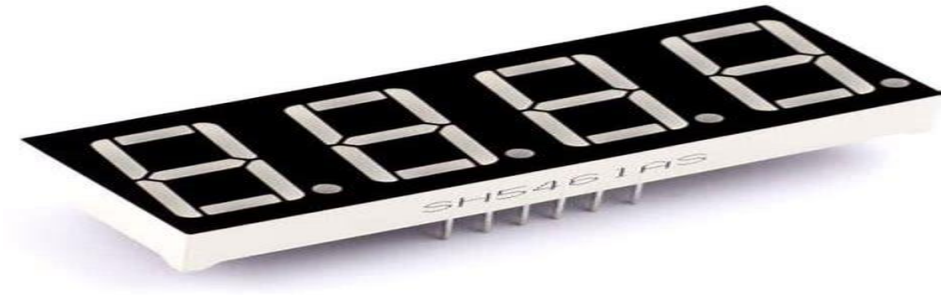
```
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
}
void seven(void) {
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(H, HIGH);
}
void eight(void) {
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
```

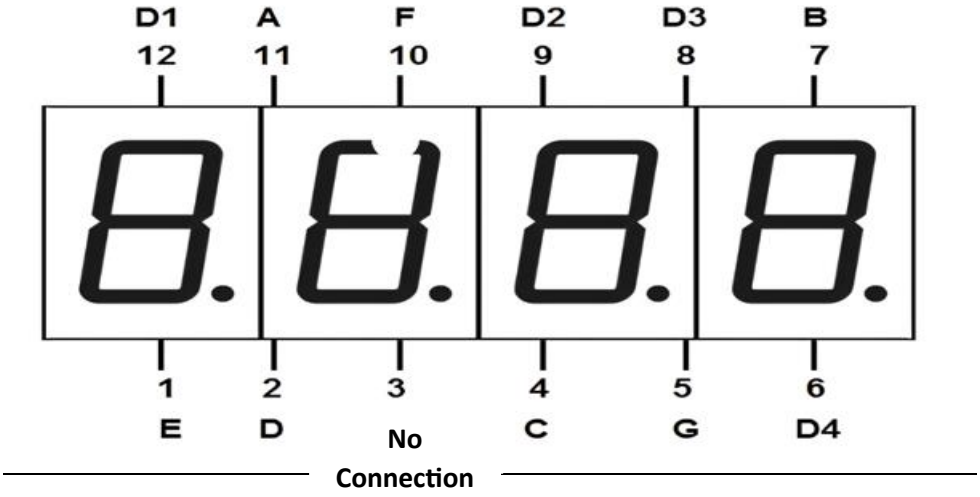
```
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
}
void nine(void) {
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(H, HIGH);
}

// Start
void loop(void)
{
  zero();
  delay(1000);

  one();
  delay(1000);
```

```
  two();
  delay(1000);

  three();
  delay(1000);

  four();
  delay(1000);

  five();
  delay(1000);

  six();
  delay(1000);

  seven();
  delay(1000);

  eight();
  delay(1000);

  nine();
  delay(1000);
}
```

4- DIGIT 7 SEGMENT DISPLAY

7 segment display with 4 digits using Arduino - Some basic things take up almost all of the digital pins on the arduino uno boards with 13 digital pins. Most displays have 12 breakout pins that connect either directly to the Arduino. This Display contains four digits where D1, D2, D3 and D4 pins are used for displaying the first, second, third and fourth digits respectively.





Identification of 7 Segment LED as Cathode or Anode:

Here the D1 Pin of the 4 digit 7 segment LED Display is connected to GND through the resistor. Any pin other than the D2,D3 and D4 pins is connected to 5V for power supply. If anyone of the LED Segment glows in the First digit of the LED display then it's not common Cathode.

If the LED doesn't glow then Connections has to be changed as follows to check if its common anode:

Here the D1 pin is connected to 5V pin through a resistor. Any other pin other than D2, D3 and D4 are connected to GND. If any one of the LED segment glows in Digit 1 then its common anode.

Connections for common Cathode:

- D1 is connected to Pin 2
- D2 is connected to Pin 3
- D3 is connected to Pin 4
- D4 is connected to Pin 5
- A is connected to Pin 6
- B is connected to Pin 7
- C is connected to Pin 8
- D is connected to Pin 9
- E is connected to Pin 10
- F is connected to Pin 11
- G is connected to Pin 12

Code:

```
int D1=2;
int D2=3;
int D3=4;
int D4=5;

int a=6;
int b=7;
int c=8;
int d=9;
int e=10;
int f=11;
int g=12;
```

```
void setup(){
  pinMode(D1,OUTPUT);
  pinMode(D2,OUTPUT);
  pinMode(D3,OUTPUT);
  pinMode(D4,OUTPUT);

  pinMode(a,OUTPUT);
  pinMode(b,OUTPUT);
  pinMode(c,OUTPUT);
  pinMode(d,OUTPUT);
  pinMode(e,OUTPUT);
  pinMode(f,OUTPUT);
  pinMode(g,OUTPUT);

}

void loop(){
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);

zero();
delay(1000);

one();
```

```
delay(1000);

two();
delay(1000);

three();
delay(1000);

four();
delay(1000);

five();
delay(1000);

six();
delay(1000);

seven();
delay(1000);

eight();
delay(1000);

nine();
delay(1000);
  }
```

```
  void zero(){
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,LOW);
}

void one(){
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,LOW);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,LOW);
}

void two(){
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,LOW);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,LOW);
```

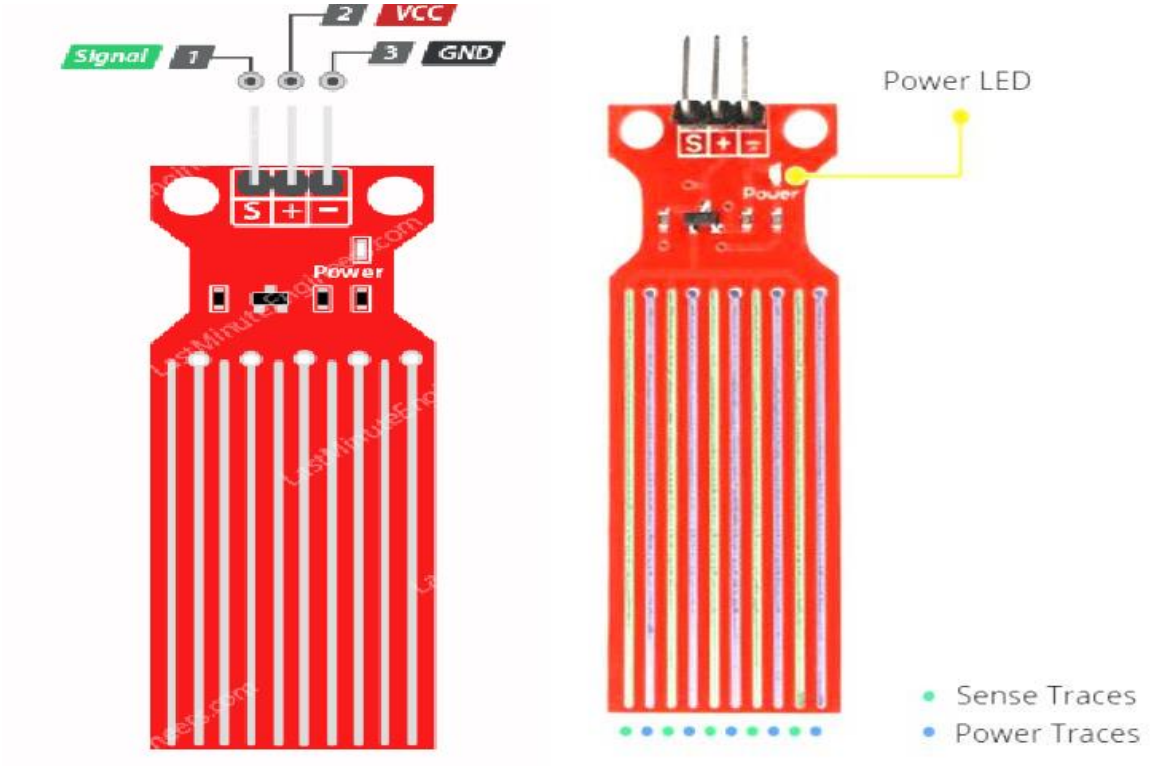
```
  digitalWrite(g,HIGH);
}

void three(){
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
  digitalWrite(g,HIGH);
}

void four(){
  digitalWrite(a,LOW);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
}

void five(){
  digitalWrite(a,HIGH);
  digitalWrite(b,LOW);
  digitalWrite(c,HIGH);
```

```
  digitalWrite(d,HIGH);
  digitalWrite(e,LOW);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
}

void six(){
  digitalWrite(a,HIGH);
  digitalWrite(b,LOW);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
}

void seven(){
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
  digitalWrite(g,LOW);
}

void eight(){
```

```
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
}

void nine(){
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,LOW);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
}
```

Interfacing Water Level Sensor with Arduino Uno

The water sensor or water level sensor is used to detect water, water leakage, rainfall, tank overflow, or to measure water level.



The water level sensor has 3 pins:

S (Signal) pin: is an analog output that will be connected to one of the analog inputs on your Arduino.

+(VCC) pin: supplies power for the sensor. It is recommended to power the sensor with between 3.3V - 5V.

- (GND) pin: is a ground connection.

The sensor has a series of ten exposed copper traces. Five are power traces and five are sense traces. These traces are interlaced in parallel so that there is one sense trace between every two power traces. These traces are not connected unless they are bridged by water when submerged.

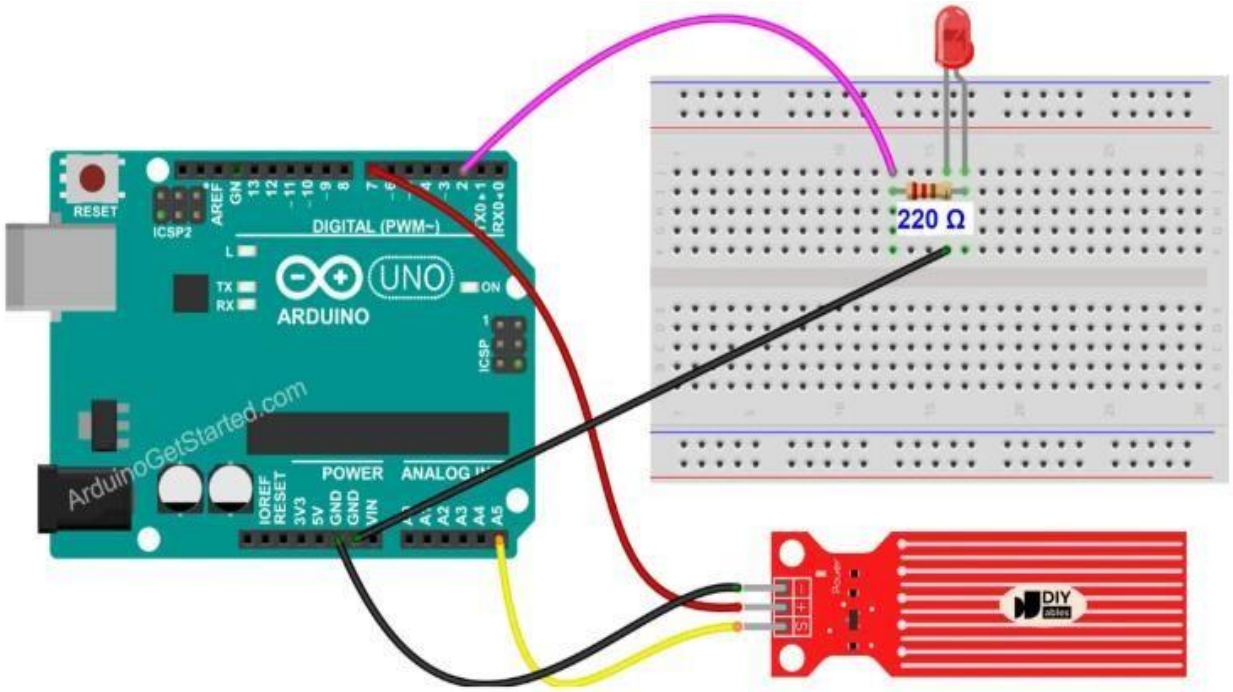
The traces act as a variable resistor whose resistance varies according to the water level. The change in resistance corresponds to the distance from the top of the sensor to the surface of the water. The resistance is inversely proportional to the height of the water:

The more water the sensor is immersed in, the better the conductivity is, the lower the resistance is. The less water the sensor is immersed in, the worse the conductivity is, the higher the resistance is.

Components Required

- Jump Wires
- Water Sensor
- Arduino Uno
- LED Bulb
- Resistor (220 Ohms)

Circuit Diagram



Connections

The Negative pin of Water sensor is connected to GND. The positive pin of the water sensor is connected to any digital Pin of Arduino Uno. The S pin of water sensor is connected to analog pin A5 of the Arduino Uno.

The positive end (Anode) of the LED is connected with one end of the resistor and the other end of the resistor is connected to any digital pin of Arduino. The negative end (Cathode) is connected to GND of Arduino Uno.

Code

```
#define LED_PIN     2
#define water_sensor  7
#define SIGNAL_PIN  A5
#define THRESHOLD   300
int value = 0; // variable to store the sensor value
void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN,   OUTPUT);   // configure D2 pin as an OUTPUT
  pinMode(water_sensor, OUTPUT);   // configure D7 pin as an OUTPUT
  digitalWrite(water_sensor, LOW); // turn the sensor OFF
  digitalWrite(LED_PIN,   LOW); // turn LED OFF
}
void loop() {
  digitalWrite(water_sensor, HIGH);  // turn the sensor ON
  delay(10);                      // wait 10 milliseconds
  value = analogRead(SIGNAL_PIN); // read the analog value from sensor
  digitalWrite(water_sensor, LOW);   // turn the sensor OFF
  Serial.print("The water level is : ");
  Serial.println(value);
  if (value > THRESHOLD) {
    Serial.println("The water is detected");
    digitalWrite(LED_PIN, HIGH);  // turn LED ON
  } else {
    digitalWrite(LED_PIN, LOW);   // turn LED OFF
  }
}
```

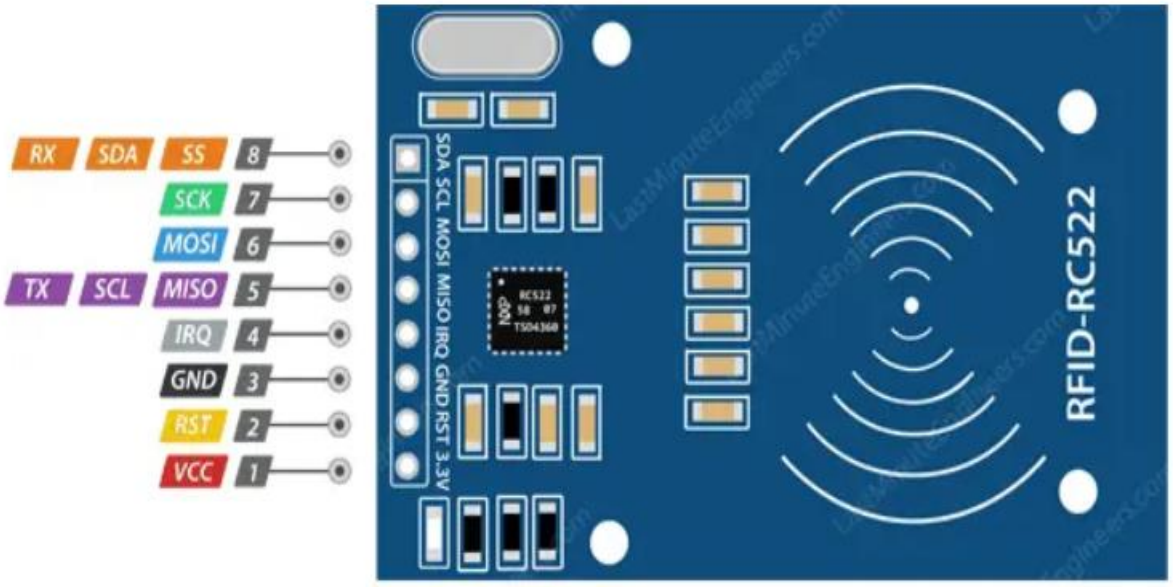
Interfacing RFID With Arduino Uno

RFID means radio-frequency identification. RFID uses electromagnetic fields to transfer data over short distances. RFID is useful to identify people, to make transactions etc., only the person with the right information on card is allowed to enter through a door in the case of an RFID system to open a door.

RFID Reader (MFRC522)

RFID Tags

RFID has 8 pins and its functionality are given below:



SDA (Serial Data): Used for data exchange between the module and microcontroller.

SCK (Serial Clock): Provides synchronization between the module and microcontroller.

MOSI (Master Out Slave In): Transfers data from the microcontroller to the module.
MISO (Master In Slave Out): Transfers data from the module to the microcontroller.

IRQ (Interrupt Request): An interrupt pin that can alert the microcontroller when RFID tag comes near the module.

RST(Reset): Resets the module.

GND(Ground*):* Serves as the ground reference.

3.3**(Power) pin:** Powers the module with 3.3 volts.

RFID includes two components:

(a) Reader

The ***reader*** consists of a radio frequency module and an antenna which generates high frequency electromagnetic field.

(b) **Tag**

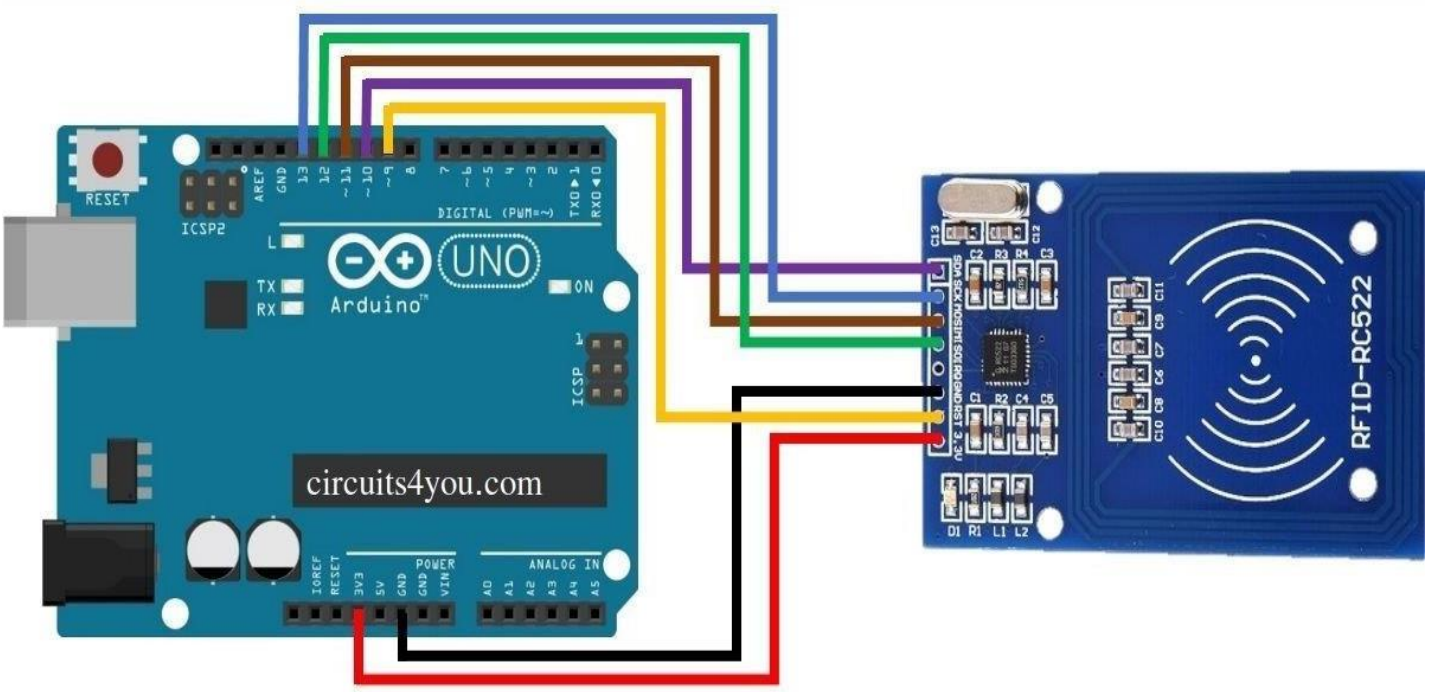
The ***tag*** is usually a passive device, which doesn't need to have power source. The tag contains a microchip that stores and processes information, and an antenna to receive and transmit a signal. The tag is used to store the information: UID (Unique ID) and data.

Multiple Tags attached to the object to be identified, like a keychain and an electromagnetic card. Each tag has its own identification (UID).

Components Required

- RFID Reader (MFRC522)
- RFID Tags (keychain and electromagnetic card)
- Jump Wires
- Arduino Uno

Circuit Diagram



- SDA pin is connected to digital pin 10 of Arduino Uno
- SCK pin is connected to digital pin 13 of Arduino Uno
- MOSI pin is connected to digital pin 11 of Arduino Uno
- MISO pin is connected to digital pin 12 of Arduino Uno
- IRQ Not Connected
- GND is connected to GND of Arduino Uno
- RST is connected to digital pin 9 of Arduino Uno
- 3.3 V is connected to 3.3 Volts of Arduino Uno

Code

```
#include <SPI.h> // The Serial Peripheral Interface (SPI) driver is a generic, full-
                 //duplex driver that transmits and receives data on a SPI bus.
#include <MFRC522.h>
#define RST_PIN  9
#define SS_PIN     10
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
Serial.begin(9600);  // Initialize serial communications with the PC
while (!Serial); //Keeps the serial monitor open
SPI.begin(); // Initializes the SPI bus
mfrc522.PCD_Init(); // Init MFRC522 card (Proximity coupling device)
delay(4);
mfrc522.PCD_DumpVersionToSerial();
Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks    "));
}
void loop() {
if ( ! mfrc522.PICC_IsNewCardPresent( )) //looks for new cards if any
{
 return;
}
if ( ! mfrc522.PICC_ReadCardSerial( )) //selects one of the PCD
{
 return;
}
mfrc522.PICC_DumpToSerial(&(mfrc522.uid)); // dump the RFID tag content
}
```

Ultrasonic sensor data collection using Arduino and MYSQL

Capturing data from sensors

The output of a device that recognizes and reacts to an input of any kind from the external world is called sensor data. The output can be used to direct a process, supply information to the end user, or serve as input for another system.

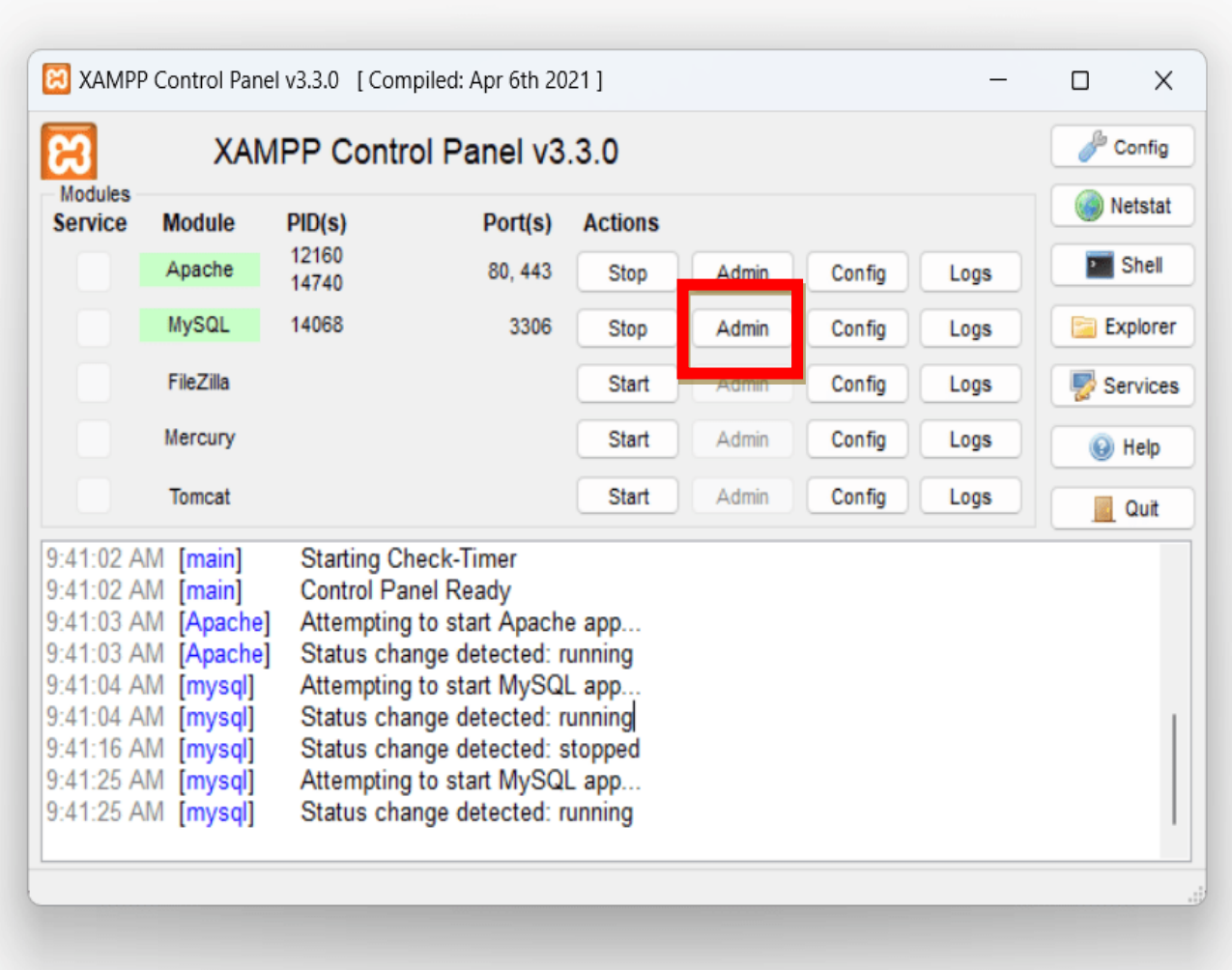
Steps to follow:

1. MYSQL Database Installation Using XAMPP

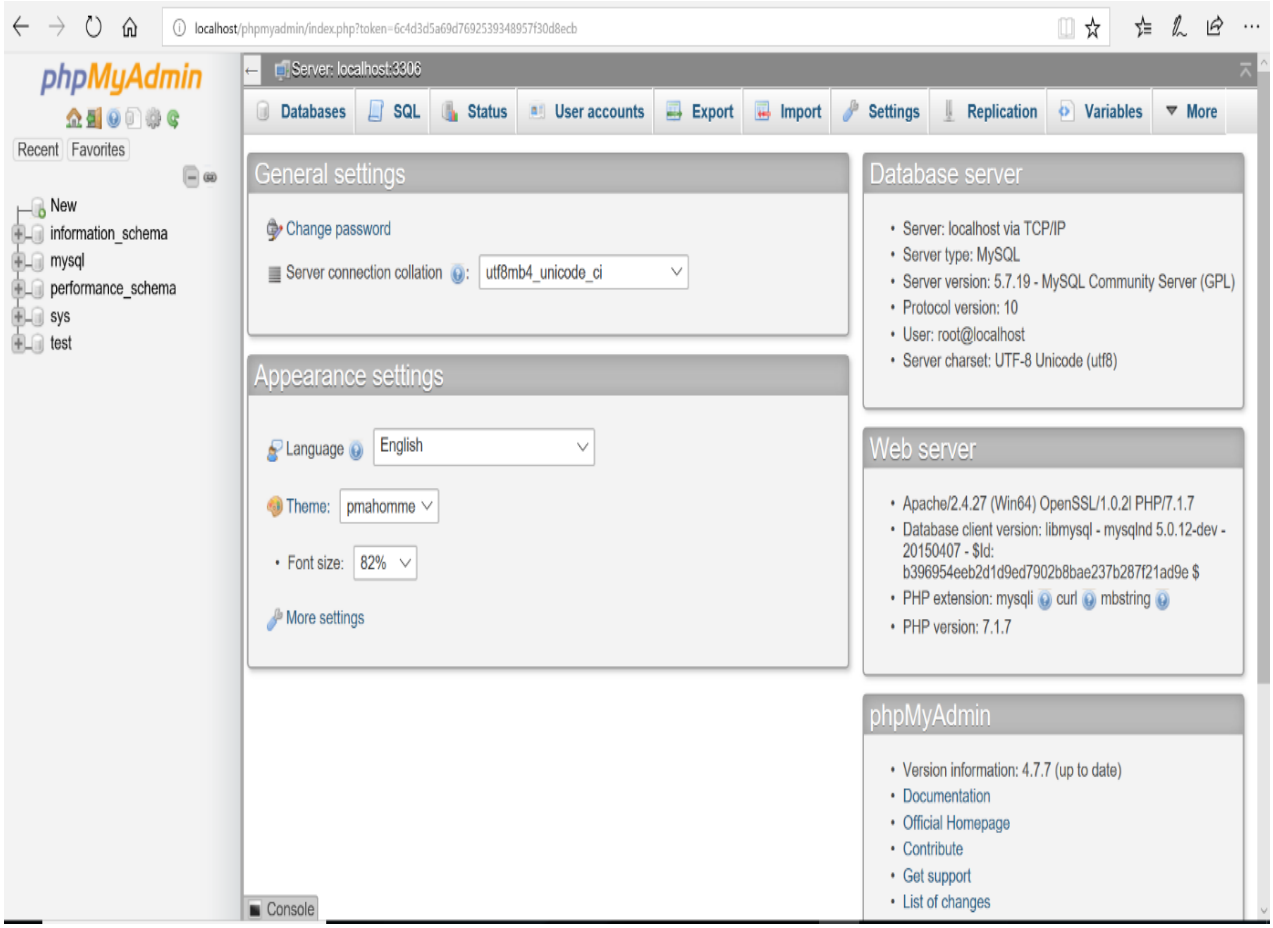
XAMPP is installed by using the following link by installing the zip file:
Download XAMPP (apachefriends.org)

Once the XAMPP is installed do the following:

a. **Start** the Apache and MySQL
b. Click on **Admin** to open MySQL IDE



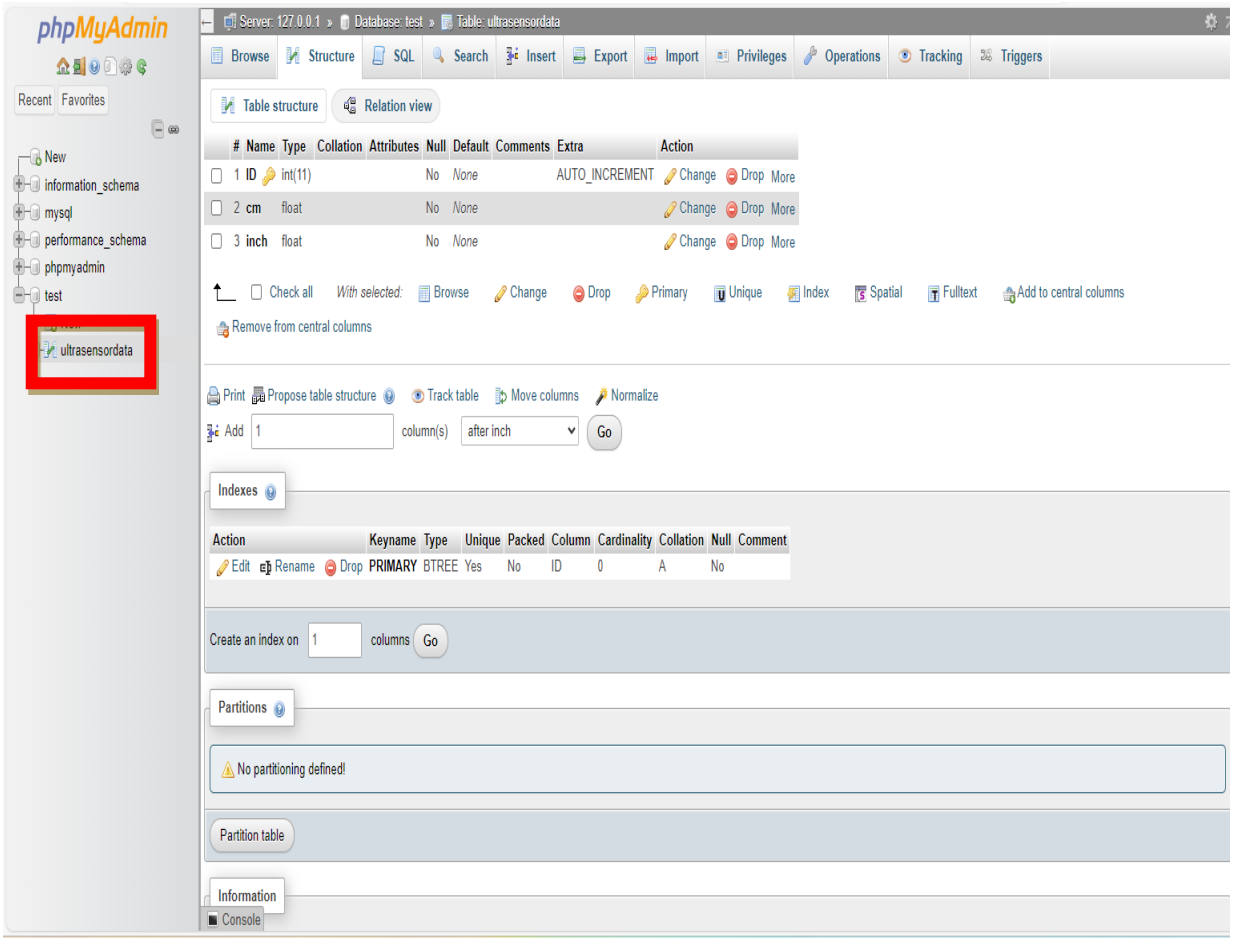
By clicking on admin page of MYSQL, PHPMYADMIN (phpMyAdmin is a free and open-source administration tool for MySQL and MariaDB as a portable web application written primarily in PHP) is opened successfully as shown below.



After installing phpMyAdmin, needs to create the following:

a. Database[test] and
b. A table [distance] inside the database [test]
c. Create columns [inside table distance] has to be created to store the data from the sensor.

For ultrasonic senor experiment,

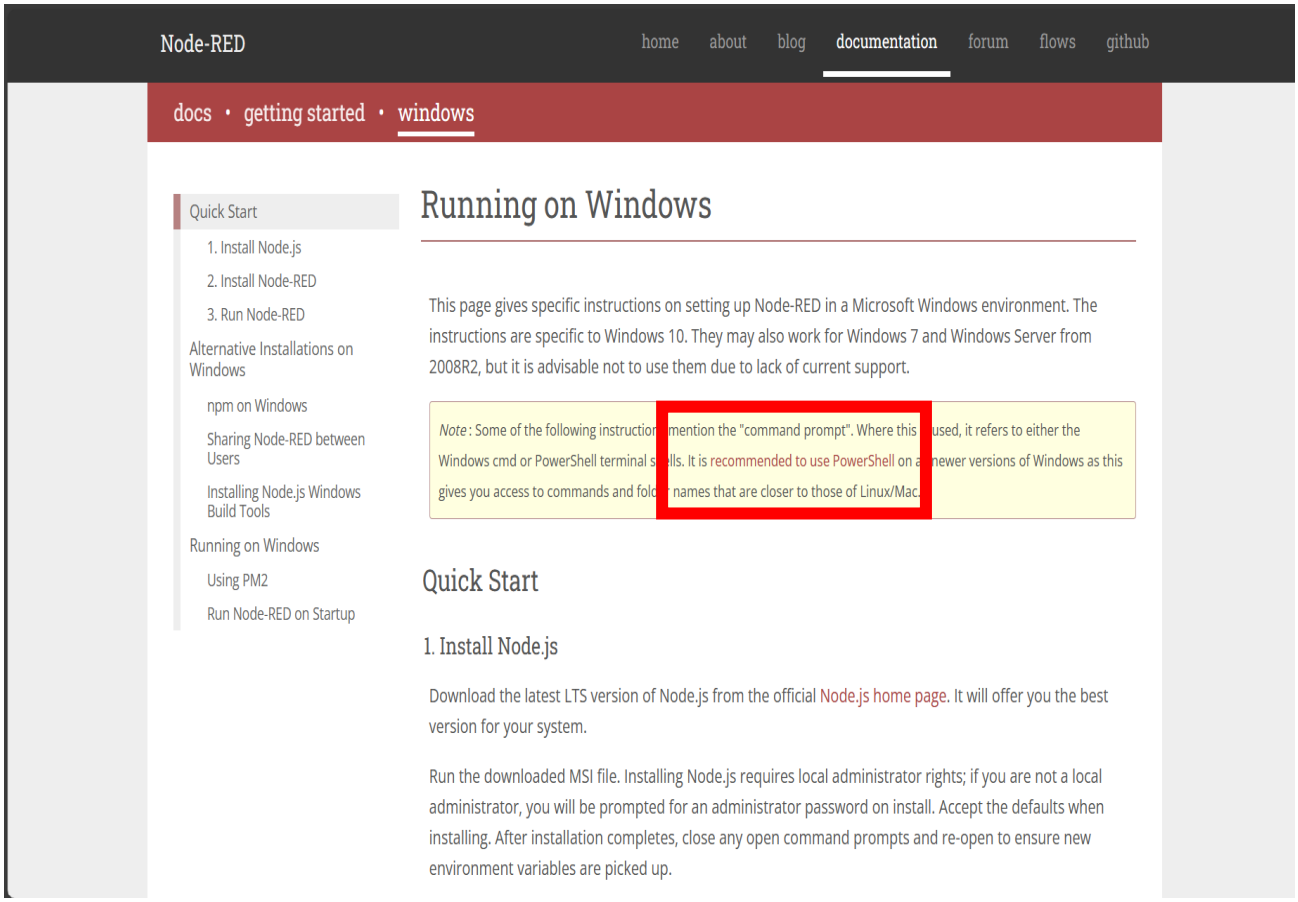


2. Node Red Installation

Node-RED is the flow-based, low-code development tool for visual programming, developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.

Node-RED is built on top of Node.js.
So, Node.js has to be installed before Node-RED to make the NODE-RED work.
Before Installing Node-Red, Node-Red is installed using the link:
[Running on Windows : Node-RED (nodered.org)]

The below page supports with the commands, which is to be followed for the installation.



3. **The highlighted link in the image is clicked to download the Node.js.**

Once installed, open a command prompt and run the following command in the command prompt to ensure Node.js and npm are installed correctly.

a. Using cmd - `node --version && npm --version`

Need to receive back output that looks similar to:

```
v18.15.0
9.5.0
```

Install Node-RED

Installing Node-RED as a global module adds the command `node-red` to your system path.

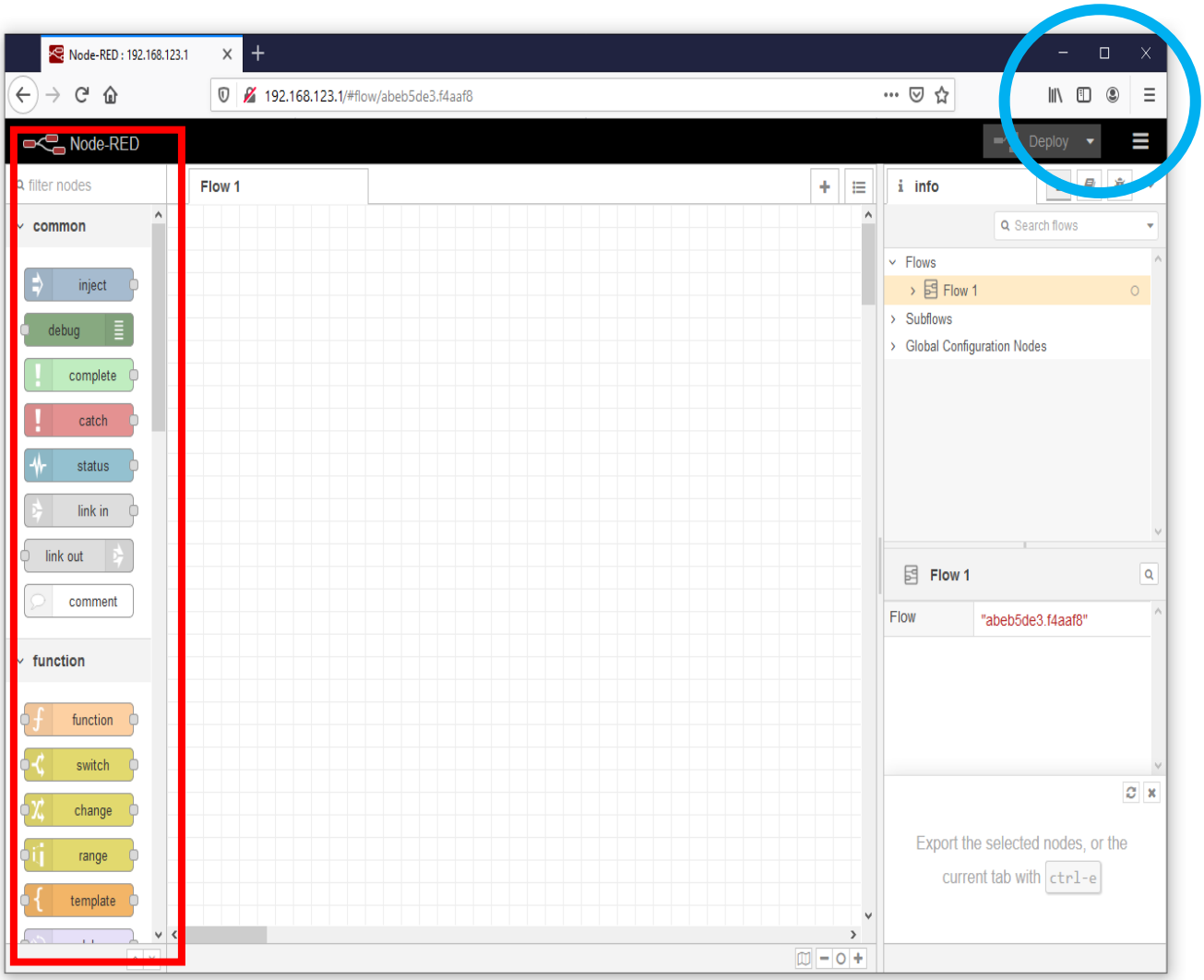
b. Execute the following at the command prompt:

```
npm install -g --unsafe-perm node-red
```

Run Node-RED

c. Once installed, now ready to run **node-red.**

d. After running the node-red command in command prompt, the server's name is used to open the home page of node red. (press ctrl+enter). Will opens the Node-RED IDE.

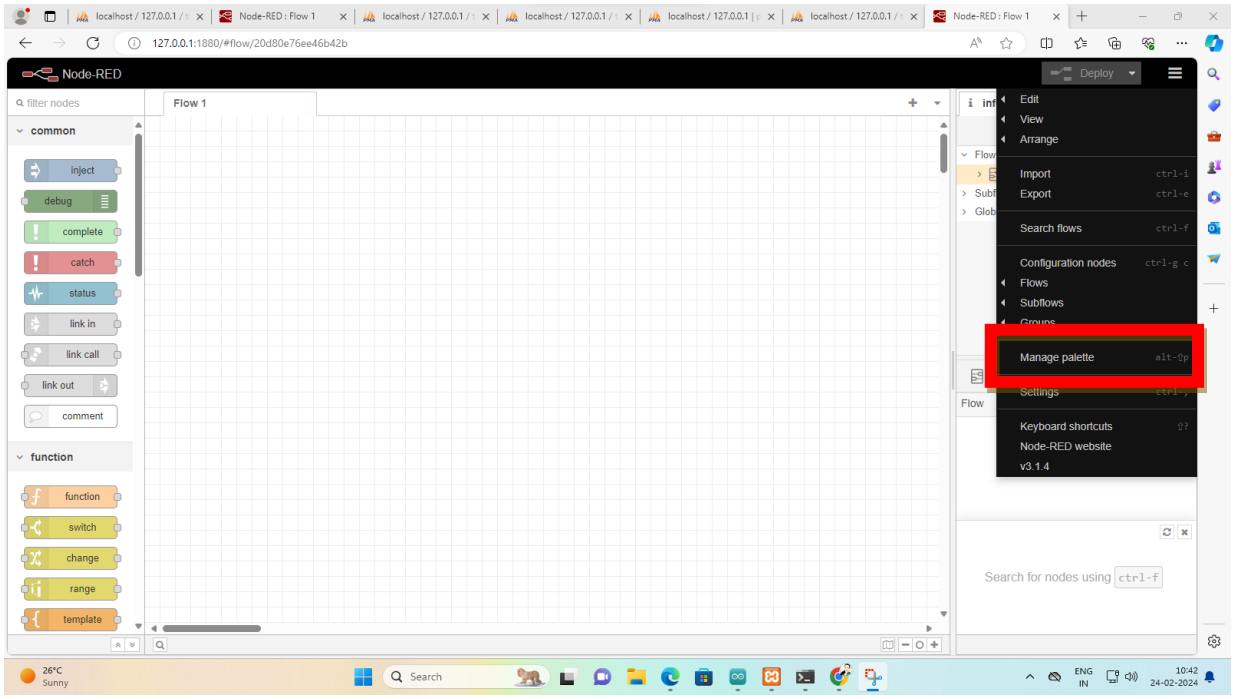


Home Page of Node-Red

As shown in the **red coloured rectangle**, the **Pallets** are used to drag and drop in the flow which is used to transfer the data from Arduino to MySQL.

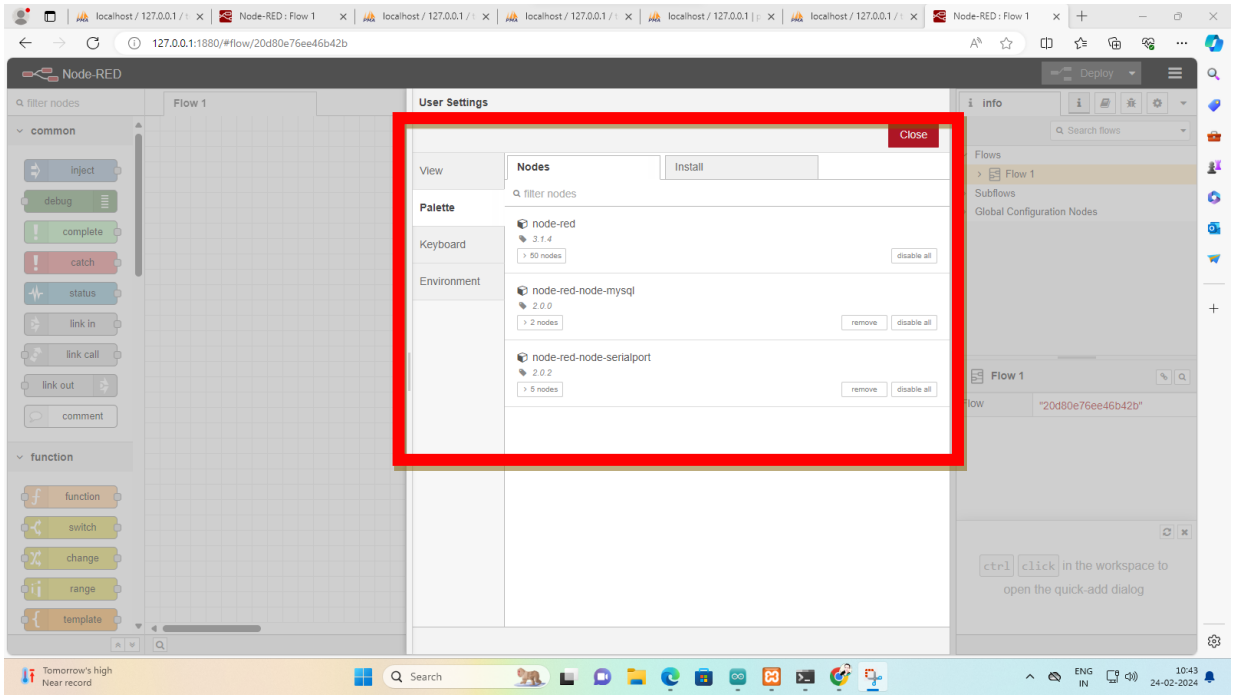
If the required pallet is not present, then that particular pallet can be installed easily by clicking on the **3 horizontal lines** in the top right corner of the node-red and selecting Manage pallets option as shown in the **blue circle**.

Aim is to transfer the data from the **sensor to MySQL** so that the data is stored permanently in the database.



a. Install the following pallets:

i) install node-red
ii) node-red-node-mysql
iii) node-red-node-serialport



4. **Example application** we are taking the code of **Ultrasonic sensor** as example and trying to store the data generated by the sensor in the MySQL database.

   a. Code of Ultrasonic Sensor to be written in the Arduino IDE:

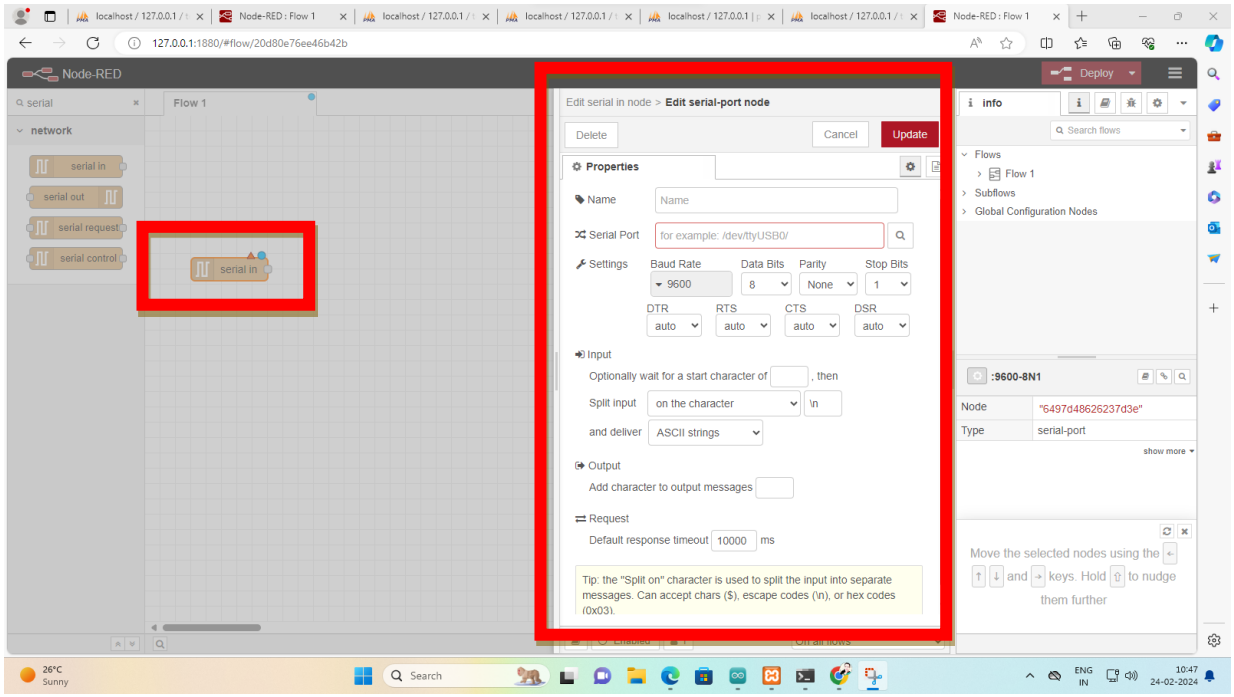
```
const int trigPin = 7;

const 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);
```

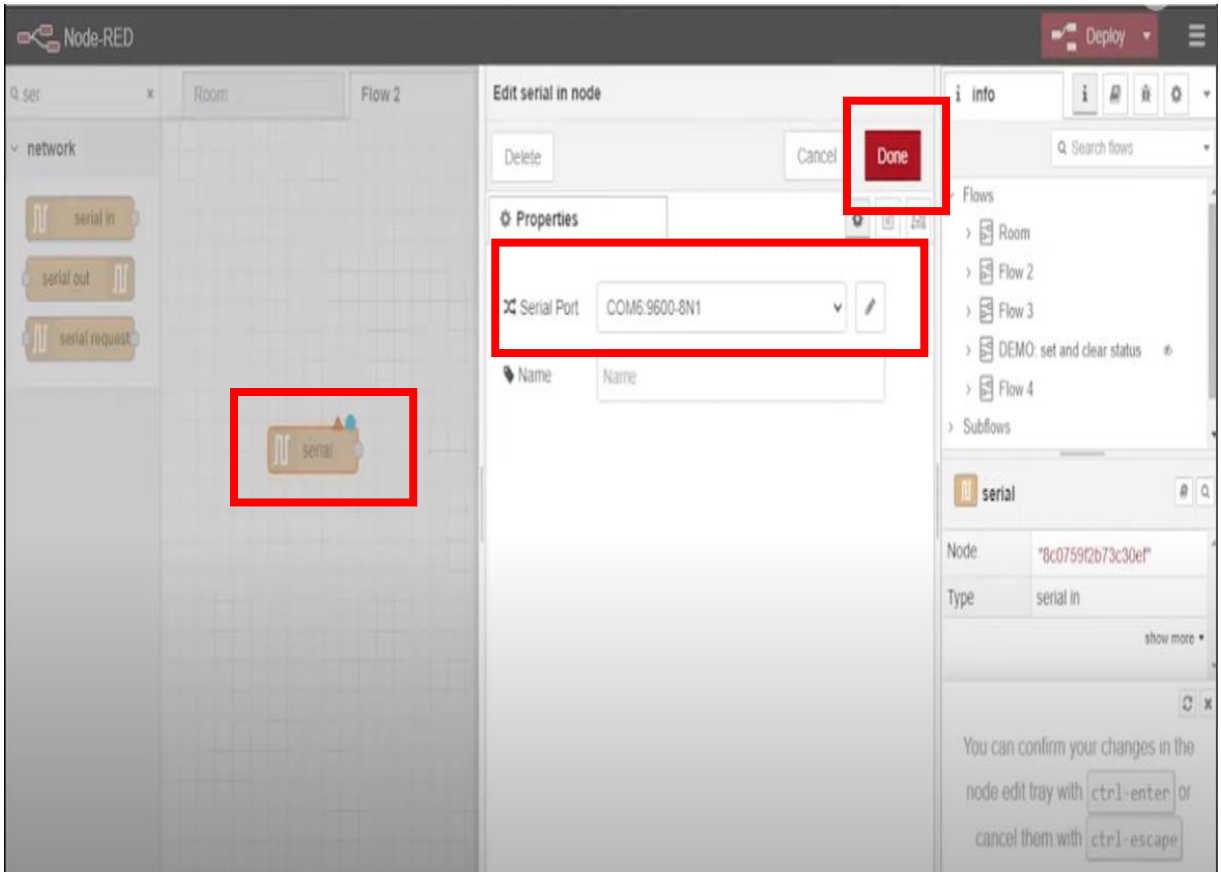
```
// The data has to be fed in this format so that it will be stored in MySQL
  Serial.print("{\"inches\":");
  Serial.print(inches);
// The data has to be fed in this format so that it will be stored in MySQL
  Serial.print(",\"cm\":");.
  Serial.print(cm);
  Serial.println("}")
  delay(1000);
}
long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2;
}
long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}
```

5. Establishing the connection Application output to Database

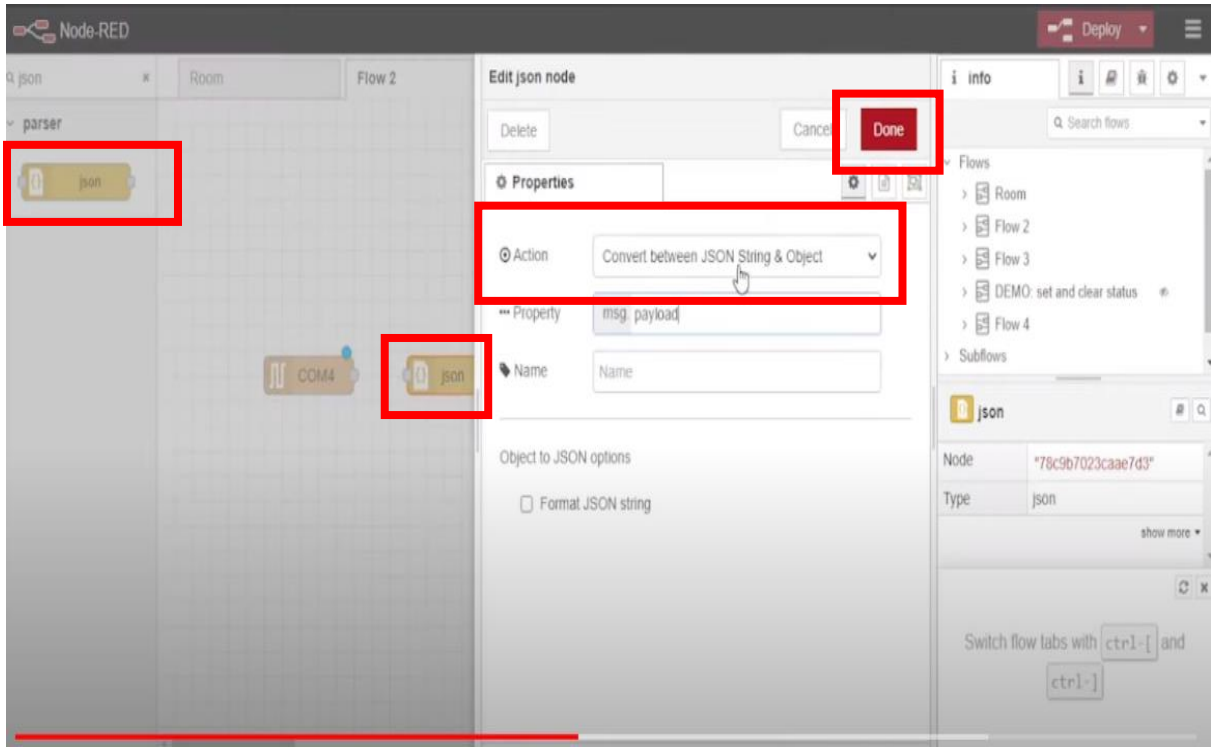
We need four different pallet to achieve this connection and they are:

a. The first pallet to be dragged from node-red is **Serial In.** Which is used to specify the port used by Arduino UNO. As given in the figure below (red rectangle), by double clicking on the **Serial In Pallet**. A dialog box is opened which is used to specify the **baud value (9600) and COM number(which ever the port identified by Arduino IDE)**. And finally press **Done** button.

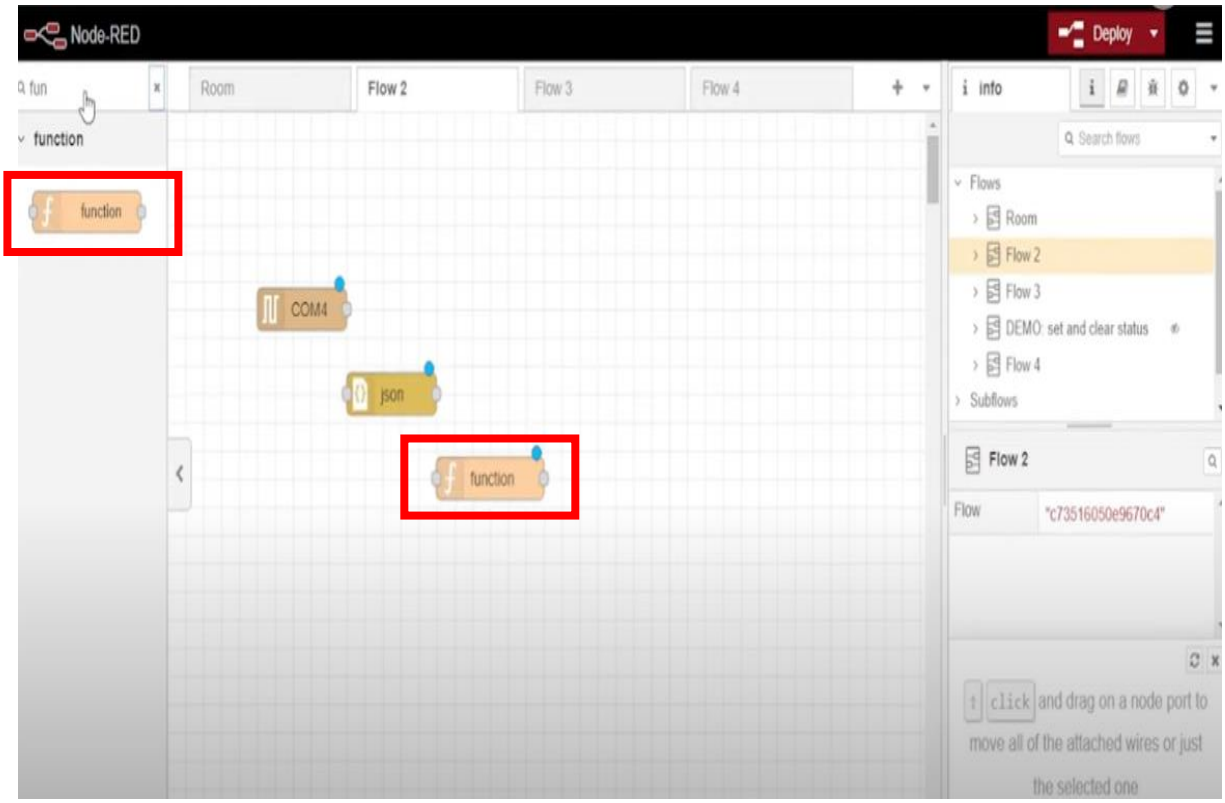

Node-RED
Deploy
Flow 1
network
serial in
serial out
serial request
serial control
Edit serial in node > Edit serial-port node
Delete
Cancel
Update
Properties
Name
Serial Port
for example: /dev/ttyUSB0/
Settings
Baud Rate
9600
Data Bits
8
Parity
None
Stop Bits
1
DTR
RTS
CTS
DSR
auto
Input
Optionally wait for a start character of
, then
Split input
on the character
\n
and deliver
ASCII strings
Output
Add character to output messages
Request
Default response timeout
10000
ms
Tip: the "Split on" character is used to split the input into separate messages. Can accept chars ($), escape codes (\n), or hex codes (0x03).
info
Search flows
Flows
Flow 1
Subflows
Global Configuration Nodes
:9600-8N1
Node
"6497d48626237d3e"
Type
serial-port
show more
Move the selected nodes using the
and
keys. Hold
to nudge them further
Search
ENG
IN
10:47
24-02-2024


Node-RED
Deploy
Flow 2
network
serial in
serial out
serial request
serial
Edit serial in node
Delete
Cancel
Done
Properties
Serial Port
COM6:9600-8N1
Name
info
Search flows
Flows
Room
Flow 2
Flow 3
DEMO: set and clear status
Flow 4
Subflows
serial
Node
"8c0759f2b73c30ef"
Type
serial in
show more
You can confirm your changes in the node edit tray with ctrl-enter or cancel them with ctrl-escape

b. The second pallet to be dragged is **JSON,** and double click on the pallet, Where the action to be selected as **Convert between JSON String and Object**.



c. The third pallet to be selected is **function** where the below code has to be written.



i) Drag and drop the function pallent
ii) Double click on the pallet
iii) Write the code on the function pallet to initialize the data values from the sensor readings of the application
iv) Write the SQL Query to insert the sensor generated values to the database for future use.

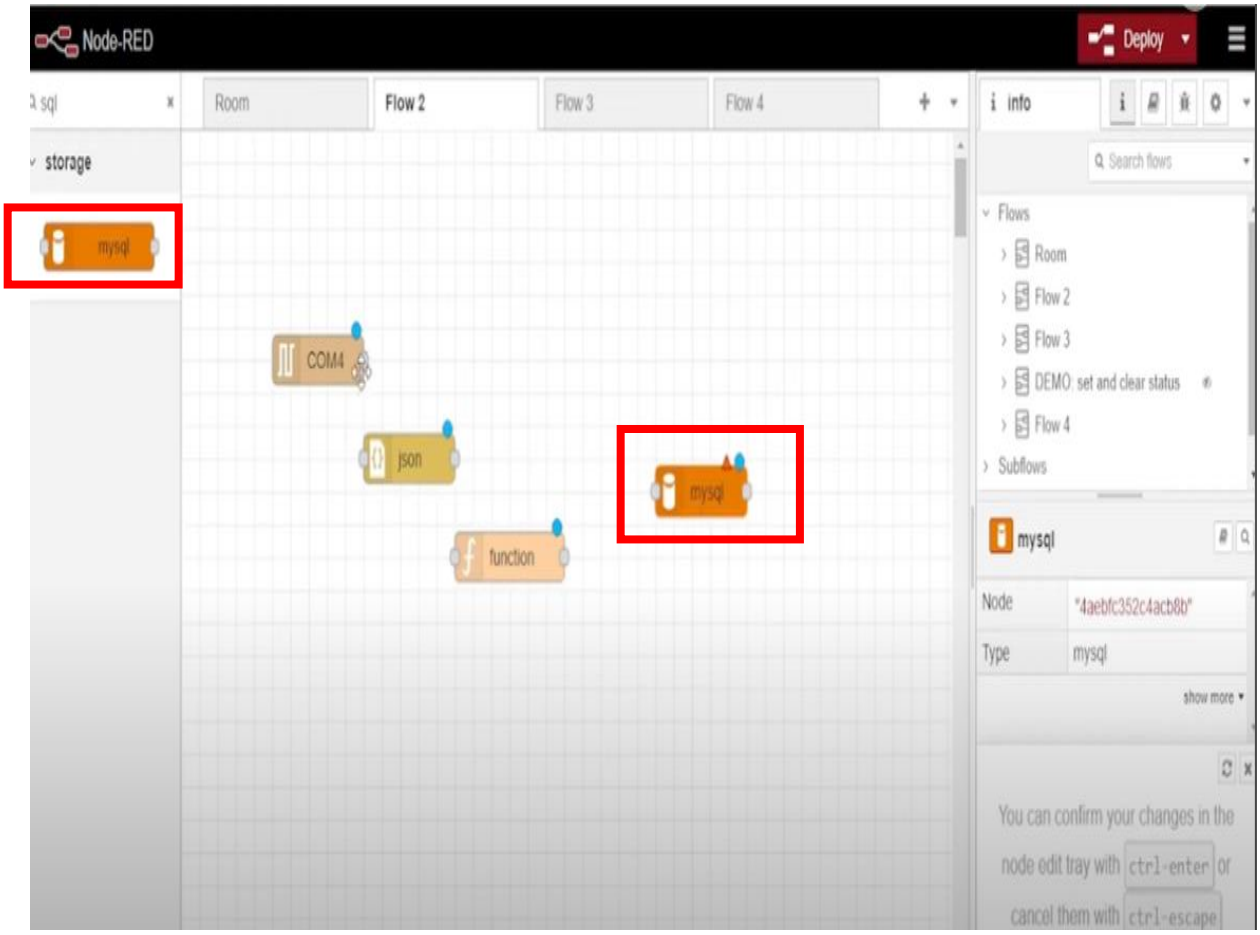
Note:

The Arduino UNO program variables and the MySQL table column names should be same

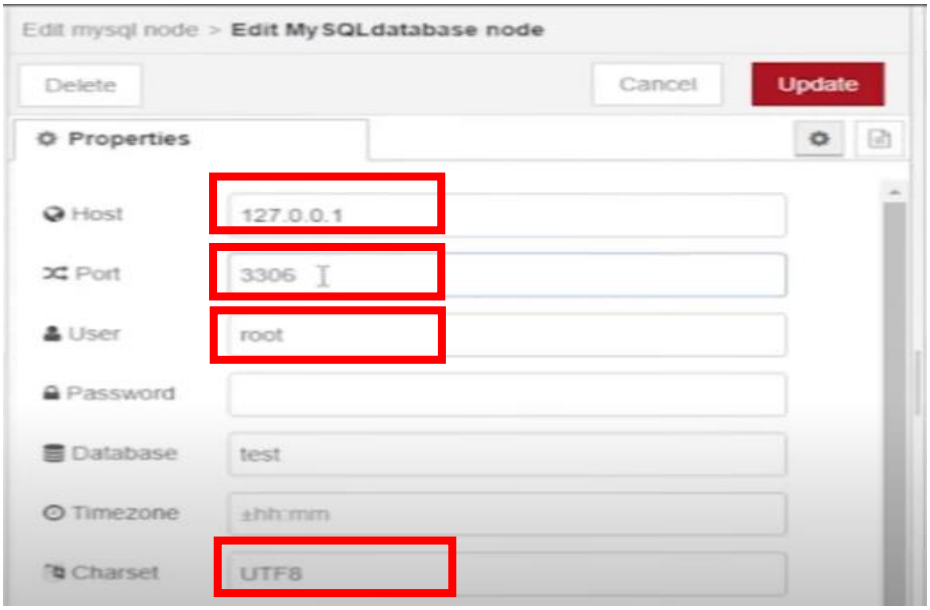
Code to be written in function pallet:

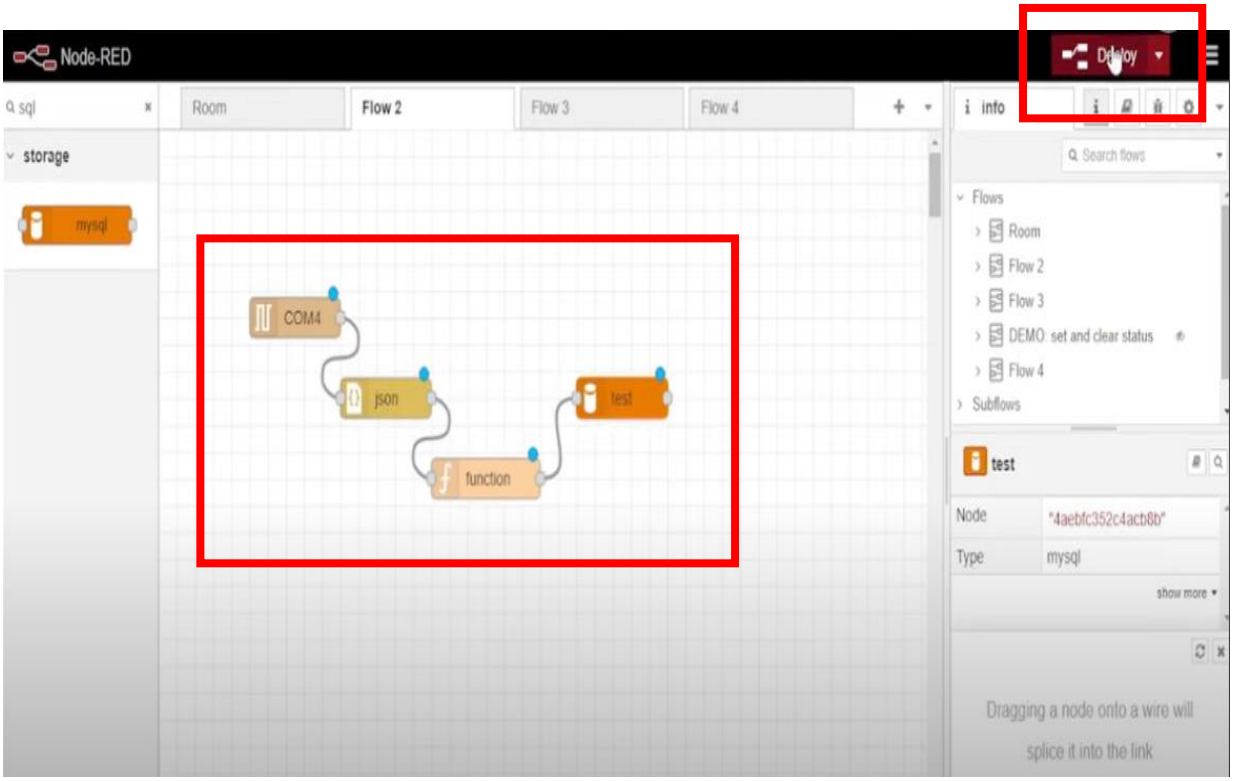
```
var value = JSON.parse(JSON.stringify(msg.payload)); //converting string to
                                                     // JSON object
value = msg;
var sensor1 = msg.payload.inches; //adding value to the payload
var sensor2 = msg.payload.cm;     //adding value to the payload
msg.payload = [sensor1, sensor2]; //adding value to the payload
msg.topic = 'INSERT INTO distance(inches,cm) values (?,?);'; //query to insert
return msg;
```

d. The fourth pallet is **mysql to include.** Which has to be installed in **manage pallets option**(if not available). The name of the pallet is ***node-red-node-mysql.***



i) Doulble click on the mysql pallet to configure.
ii) The **database name** (test) has to be mentioned
iii) Enter **root** information
iv) Enter the the port number **3306** of xampp
v) Checkout default charset as Charset as UTF-8
vi) Click on **update** button





e. After considering all the pallets, these pallets are to get interconnected automatically, as shown in the figure above. Finally **deploy button** is used to transfer the data.

6. Verification of the connection with MySQL

   After deploying both the **COM pallet** and **Mysql pallet** should show as **connected**. Which is shown in the figure below, then the data shall transfer successfully from Arduino circuit to MySql. Also we should get **OK** at the end to

confirm for the successful connection establishment from sensor application with MySQL.