LBD ROBOTICS

LET'S BUILD & DEVELOP

17 JUNE 2019 – 17 JULY 2019

INTERNSHIP

2019

Authored by: HARSH PANCHAL

LBD

Let's Build and Develop

https://lbdrobotics.com/

About the company

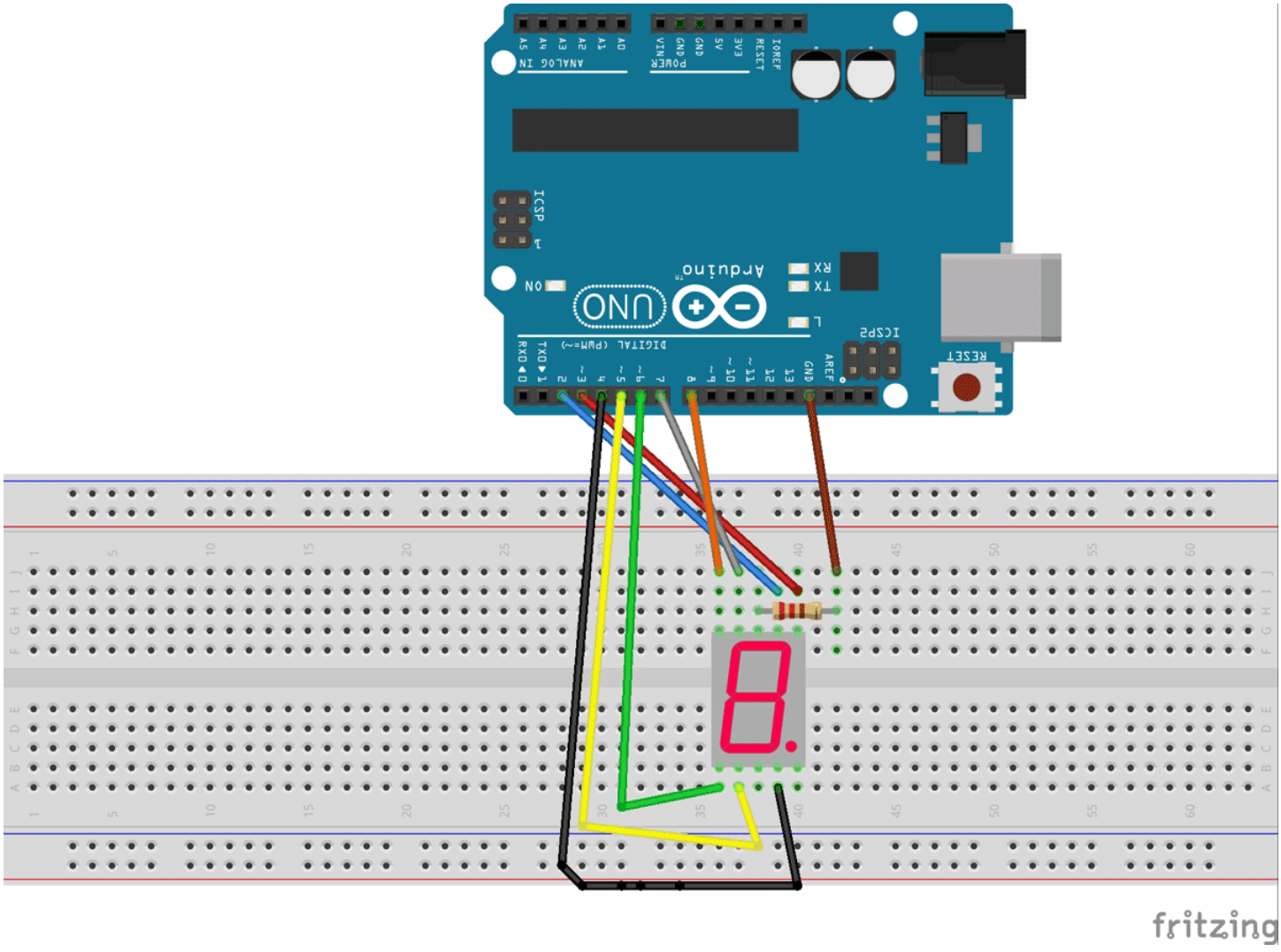
It is the embedded system development company which includes PCB design, IoT deployment and product designing.
This company has its one of the office in Kohat Enclave, New Delhi from where I've done my internship.
The company has its main office in Greater Noida, UP which employs large number of staff.
A unit of the company in Samaypur Badli, New Delhi focused on product design and home appliances.

PROJECT

7 Segment Display Interfacing with Arduino

A seven segment display got its name from the very fact that it got seven illuminating segments. Each of these segments has a LED (Light Emitting Diode), hence the lighting. The LEDs are so fabricated that lighting of each LED is contained to its own segment. The important thing to notice here that the LEDs in any seven segment display are arranged in common anode mode (common positive) or common cathode mode (common negative).

Circuit and Working



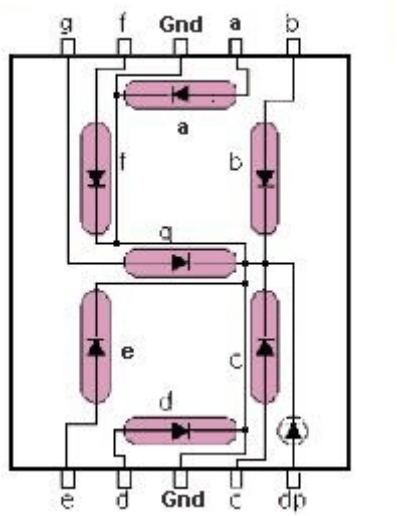
PIN CONNECTIONS

PIN1 or E to PIN 6 of ARDUINO UNO
PIN2 or D to PIN 5
PIN4 or C to PIN 4

PIN5 or H or DP to PIN 9 ///not needed as we are not using decimal point
PIN6 or B to PIN 3
PIN7 or A to PIN 2
PIN9 or F to PIN 7
PIN10 or G to PIN 8
PIN3 or PIN8 or CC to ground through 100Ω resistor.

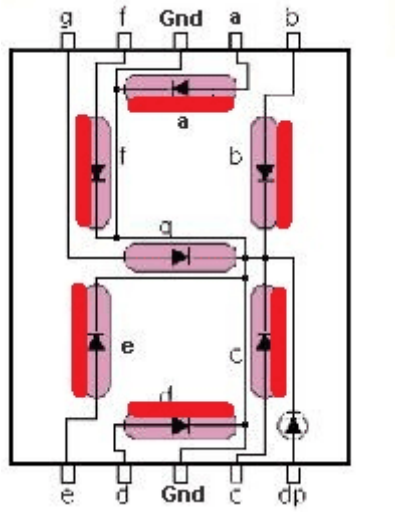
WORKING

Now to understand the working, consider a seven segment display is connected to a port, so say we have connected “A segment of display to PIN0”, “B segment of display to PIN1”, “A segment of display to PIN3”, “A segment of display to PIN4”, “A segment of display to PIN5”, “A segment of display to PIN6”. And is common ground type as shown in figure.



Here the common ground has to be connected to ground for the display to work. One can check each segment of display by using multimeter in diode mode. Each segment should not be power with a voltage greater than 4v, if did the display will be damaged permanently. For avoiding this a common resistor can be provider at common terminal, as shown in circuit diagram.

Now, if we want to display a “0” in this display as shown in below figure.



We need to turn the LEDs of segments “A, B, C, D, E F”, so we need to power PIN0, PIN1, PIN2, PIN3, PIN4 and PIN5. So every time we need a “0”, we need to power all the pins mentioned.

Now, if we want “1” on display

CODE

```
#define segA 2//connecting segment A to PIN2
#define segB 3// connecting segment B to PIN3
#define segC 4// connecting segment C to PIN4
#define segD 5// connecting segment D to PIN5
#define segE 6// connecting segment E to PIN6
#define segF 7// connecting segment F to PIN7
#define segG 8// connecting segment G to PIN8

        int COUNT=0;//count integer for 0-9 increment
void setup()
{
         for (int i=2;i<9;i++)
         {
                  pinMode(i, OUTPUT);// taking all pins from 2-8 as output
         }
}

void loop()
{
switch (COUNT)
        {

        case 0://when count value is zero show”0” on disp
        digitalWrite(segA, HIGH);
        digitalWrite(segB, HIGH);
```

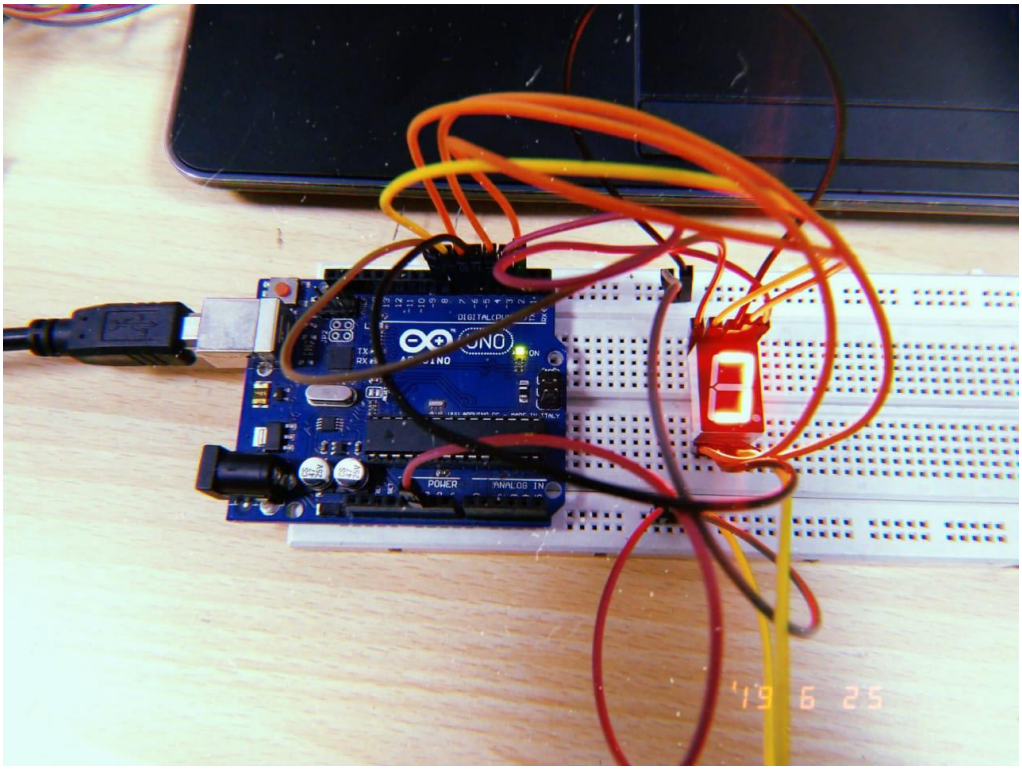
```
digitalWrite(segC, HIGH);
digitalWrite(segD, HIGH);
digitalWrite(segE, HIGH);
digitalWrite(segF, HIGH);
digitalWrite(segG, LOW);
break;

case 1:// when count value is 1 show”1” on disp
digitalWrite(segA, LOW);
digitalWrite(segB, HIGH);
digitalWrite(segC, HIGH);
digitalWrite(segD, LOW);
digitalWrite(segE, LOW);
digitalWrite(segF, LOW);
digitalWrite(segG, LOW);
break;

case 2:// when count value is 2 show”2” on disp
digitalWrite(segA, HIGH);
digitalWrite(segB, HIGH);
digitalWrite(segC, LOW);
digitalWrite(segD, HIGH);
digitalWrite(segE, HIGH);
digitalWrite(segF, LOW);
digitalWrite(segG, HIGH);
```

```
break;

case 3:// when count value is 3 show”3” on disp
digitalWrite(segA, HIGH);
digitalWrite(segB, HIGH);
digitalWrite(segC, HIGH);
digitalWrite(segD, HIGH);
digitalWrite(segE, LOW);
digitalWrite(segF, LOW);
digitalWrite(segG, HIGH);
break;

case 4:// when count value is 4 show”4” on disp
digitalWrite(segA, LOW);
digitalWrite(segB, HIGH);
digitalWrite(segC, HIGH);
digitalWrite(segD, LOW);
digitalWrite(segE, LOW);
digitalWrite(segF, HIGH);
digitalWrite(segG, HIGH);
break;

case 5:// when count value is 5 show”5” on disp
digitalWrite(segA, HIGH);
digitalWrite(segB, LOW);
```

```
digitalWrite(segC, HIGH);
digitalWrite(segD, HIGH);
digitalWrite(segE, LOW);
digitalWrite(segF, HIGH);
digitalWrite(segG, HIGH);
break;

case 6:// when count value is 6 show”6” on disp
digitalWrite(segA, HIGH);
digitalWrite(segB, LOW);
digitalWrite(segC, HIGH);
digitalWrite(segD, HIGH);
digitalWrite(segE, HIGH);
digitalWrite(segF, HIGH);
digitalWrite(segG, HIGH);
break;

case 7:// when count value is 7 show”7” on disp
digitalWrite(segA, HIGH);
digitalWrite(segB, HIGH);
digitalWrite(segC, HIGH);
digitalWrite(segD, LOW);
digitalWrite(segE, LOW);
digitalWrite(segF, LOW);
digitalWrite(segG, LOW);
```

```
break;


case 8:// when count value is 8 show”8” on disp
digitalWrite(segA, HIGH);
digitalWrite(segB, HIGH);
digitalWrite(segC, HIGH);
digitalWrite(segD, HIGH);
digitalWrite(segE, HIGH);
digitalWrite(segF, HIGH);
digitalWrite(segG, HIGH);
break;

case 9:// when count value is 9 show”9” on disp
digitalWrite(segA, HIGH);
digitalWrite(segB, HIGH);
digitalWrite(segC, HIGH);
digitalWrite(segD, HIGH);
digitalWrite(segE, LOW);
digitalWrite(segF, HIGH);
digitalWrite(segG, HIGH);
break;

break;
}
if (COUNT<10)
```

```
{
        COUNT++;
        delay(1000);///increment count integer for every second
}
if (COUNT==10)
{
        COUNT=0;// if count integer value is equal to 10, reset it to zero.
        delay(1000);
}
}
```

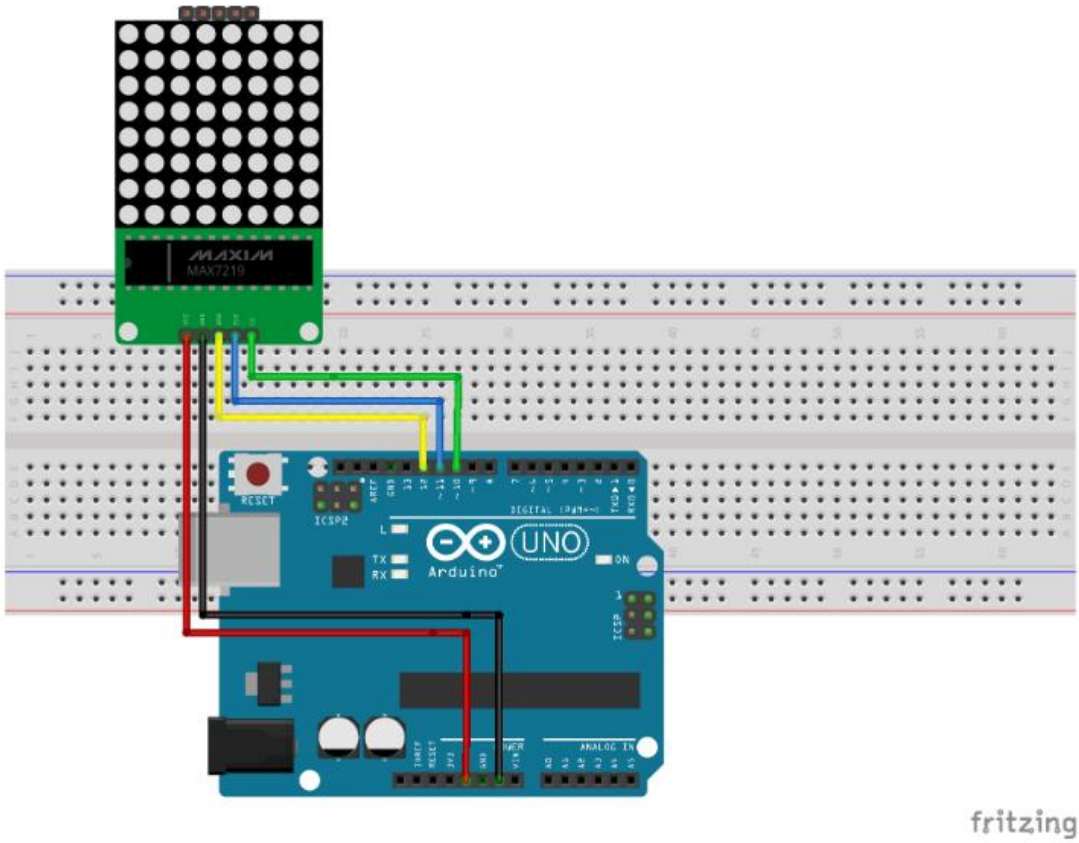


PROJECT

Print on 8x8 matrix Led Using ARDUINO UNO

The 8×8 LED matrix displays are typically used for displaying Short text, counters, symbols etc. They can be daisy chained to form large displays which can be used for the display of scrolling texts, logos among others .

Circuit and Working



PIN CONNECTIONS

LED Matrix - Arduino

CS - D11

CLK- D10

DIN- D12

GND- GND

VCC - 5V

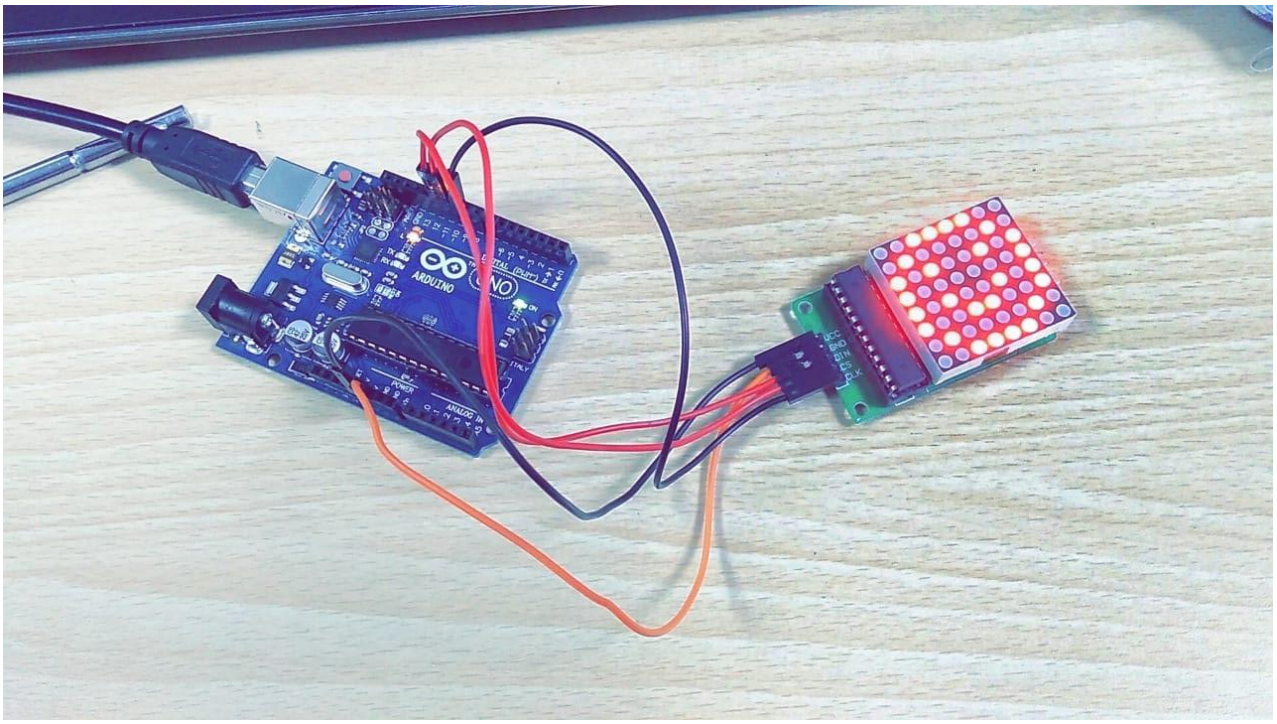
WORKING

The 8×8 LED matrix is made up of 64 LEDs, graphics, text, and symbols are displayed by turning certain LEDs on while other LEDs are turned off. Instead of writing the byte array to instruct the code on which LED to turn on or off when a particular text or symbol is to be displayed, we can generate the byte array with the help of a simple software called pixeltomatrix. It generates byte arrays for LED matrix after the design has been done to show which LED will stay on and Which LED will go off to properly represent the Image, symbol, or text.

CODE

```
#include <LedControl.h>

int DIN = 12;
int CS =  11;
int CLK = 10;


LedControl lc=LedControl(DIN,CLK,CS,0);

void setup(){
 lc.shutdown(0,false);       //The MAX72XX is in power-saving mode on startup
 lc.setIntensity(0,15);      // Set the brightness to maximum value
 lc.clearDisplay(0);         // and clear the display
}

void loop(){

    byte smile[8]=   {0x3C,0x42,0xA5,0x81,0xA5,0x99,0x42,0x3C};
    byte neutral[8]= {0x3C,0x42,0xA5,0x81,0xBD,0x81,0x42,0x3C};
    byte frown[8]=   {0x3C,0x42,0xA5,0x81,0x99,0xA5,0x42,0x3C};

    printByte(smile);

    delay(10000);

    printByte(neutral);

    delay(1000);

    printByte(frown);

    delay(1000);

    lc.clearDisplay(0);

    delay(1000);
}

void printByte(byte character [])
{
  int i = 0;
```

```
  for(i=0;i<8;i++)
  {
    lc.setRow(0,i,character[i]);
  }
}
```

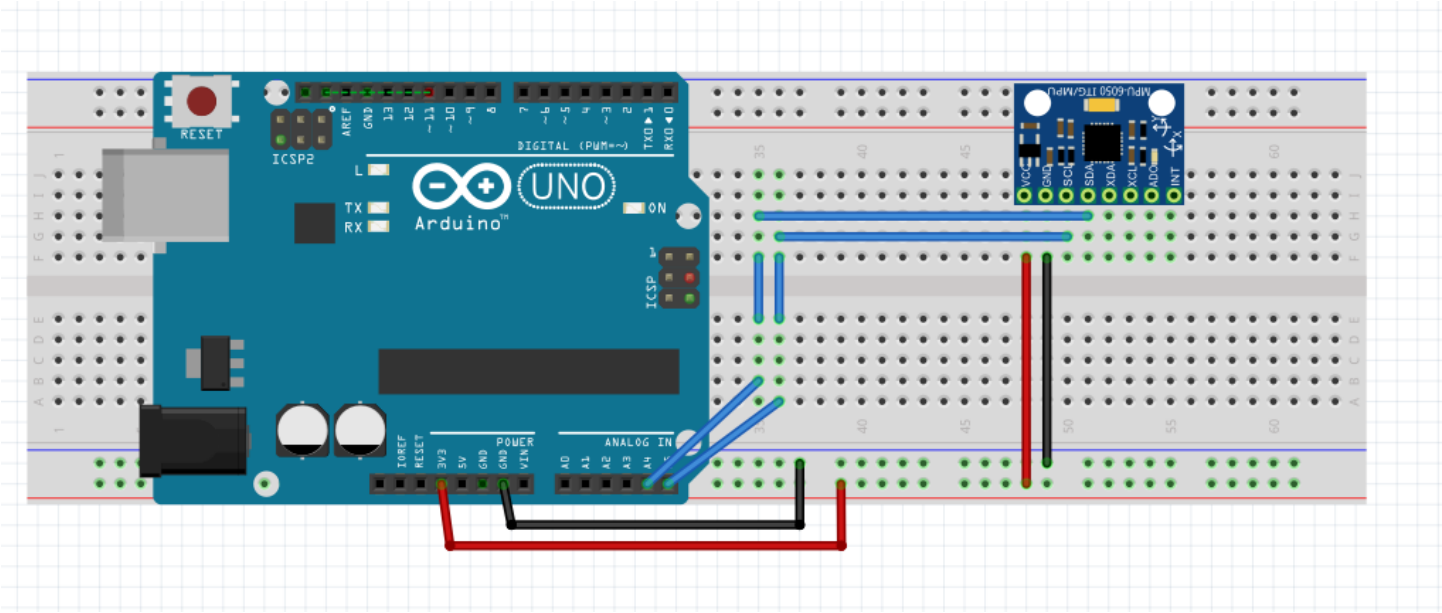


PROJECT

How to use the accelerometer- gyroscope GY-521 with Arduino

The accelerometer measures the acceleration along one direction, while the gyroscope measures the angular acceleration on one axis.

Circuit and Working



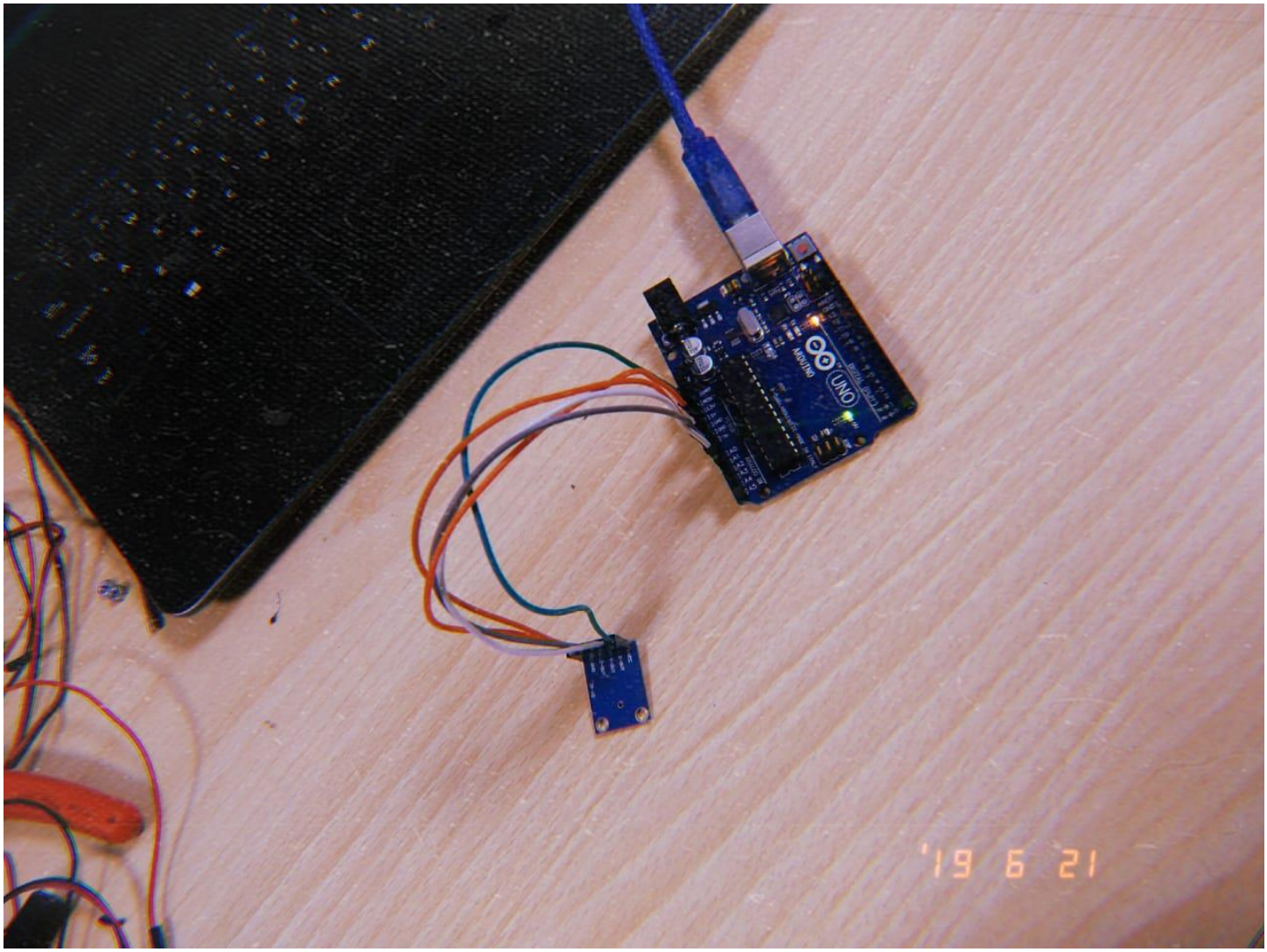
PIN CONNECTIONS
AND
WORKING

VCC -> 3.3 V / 5 V (better)
GND -> GND
SCL -> A5
SDA -> A4
XDA ->
XCL ->
ADO ->
INT ->

The analogic pins are not set on INPUT because it's their default setting. The values read by the analogic pins will be sent to the serial port.
Open the S*erial Monitor*, move the sensor and try to see how the values change.
Accelerometers can be used for fun projects, for example to realize a game controller.

CODE

```
#include<Wire.h>
const int MPU=0x68;
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

void setup(){
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.begin(9600);
}
void loop(){
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,12,true);
  AcX=Wire.read()<<8|Wire.read();
  AcY=Wire.read()<<8|Wire.read();
  AcZ=Wire.read()<<8|Wire.read();
  GyX=Wire.read()<<8|Wire.read();
  GyY=Wire.read()<<8|Wire.read();
  GyZ=Wire.read()<<8|Wire.read();

  Serial.print("Accelerometer: ");
  Serial.print("X = "); Serial.print(AcX);
  Serial.print(" | Y = "); Serial.print(AcY);
  Serial.print(" | Z = "); Serial.println(AcZ);

  Serial.print("Gyroscope: ");
  Serial.print("X = "); Serial.print(GyX);
  Serial.print(" | Y = "); Serial.print(GyY);
  Serial.print(" | Z = "); Serial.println(GyZ);
  Serial.println(" ");
  delay(333);
}
```

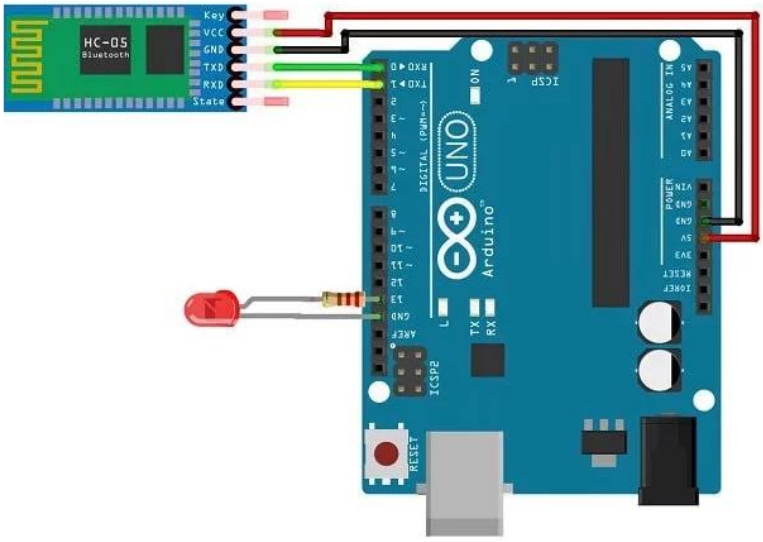

ARDUINO
UNO
'19 6 21

PROJECT

Bluetooth controlled led using Arduino

HC 05/06 works on serial communication. The Android app is designed to send serial data to the Arduino Bluetooth module when a button is pressed on the app. The Arduino Bluetooth module at the other end receives the data and sends it to the Arduino through the TX pin of the Bluetooth module (connected to RX pin of Arduino). The code uploaded to the Arduino checks the received data and compares it. If the received data is 1, the LED turns ON. The LED turns OFF when the received data is 0. You can open the serial monitor and watch the received data while connecting.

Circuit and Working



PIN CONNECTIONS

RX (Pin 0) ———> TX
TX (Pin 1) ———> RX
5V ———> VCC
GND ———> GND

Connect an LED positive to pin 13 of the Arduino through a resistance (valued between 220Ω–1KΩ). Connect its negative to GND, and you're done with the circuit!

WORKING

1. Turn ON the HC 05/06 Bluetooth module by powering the Arduino.

2. Scan your smartphone for available devices.
3. 3. Pair your smartphone to the HC 05/06 by entering default password 1234 OR 0000.

- Install the application on your Android device. https://play.google.com/store/apps/details?id=com.circuitmagic.arduinobluetooth
- Open the application.

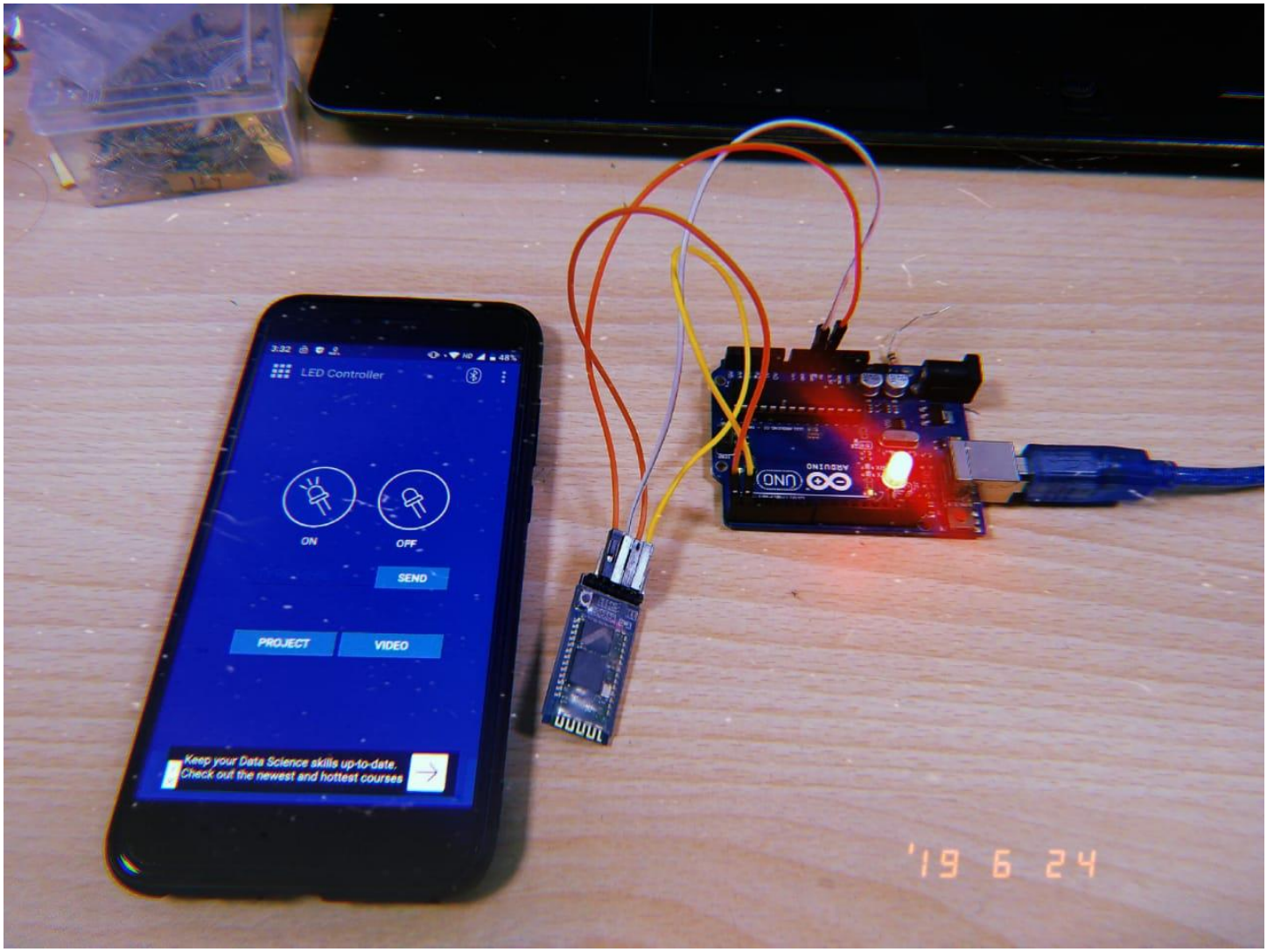
- Press "paired devices".
- Select your Bluetooth module from the list (HC-05/06)

- After connecting successfully, press the ON button to turn the LED on and the OFF button to turn the LED off.
- Disconnect the button to disconnect the Bluetooth module.

CODE

```
#include "Wire.h" // This library allows you to communicate with I2C devices.

const int MPU_ADDR = 0x68; // I2C address of the MPU-6050. If AD0 pin is set to HIGH, the I2C address will be 0x69.

int16_t accelerometer_x, accelerometer_y, accelerometer_z; // variables for accelerometer raw data
int16_t gyro_x, gyro_y, gyro_z; // variables for gyro raw data
int16_t temperature; // variables for temperature data

char tmp_str[7]; // temporary variable used in convert function

char* convert_int16_to_str(int16_t i) { // converts int16 to string. Moreover, resulting strings will have the same length in the debug monitor.
  sprintf(tmp_str, "%6d", i);
  return tmp_str;
}

void setup() {
  Serial.begin(9600);
  Wire.begin();
  Wire.beginTransmission(MPU_ADDR); // Begins a transmission to the I2C slave (GY-521 board)
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
}
void loop() {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H) [MPU-6000 and MPU-6050
```

```
Register Map and Descriptions Revision 4.2, p.40]
  Wire.endTransmission(false); // the parameter indicates that the Arduino will send a restart. As a result, the connection is kept active.
  Wire.requestFrom(MPU_ADDR, 7*2, true); // request a total of 7*2=14 registers

  // "Wire.read()<<8 | Wire.read();" means two registers are read and stored in the same variable
  accelerometer_x = Wire.read()<<8 | Wire.read(); // reading registers: 0x3B (ACCEL_XOUT_H) and 0x3C (ACCEL_XOUT_L)
  accelerometer_y = Wire.read()<<8 | Wire.read(); // reading registers: 0x3D (ACCEL_YOUT_H) and 0x3E (ACCEL_YOUT_L)
  accelerometer_z = Wire.read()<<8 | Wire.read(); // reading registers: 0x3F (ACCEL_ZOUT_H) and 0x40 (ACCEL_ZOUT_L)
  temperature = Wire.read()<<8 | Wire.read(); // reading registers: 0x41 (TEMP_OUT_H) and 0x42 (TEMP_OUT_L)
  gyro_x = Wire.read()<<8 | Wire.read(); // reading registers: 0x43 (GYRO_XOUT_H) and 0x44 (GYRO_XOUT_L)
  gyro_y = Wire.read()<<8 | Wire.read(); // reading registers: 0x45 (GYRO_YOUT_H) and 0x46 (GYRO_YOUT_L)
  gyro_z = Wire.read()<<8 | Wire.read(); // reading registers: 0x47 (GYRO_ZOUT_H) and 0x48 (GYRO_ZOUT_L)

  // print out data
  Serial.print("aX = "); Serial.print(convert_int16_to_str(accelerometer_x));
  Serial.print(" | aY = "); Serial.print(convert_int16_to_str(accelerometer_y));
  Serial.print(" | aZ = "); Serial.print(convert_int16_to_str(accelerometer_z));
  // the following equation was taken from the documentation [MPU-6000/MPU-6050 Register Map and Description, p.30]
  Serial.print(" | tmp = "); Serial.print(temperature/340.00+36.53);
  Serial.print(" | gX = "); Serial.print(convert_int16_to_str(gyro_x));
  Serial.print(" | gY = "); Serial.print(convert_int16_to_str(gyro_y));
  Serial.print(" | gZ = "); Serial.print(convert_int16_to_str(gyro_z));
  Serial.println();

  // delay
  delay(1000);
}
```

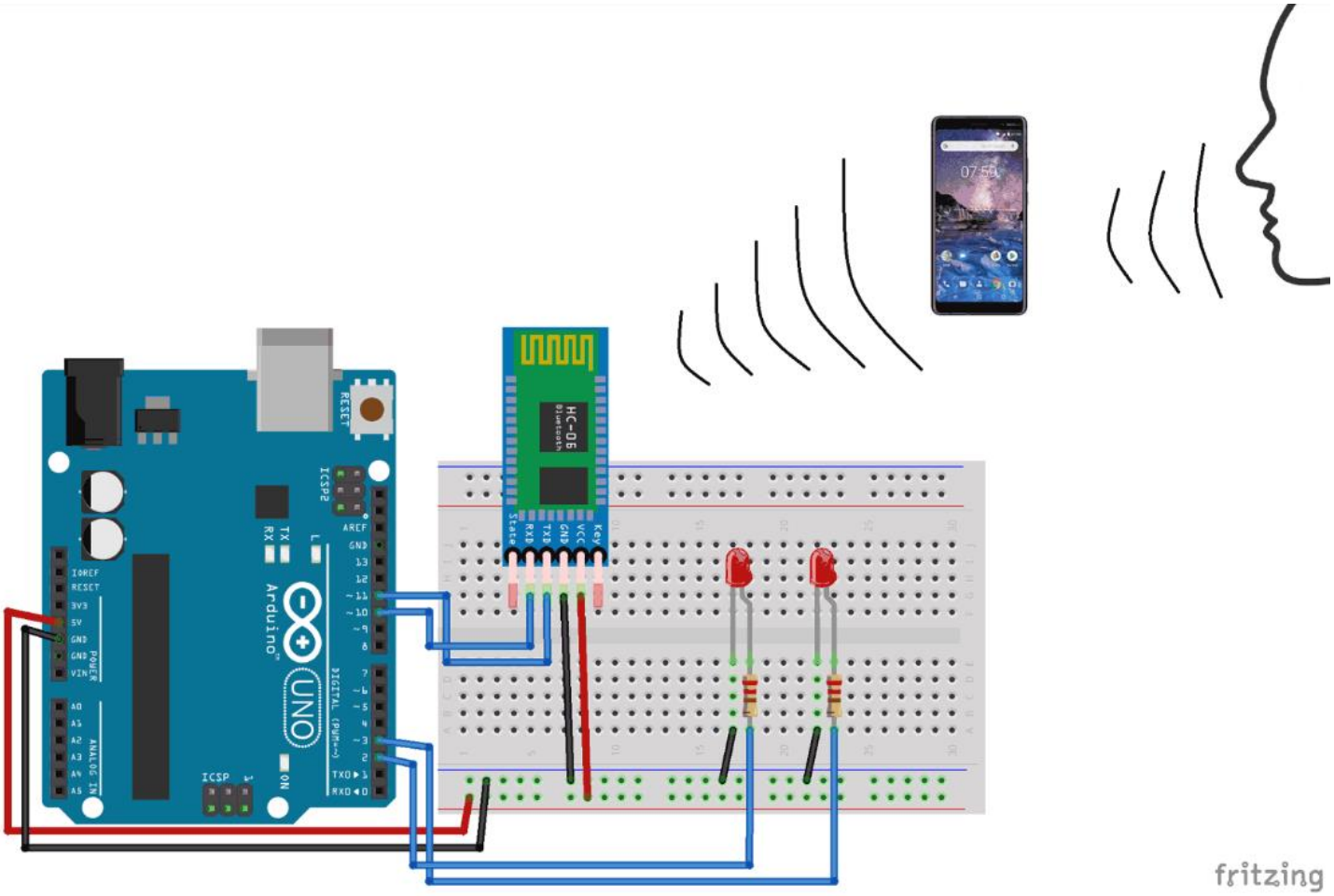

LED Controller
ON
OFF
SEND
PROJECT
VIDEO
Keep your Data Science skills up-to-date.
Check out the newest and hottest courses
UNO
'19 6 24

PROJECT

Voice Controlled LEDs using Arduino and Bluetooth

Controlling LEDs with voice command seems to be a difficult task, but it's easy and you can quickly build it. We just need an Arduino UNO to serially communicate with HC-06 Bluetooth module and a smartphone to send voice command to Bluetooth module HC-06. For receiving voice command we are using "Arduino Bluetooth Voice Controller" android app which you can download from play store

Circuit and Working



PIN CONNECTIONS

connect TxD to 11 (blu module to arduino)
connect RxD to 10 (blu module to arduino)
vcc to 5v and gnd to gnd

led connections:
red led to pin 2

green to pin 3
and negative to gnd

WORKING

Step 1:- Connect all components as per the circuit diagram; disconnect Rx and Tx pins while uploading the code.

Step 2:- Download the app called “Arduino Bluetooth Voice Controller” which is free on play store.

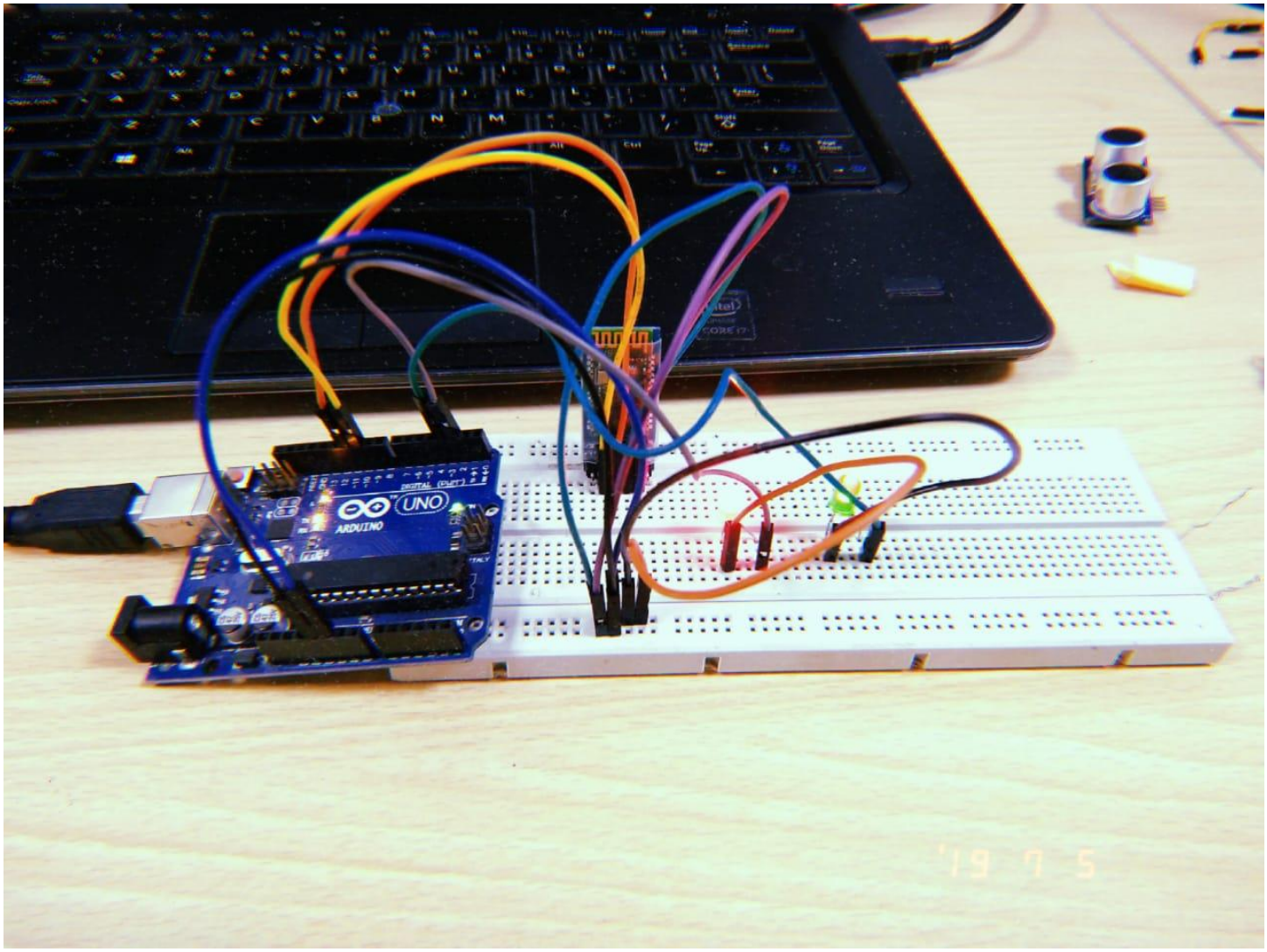
Step 3:- Open the app and follow the image below, like first click on “connect to Bluetooth device” and select your Bluetooth module and check if it is connected or not. Then click on the mic icon to speak and send the voice command to the HC-06 module.
Note: when you are connecting your Bluetooth module for the first time with your smartphone it will ask for the passcode, use 0000 or 1234.
Step 4:- After setting up all the things, you just have to send the voice command by using the app which is further sent to Bluetooth module HC-06 and the HC-06 serially communicate with the Arduino UNO and then the task is performed as per the command. The below shows the command and the action to be performed by the command:

S. No.	Command	Action
1.	all LED turn on	Both Red and Green LED turns ON
2.	all LED turn off	Both Red and Green LED turns OFF
3.	turn on Red LED	Red LED turns ON
4.	turn on green LED	Green LED turns ON
5.	turn off red LED	Red LED turns OFF
6.	turn off green LED	Green LED turns OFF

CODE

```
#include <SoftwareSerial.h>
String value;
int TxD = 11;
int RxD = 10;
int servoposition;
SoftwareSerial bluetooth(TxD, RxD);

void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  Serial.begin(9600);       // start serial communication at 9600bps
  bluetooth.begin(9600);
}

void loop() {
  Serial.println(value);
 if (bluetooth.available())
```

```
  {
   value = bluetooth.readString();

   if (value == "all LED turn on"){
   digitalWrite(2, HIGH);
   digitalWrite(3, HIGH);
     }

   if (value == "all LED turn off"){
     digitalWrite(2, LOW);
     digitalWrite(3, LOW);
     }

   if (value == "turn on Red LED"){
   digitalWrite(2, HIGH);
     }

   if (value == "turn on green LED"){
     digitalWrite(3, HIGH);
     }

   if (value == "turn off red LED"){
   digitalWrite(2, LOW);
     }

   if (value == "turn off green LED"){
     digitalWrite(3, LOW);
     }

 }

}
```

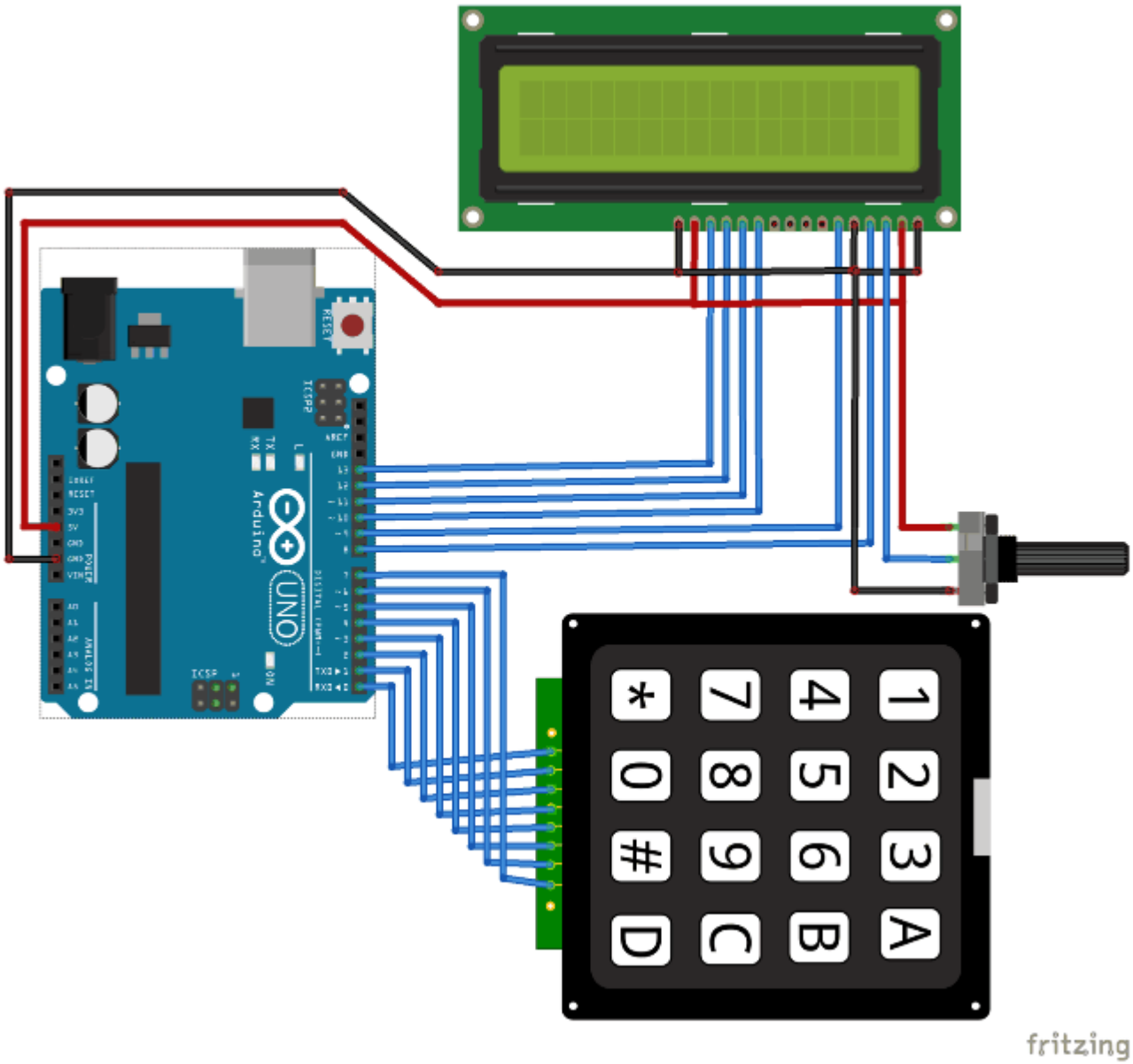

UNO
ARDUINO

PROJECT

CALCULATOR Using ARDUINO UNO

I build my own calculator with Arduino. The values can be sent in through a **keypad (4×4 keypad)** and result can be viewed on a **LCD screen** (16×2 Dot-matrix). This calculator could perform simple operations like Addition, Subtraction, Multiplication and Division with whole numbers

Circuit and Working



PIN CONNECTIONS

Arduino Pin Name:	Connected to:

D0	1st pin of the keyboard
D1	2nd pin of the keyboard
D2	3rd pin of the keyboard
D3	4th pin of the keyboard
D4	5th pin of the keyboard
D5	6th pin of the keyboard
D6	7th pin of the keyboard
D7	8th pin of the keyboard
D8	Register select pin of LCD (pin 4)
D10	Data pin 4 (pin 11)
D11	Data pin 4 (pin 11)
D12	Data pin 4 (pin 11)
D13	Data pin 4 (pin 11)
+5V	Connected to Vdd pin of LCD (pin 2)
Ground	Connected to Vss,Vee and RW pin of LCD (pin 1,3 and 5)

WORKING

Make the connections as per circuit diagram and upload the code below. If it shows error make sure you have added the library as per the instruction given above. You can also try the simulation to check if the problem is with your hardware. If everything is done as it's supposed to be, then your hardware will look something like this below with the LCD displaying this

Character on Keypad	**Assumed to be**
"A"	Addition (+)
"B"	Subtraction (-)
"C"	Multiplication (*)
"D"	Division (/)
"*"	Clear (C)
"#"	Equals (=)

CODE

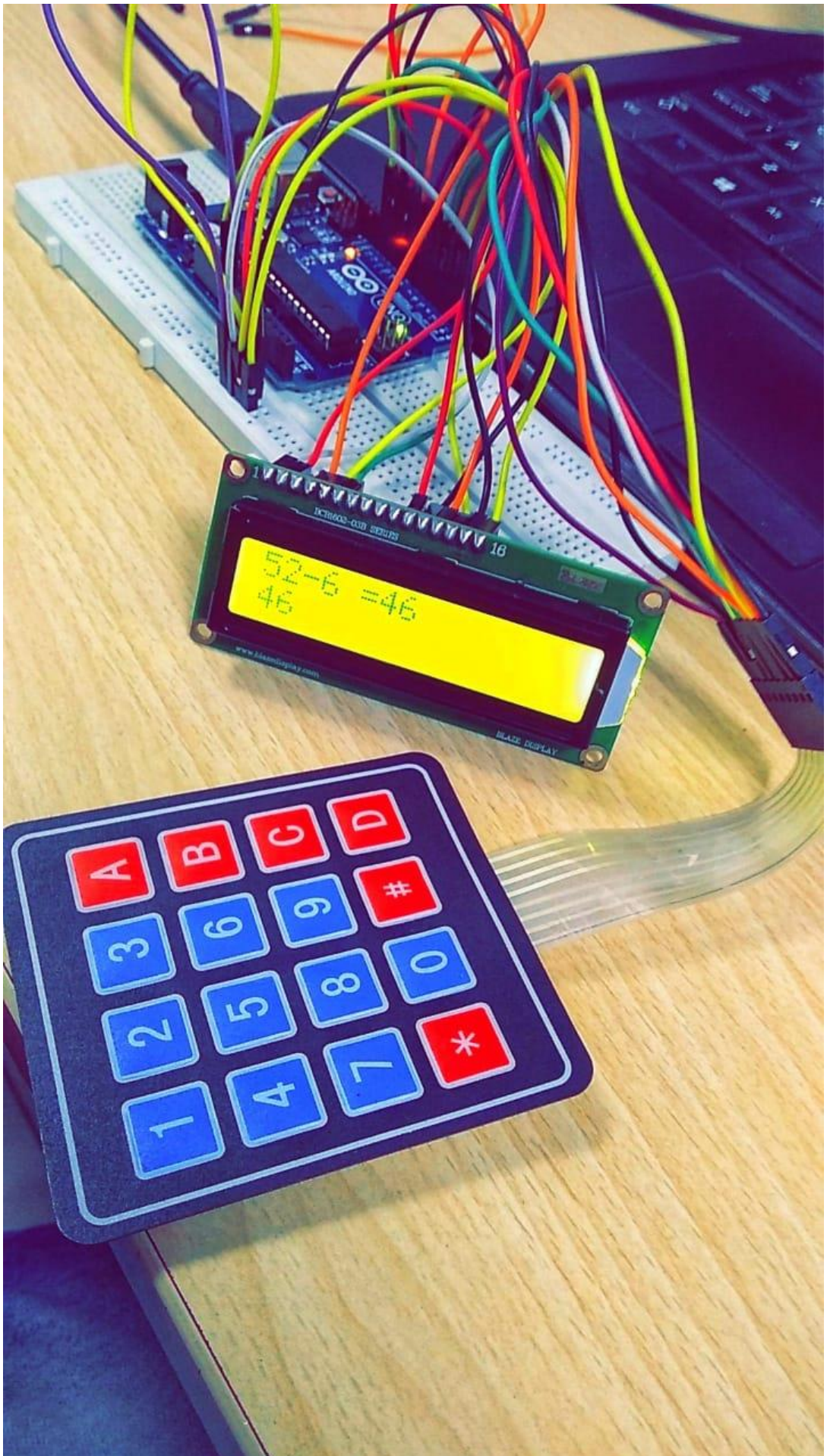
```
#include <LiquidCrystal.h>
#include <Keypad.h>

const byte ROWS = 4; // Four rows
const byte COLS = 4; // Three columns

// Define the Keymap
char keys[ROWS][COLS] = {

  {'7','8','9','D'},

  {'4','5','6','C'},

  {'1','2','3','B'},
```

```
  {'*','0','#','A'}

};

byte rowPins[ROWS] = { 0, 1, 2, 3 };// Connect keypad ROW0, ROW1, ROW2 and ROW3 to these Arduino pins.
byte colPins[COLS] = { 4, 5, 6, 7 }; // Connect keypad COL0, COL1 and COL2 to these Arduino pins.

Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ); //  Create the Keypad

const int rs = 8, en = 9, d4 = 10, d5 = 11, d6 = 12, d7 = 13; //Pins to which LCD is connected
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

 long Num1,Num2,Number;
 char key,action;
 boolean result = false;

void setup() {
  lcd.begin(16, 2); //We are using a 16*2 LCD display
  lcd.print("DIY Calculator"); //Display a intro message
  lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
  lcd.print("-CircuitDigest"); //Display a intro message

   delay(2000); //Wait for display to show info
    lcd.clear(); //Then clean it
}

void loop() {

key = kpd.getKey(); //storing pressed key value in a char

if (key!=NO_KEY)
DetectButtons();

if (result==true)
CalculateResult();

DisplayResult();
}

void DetectButtons()
{
    lcd.clear(); //Then clean it
   if (key=='*') //If cancel Button is pressed
   {Serial.println ("Button Cancel"); Number=Num1=Num2=0; result=false;}

    if (key == '1') //If Button 1 is pressed
   {Serial.println ("Button 1");
   if (Number==0)
   Number=1;
```

```
else
Number = (Number*10) + 1; //Pressed twice
}

 if (key == '4') //If Button 4 is pressed
{Serial.println ("Button 4");
if (Number==0)
Number=4;
else
Number = (Number*10) + 4; //Pressed twice
}

 if (key == '7') //If Button 7 is pressed
{Serial.println ("Button 7");
if (Number==0)
Number=7;
else
Number = (Number*10) + 7; //Pressed twice
}


if (key == '0')
{Serial.println ("Button 0"); //Button 0 is Pressed
if (Number==0)
Number=0;
else
Number = (Number*10) + 0; //Pressed twice
}

 if (key == '2') //Button 2 is Pressed
{Serial.println ("Button 2");
 if (Number==0)
Number=2;
else
Number = (Number*10) + 2; //Pressed twice
}

 if (key == '5')
{Serial.println ("Button 5");
 if (Number==0)
Number=5;
else
Number = (Number*10) + 5; //Pressed twice
}

 if (key == '8')
{Serial.println ("Button 8");
 if (Number==0)
Number=8;
```

```
  else
  Number = (Number*10) + 8; //Pressed twice
  }


  if (key == '#')
  {Serial.println ("Button Equal");
  Num2=Number;
  result = true;
  }

   if (key == '3')
  {Serial.println ("Button 3");
   if (Number==0)
  Number=3;
  else
  Number = (Number*10) + 3; //Pressed twice
  }

   if (key == '6')
  {Serial.println ("Button 6");
  if (Number==0)
  Number=6;
  else
  Number = (Number*10) + 6; //Pressed twice
  }

   if (key == '9')
  {Serial.println ("Button 9");
  if (Number==0)
  Number=9;
  else
  Number = (Number*10) + 9; //Pressed twice
  }

    if (key == 'A' || key == 'B' || key == 'C' || key == 'D') //Detecting Buttons on Column 4
{
  Num1 = Number;
  Number =0;
  if (key == 'A')
  {Serial.println ("Addition"); action = '+';}
   if (key == 'B')
  {Serial.println ("Subtraction"); action = '-'; }
   if (key == 'C')
  {Serial.println ("Multiplication"); action = '*';}
   if (key == 'D')
  {Serial.println ("Devesion"); action = '/';}

  delay(100);
```

```
  }

}

void CalculateResult()
{
  if (action=='+')
    Number = Num1+Num2;

  if (action=='-')
    Number = Num1-Num2;

  if (action=='*')
    Number = Num1*Num2;

  if (action=='/')
    Number = Num1/Num2;
}

void DisplayResult()
{
  lcd.setCursor(0, 0);   // set the cursor to column 0, line 1
  lcd.print(Num1); lcd.print(action); lcd.print(Num2);

  if (result==true)
  {lcd.print(" ="); lcd.print(Number);} //Display the result

  lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
  lcd.print(Number); //Display the result
}
```

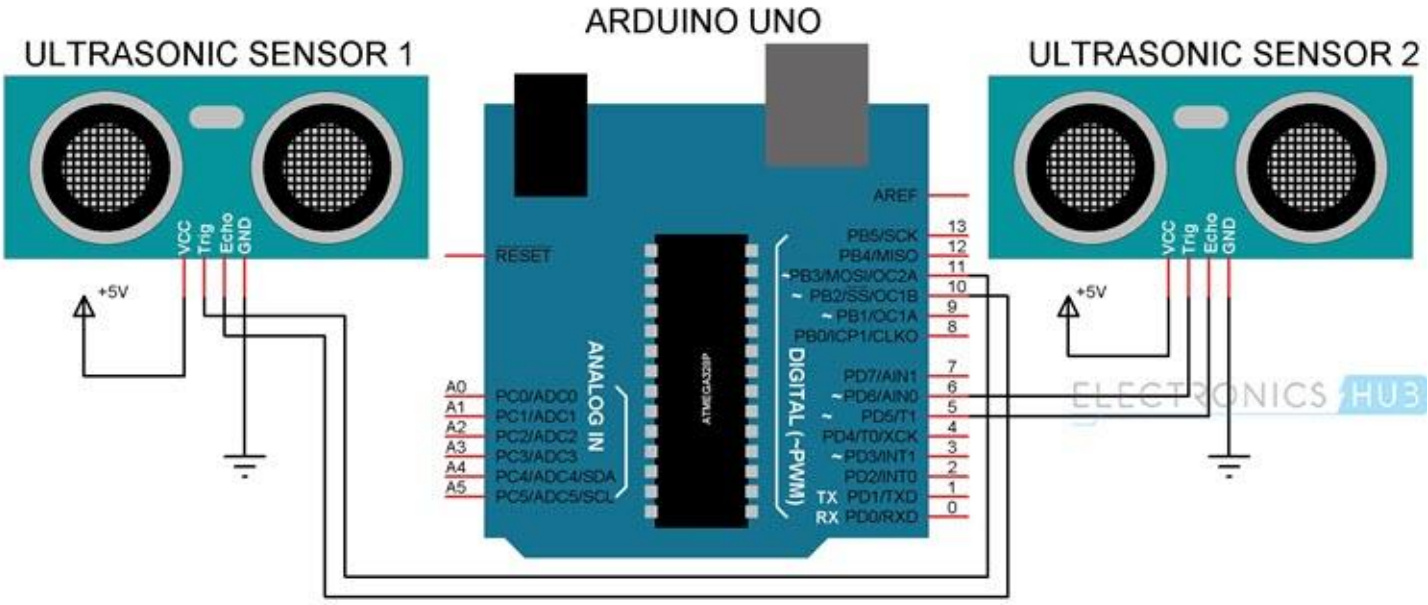

A
B
C
D
3
6
9
#
2
5
8
0
1
4
7
*
16

PROJECT

Arduino based Hand Gesture Control of Computer

In this project, we have implemented a simple Arduino based hand gesture control where you can control few functions of your web browser like switching between tabs, scrolling up and down in web pages, shift between tasks (applications), play or pause a video and increase or decrease the volume (in VLC Player) with the help of hand gestures.

Circuit and Working



PIN CONNECTIONS

trigger Pin = 11 (sensor 1)
echo Pin = 10 (sensor 1)
trig Pin = 6 (sensor 2)
echoPin2 = 5 (sensor 2)
vcc to 5v in arduino
gnd to gnd arduino

WORKING

The following are the 5 different hand gestures or actions that I've programmed for demonstration purpose.

Gesture 1: Place your hand in front of the Right Ultrasonic Sensor at a distance (between 15CM to 35CM) for a small duration and move your hand away from the sensor. This gesture will Scroll Down the Web Page or Decrease the Volume.

Gesture 2: Place your hand in front of the Right Ultrasonic Sensor at a distance (between 15CM to 35CM) for a small duration and move your hand towards the sensor. This gesture will Scroll up the Web Page or Increase the Volume.

Gesture 3: Swipe your hand in front of the Right Ultrasonic Sensor. This gesture will move to the Next Tab.

Gesture 4: Swipe your hand in front of the Left Ultrasonic Sensor. This gesture will move to the Previous Tab or Play/Pause the Video.

Gesture 5: Swipe your hand across both the sensors (Left Sensor first). This action will Switch between Tasks.

CODE for arduino

```
const int trigPin1 = 11; // the number of the trigger output pin ( sensor 1 )
const int echoPin1 = 10; // the number of the echo input pin ( sensor 1 )
const int trigPin2 = 6;  // the number of the trigger output pin ( sensor 2 )
const int echoPin2 = 5;  // the number of the echo input pin ( sensor 2 )

//////////////////////////////// variables used for distance calculation
long duration;
int distance1, distance2;
float r;
unsigned long temp=0;
int temp1=0;
int l=0;
////////////////////////////////

void find_distance (void);

// this function returns the value in cm.
/*we should not trigger the both ultrasonic sensor at the same time.
it might cause error result due to the intraction of the both soundswaves.*/
void find_distance (void)
{
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);

  duration = pulseIn(echoPin1, HIGH, 5000);// here this pulsein function wont wait more then 5000us for the ultrasonic sound to came back. (due to this it wont measure more than 60cm)
                                           // it helps this project to use the gesture control in the defined space.
                                           // so that, it will return zero if distance greater then 60m. ( it helps usually if we remove our hands infront of the sensors ).

  r = 3.4 * duration / 2;                  // calculation to get the measurement in cm using the time returned by the pulsein function.
  distance1 = r / 100.00;
  ////////////////////////////////////////upper part for left sensor and lower part for right sensor
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);

  duration = pulseIn(echoPin2, HIGH, 5000);
  r = 3.4 * duration / 2;
```

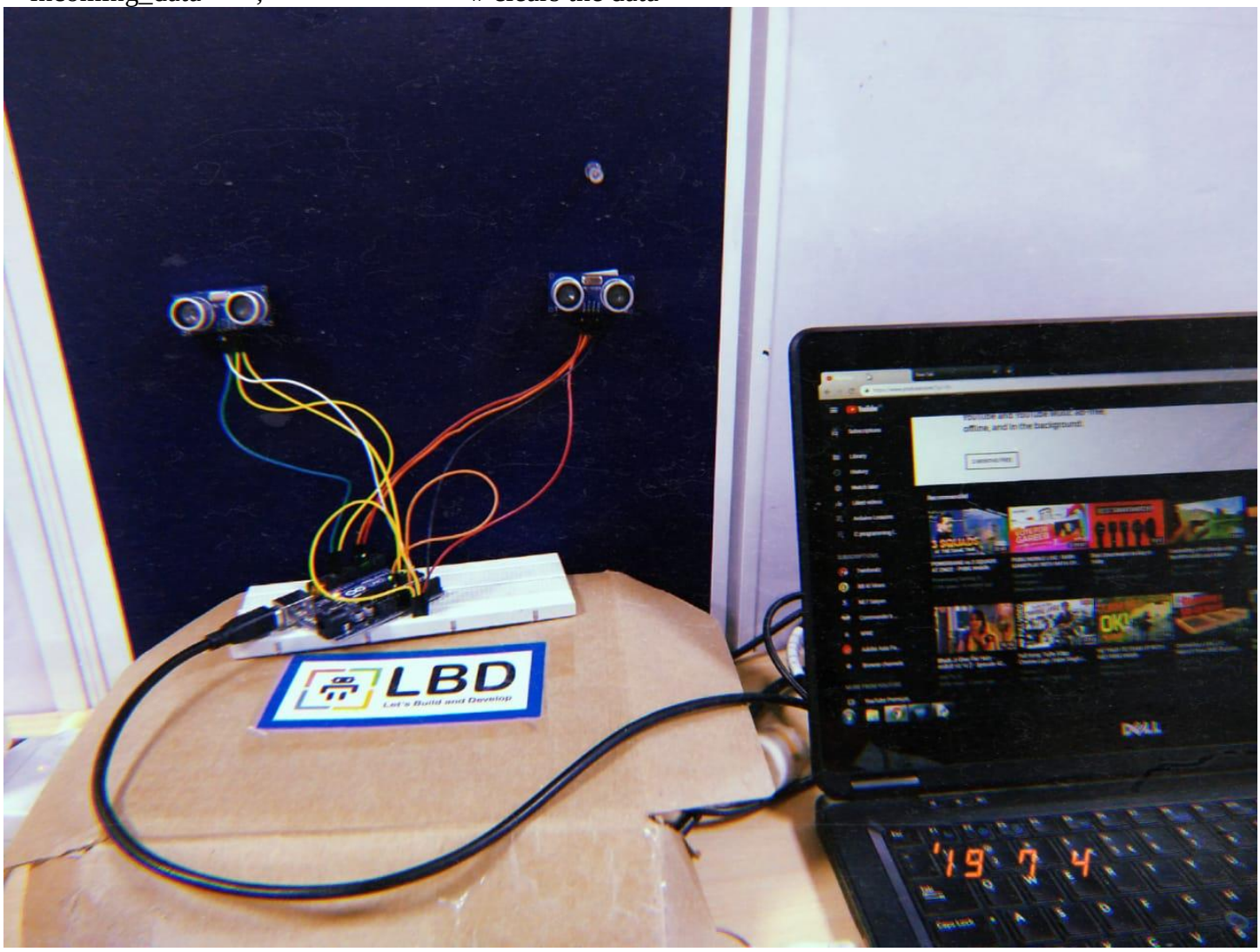
```
  distance2 = r / 100.00;
  delay(100);
}

void setup()
{
  Serial.begin(9600);
  pinMode(trigPin1, OUTPUT); // initialize the trigger and echo pins of both the sensor as input and output:
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  delay (1000);

}

void loop()
{
  find_distance(); // this function will stores the current distance measured by the ultrasonic sensor in the global
variable "distance1 and distance2"
                   // no matter what, the program has to call this "find_distance" function continuously to get the distance
value at all time.

  if(distance2<=35 && distance2>=15) // once if we placed our hands in front of the right sensor in the range
between 15 to 35cm this condition becomes true.
  {
    temp=millis();                  // store the current time in the variable temp. (" millis " Returns the number of
milliseconds since the Arduino board began running the current program )
    while(millis()<=(temp+300))      // this loop measures the distance for another 300 milliseconds. ( it helps to find
the difference between the swipe and stay of our hand in front of the right sensor )
    find_distance();
    if(distance2<=35 && distance2>=15) // this condition will true if we place our hand in front of the right sensor for
more then 300 milli seconds.
    {
     temp=distance2;                        // store the current position of our hand in the variable temp.
     while(distance2<=50 || distance2==0)    // this loop will run untill we removes our hand in front of the right
sensor.
     {
       find_distance();                    // call this function continuously to get the live data.
       if((temp+6)<distance2)               // this condition becomes true if we moves our hand away from the right
sensor (**but in front of it ). here " temp+6 " is for calibration.
       {
       Serial.println("down");              // send "down" serially.
       }
       else if((temp-6)>distance2)          // this condition becomes true if we moves our hand closer to the right sensor.
       {
        Serial.println("up");               // send "up" serially.
       }
     }
    }
```

```
  else                              // this condition becomes true, if we only swipe in front of the right sensor .
  {
    Serial.println("next");              // send "next" serially.
  }
 }

  else if(distance1<=35 && distance1>=15)   // once if we placed our hands in front of the left sensor in the range between 15 to 35cm this condition becomes true.
  {

   temp=millis();

   while(millis()<=(temp+300))
   {
     find_distance();
     if(distance2<=35 && distance2>=15)  // if our hand detects in the right sensor before 300 milli seconds this condition becomes true. ( usually it happens if we swipe our hand from left to right sensor )
     {
       Serial.println("change");         // send "change" serially.
       l=1;                             // store 1 in variable l. ( it avoids the program to enter into the upcoming if condition )
       break;                           // break the loop.
     }
   }

   if(l==0)                             // this condition will become true, only if we swipe our hand in front of left sensor.
   {
   Serial.println("previous");           // send "previous" serially.
   while(distance1<=35 && distance1>=15) // this loop will rotate untill we removes our hand infront of the left sensor. this will avoid not to enter this if condition again.
   find_distance();
   }
   l=0;                                 // make l=0 for the next round.
  }

}
```

CODE for python IDE (2.7)

```
import serial                           # add Serial library for serial communication
import pyautogui                        # add pyautogui library for programmatically controlling the mouse and keyboard.

Arduino_Serial = serial.Serial('com8',9600)      # Initialize serial and Create Serial port object called Arduino_Serial

while 1:
   incoming_data = str (Arduino_Serial.readline()) # read the serial data and print it as line
   print incoming_data                  # print the incoming Serial data


   if 'next' in incoming_data:          # if incoming data is 'next'
```

```
    pyautogui.hotkey('ctrl', 'pgdn')          # perform "ctrl+pgdn" operation which moves to the next tab

if 'previous' in incoming_data:               # if incoming data is 'previous'
    pyautogui.hotkey('ctrl', 'pgup')          # perform "ctrl+pgup" operation which moves to the previous tab

if 'down' in incoming_data:                   # if incoming data is 'down'
    #pyautogui.press('down')                  # performs "down arrow" operation which scrolls down the page
    pyautogui.scroll(-100)

if 'up' in incoming_data:                     # if incoming data is 'up'
    #pyautogui.press('up')                    # performs "up arrow" operation which scrolls up the page
    pyautogui.scroll(100)

if 'change' in incoming_data:                 # if incoming data is 'change'
    pyautogui.keyDown('alt')                  # performs "alt+tab" operation which switches the tab
    pyautogui.press('tab')
    pyautogui.keyUp('alt')

incoming_data = "";                           # clears the data
```

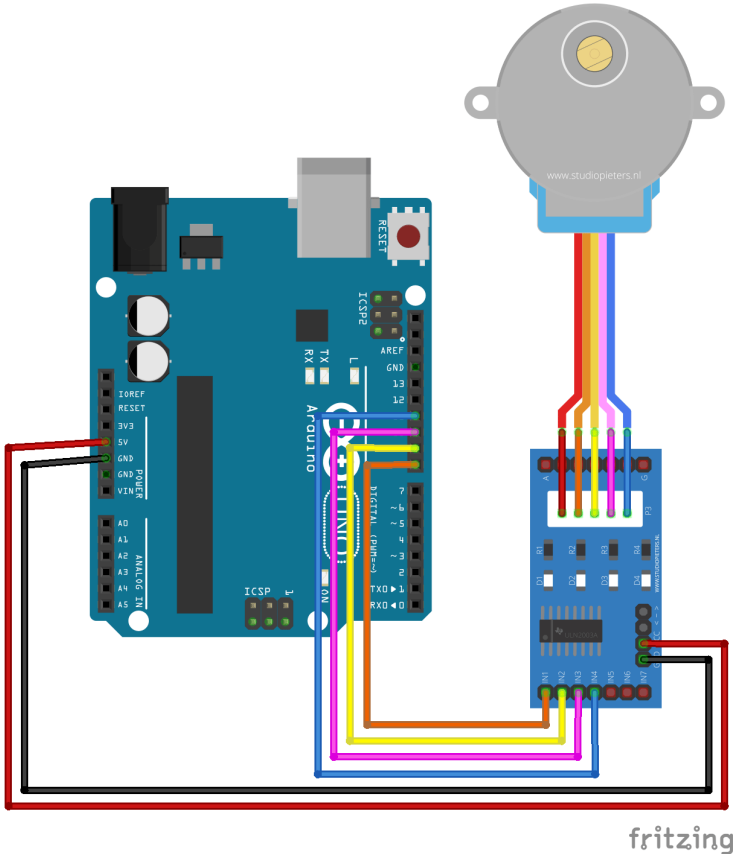


PROJECT

Interfacing Stepper Motor with Arduino Uno

HC 05/06 works on serial communication. The Android app is designed to send serial data to the Arduino Bluetooth module when a button is pressed on the app. The Arduino Bluetooth module at the other end receives the data and sends it to the Arduino through the TX pin of the Bluetooth module (connected to RX pin of Arduino). The code uploaded to the Arduino checks the received data and compares it. If the received data is 1, the LED turns ON. The LED turns OFF when the received data is 0. You can open the serial monitor and watch the received data while connecting.

Circuit and Working



PIN CONNECTIONS

the four coils of the stepper motor we are using the digital pins 8,9,10 and 11. The driver module is powered by the 5V pin of the Arduino Board.

But, power the driver with External Power supply when you are connecting some load to the steppe

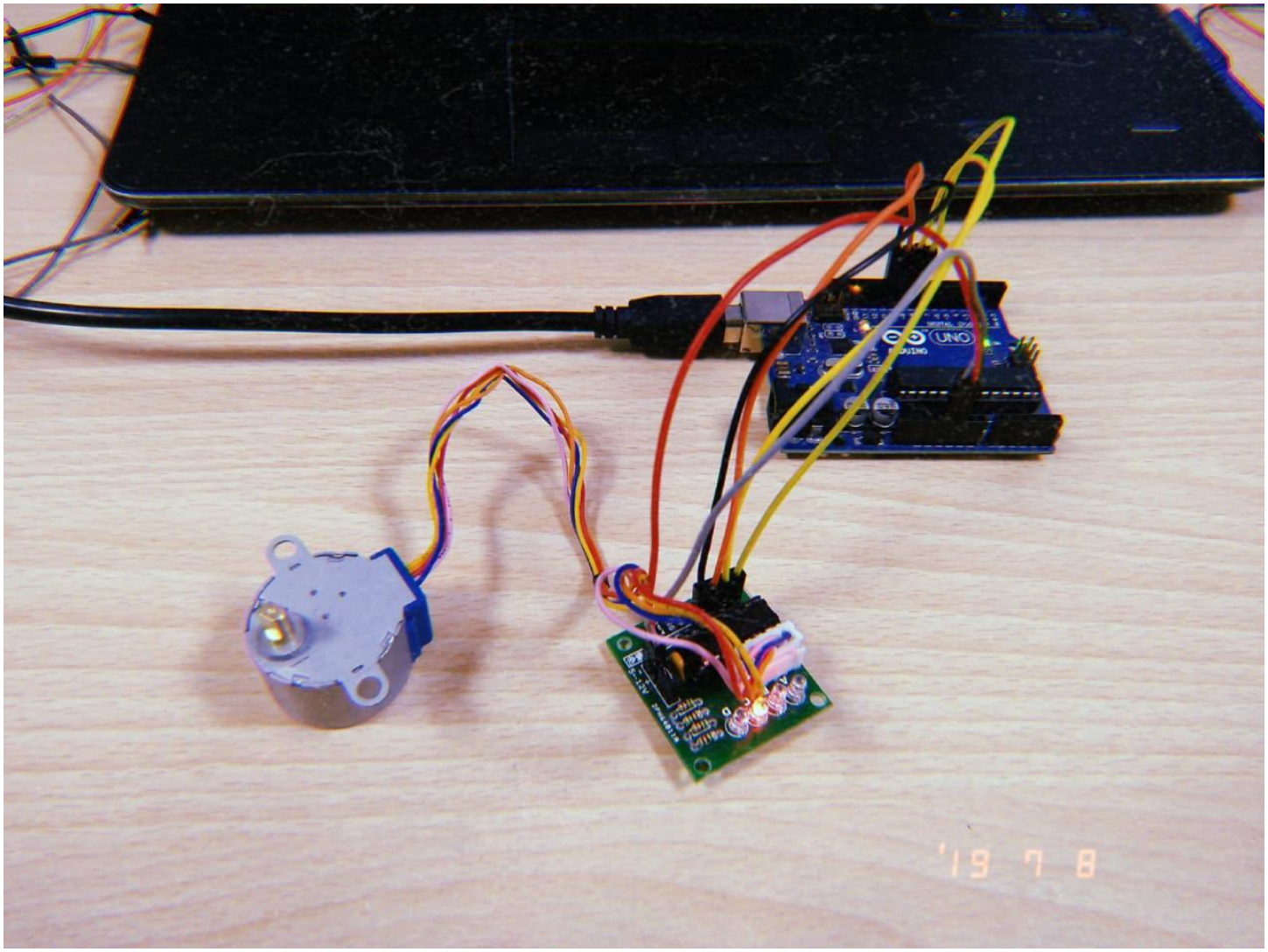
motor. Since I am just using the motor for demonstration purpose I have used the +5V rail of the Arduino Board. Also remember to connect the Ground of the Arduino with the ground of the Diver module.

WORKING

we will have to make 2048 steps to make one complete rotation, so when we enter 2048 the motor will make one complete rotation in clockwise direction by making 2048 steps. To rotate in anti-clockwise just enter the number with "–"negative sign. So, entering -1024 will make the motor to rotate half the way in anti-clock wise direction. You can enter any desired values, like entering 1will make the motor to take only one step.

CODE

```
int motorPin1 = 8;
int motorPin2 = 9;
int motorPin3 = 10;
int motorPin4 = 11;
int delayTime = 150;

void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
}

void loop() {
  digitalWrite(motorPin4, HIGH);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin1, LOW);
  delay(delayTime);
  digitalWrite(motorPin4, LOW);
  digitalWrite(motorPin3, HIGH);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin1, LOW);
  delay(delayTime);
  digitalWrite(motorPin4, LOW);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin2, HIGH);
  digitalWrite(motorPin1, LOW);
  delay(delayTime);
  digitalWrite(motorPin4, LOW);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin1, HIGH);
  delay(delayTime);
}
```


UNO
'19 7 8

PROJECT

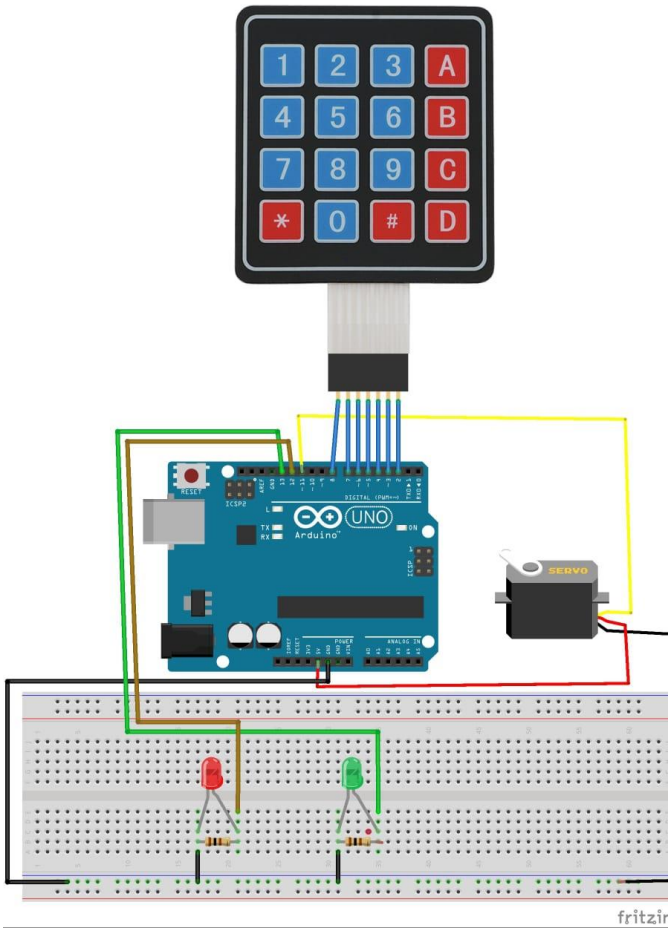
Arduino Door Lock Using 4x4 Keypad and Servo Motor

In this project i have tried to explain how to built a Door Lock Security System with an Arduino and Keypad.

I have used 4x4 Keypad for this project but you can use different models but do not forget to change some parts in the code.

Actually it is very easy to use this component with an Arduino , however only thing that you may not like is 4x4 keypad need 8 PIN to work properly.

Circuit and Working



PIN CONNECTIONS

row Pins is connected to { 8, 7, 6, 9 };
col Pins to { 5, 4, 3, 2 };
red led Pin to 12;
 green led Pin to 13;

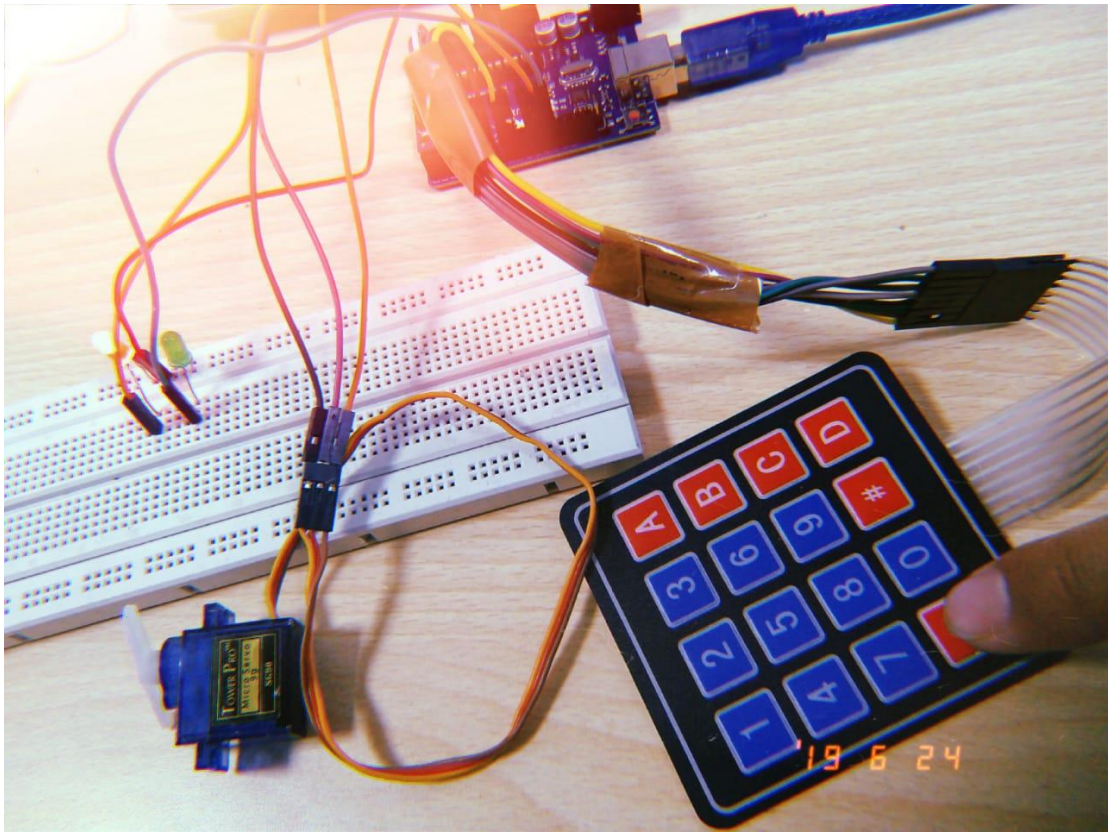
WORKING

As theft is increasing day by day, security is becoming a major concern. In this project, we will make a digital door lock system with keypad using an Arduino Uno. It will open your door only when the right password (***#C9**) is entered and it will start beeping for a wrong password.

CODE

```
#include <Keypad.h>
#include <Servo.h>

Servo servo_Motor;
char* password = "123";
int position = 0;
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};

byte rowPins[ROWS] = { 8, 7, 6, 9 };
byte colPins[COLS] = { 5, 4, 3, 2 };
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
int redPin = 12;
int greenPin = 13;

void setup()
{
pinMode(redPin, OUTPUT);
pinMode(greenPin, OUTPUT);
servo_Motor.attach(11);
setLocked(true);
}

void loop()
{
char key = keypad.getKey();
if (key == '*' || key == '#')
{
```

```
position = 0;
setLocked(true);
}
if (key == password[position])
{
position ++;
}
if (position == 3)
{
setLocked(false);
}
delay(100);
}
void setLocked(int locked)
{
if (locked)
{
digitalWrite(redPin, HIGH);
digitalWrite(greenPin, LOW);
servo_Motor.write(11);
}
else
{
digitalWrite(redPin, LOW);
digitalWrite(greenPin, HIGH);
servo_Motor.write(90);
}
}
```


A B C D
3 6 9 #
2 5 8 0
1 4 7
'19 6 24

PROJECT

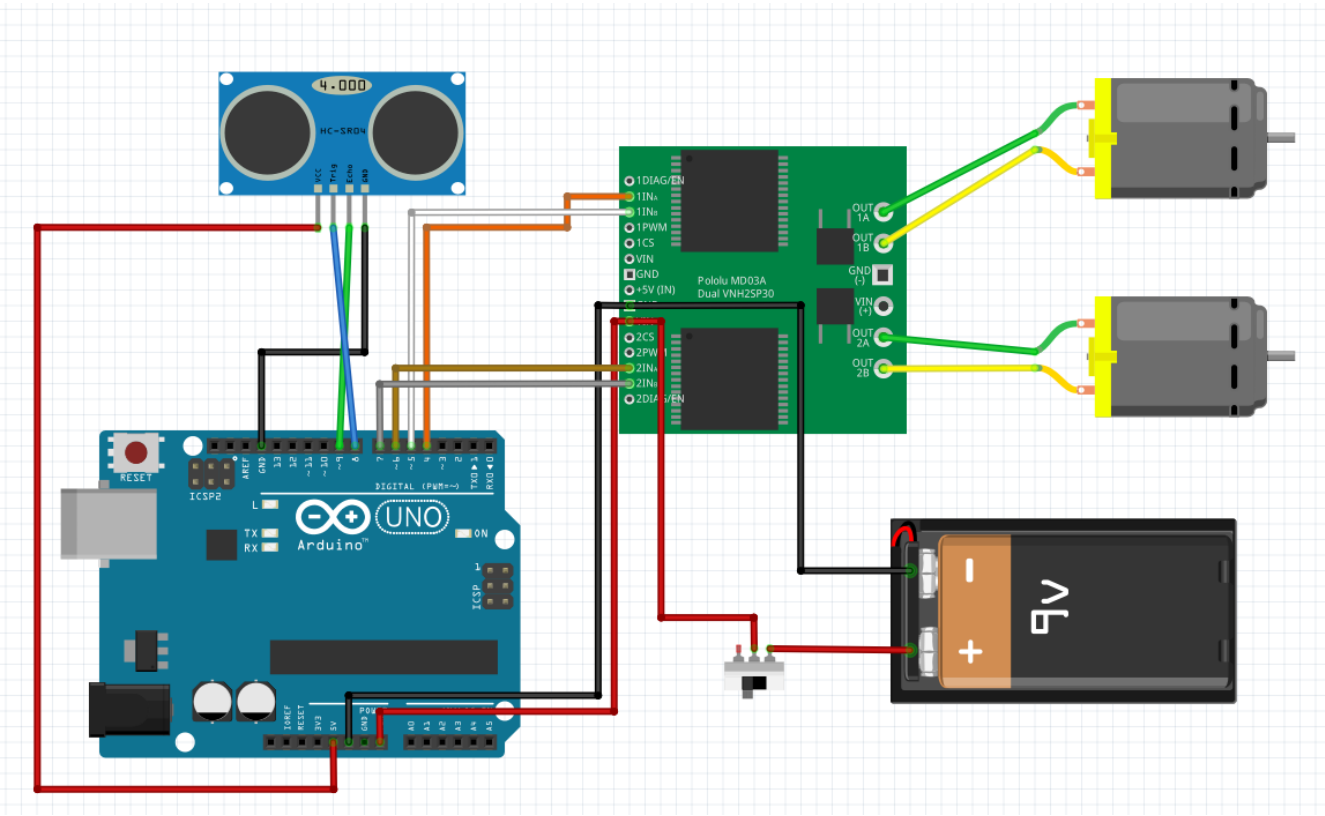
OBSTACLE AVOIDING CAR Using ARDUINO UNO

This car was made using Arduino Uno board and an Ultrasonic sensor as major components.

To drive the car a two sets of geared motor which was driven by motor driver connected to Arduino for input.

Whenever an obstacle comes in front of Ultrasonic sensor at a particular distance a specific command gets executed which drives specific motors.

Circuit and Working



PIN CONNECTIONS

Ultrasonic sensor has four pins namely GND, VCC, ECHO, TRIG GND and VCC are connected to GND and +5V of the Arduino respectively, ECHO and TRIG pins are connected to pin no.8 and pin no.9 respectively.

Motor driver has many pins but we use only 10 pins from it namely 1INa, 1INb, GND, Vin, 2INa, 2INb, OUT 1a, OUT 1b, OUT2a, OUT2b.

The out pins are connected to motors as shown in the circuit, GND and VCC are used for power supply for the driver, rest of the pins are used for input to the driver board from Arduino.

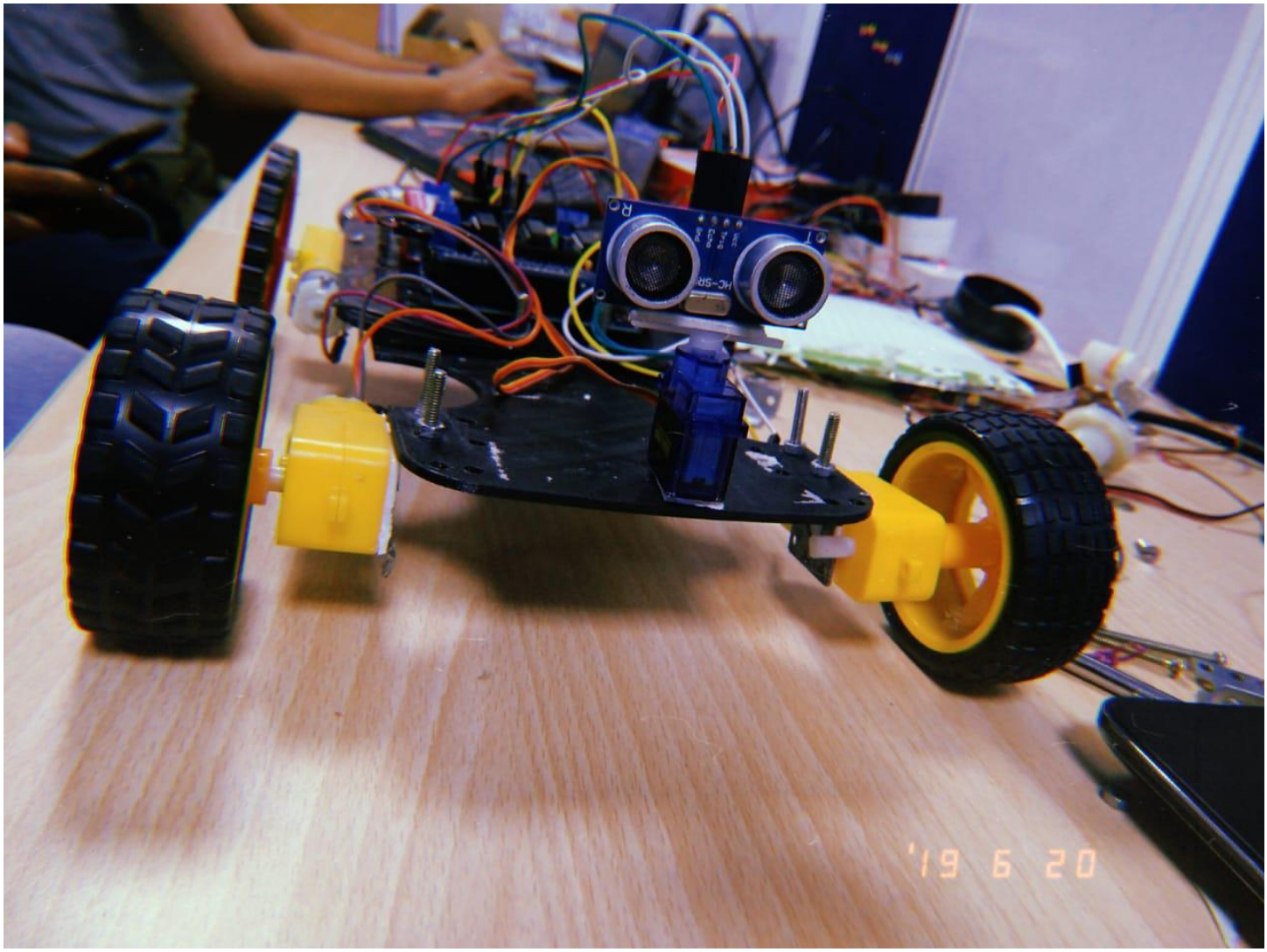
WORKING

Whenever an object comes under the range of ultrasonic sensor a specified condition gets executed in Arduino board as per the input from Ultrasonic sensor which gives input to motor driver and hence drives the motor in the particular direction.
If no object is in front of Ultrasonic sensor the car will move forward in straight direction.

CODE

```
inttP=8;
inteP=9;
intrevrht=6;
intfowrht=7;
intrevlft=4;
intfowlft=5;
floatspd=343;
floattD;
floatping;

voidsetup(){
  pinMode(tP,OUTPUT);
  pinMode(eP,INPUT);
  pinMode(revlft,OUTPUT);
  pinMode(revrht,OUTPUT);
  pinMode(fowlft,OUTPUT);
  pinMode(fowrht,OUTPUT);
  Serial.begin(9600);
}

voidloop(){
  digitalWrite(tP,HIGH);
  delayMicroseconds(2000);
  digitalWrite(tP,LOW);
  delayMicroseconds(15);
  digitalWrite(tP,HIGH);
  delayMicroseconds(10);

  ping=pulseIn(eP,HIGH);
  ping=ping/1000000;
  tD=spd*ping;
  tD=tD/2;
  tD=tD*100;
  Serial.println(tD);
```

```
  if(tD>18.){
  digitalWrite(fowlft,HIGH);
  digitalWrite(fowrht,HIGH);
  digitalWrite(revlft,LOW);
  digitalWrite(revrht,LOW);
   }

  if(tD<18.){
    digitalWrite(fowlft,LOW);
    digitalWrite(fowrht,LOW);
    digitalWrite(revlft,LOW);
    digitalWrite(revrht,LOW);
    delay(500);
    digitalWrite(fowlft,LOW);
    digitalWrite(fowrht,LOW);
    digitalWrite(revlft,HIGH);
    digitalWrite(revrht,HIGH);
    delay(500);
    digitalWrite(fowlft,LOW);
    digitalWrite(fowrht,LOW);
    digitalWrite(revlft,LOW);
    digitalWrite(revrht,LOW);
    delay(100);
    digitalWrite(fowlft,LOW);
    digitalWrite(fowrht,HIGH);
    digitalWrite(revlft,LOW);
    digitalWrite(revrht,LOW);
    delay(100);
    digitalWrite(fowlft,HIGH);
    digitalWrite(fowrht,LOW);
    digitalWrite(revlft,LOW);
    digitalWrite(revrht,LOW);
    delay(500);
   }
}
```

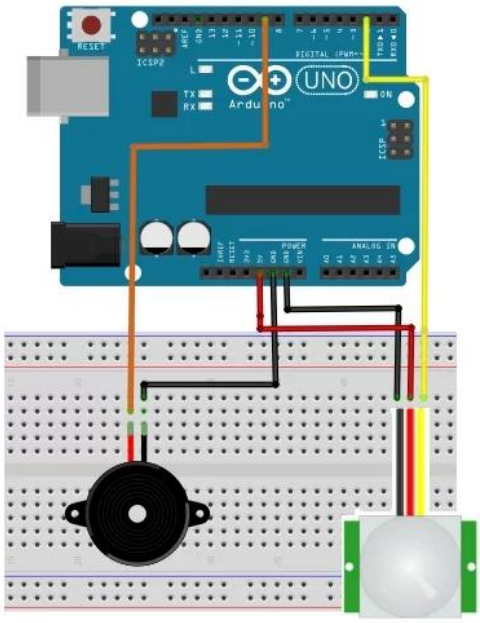

'19 6 20

PROJECT

PIR motion sensor alarm using Arduino

Here, we are using a PIR motion sensor. This motion sensor consists of a fresnel lens, an infrared detector, and supporting detection circuitry. The lens on the sensor focuses any infrared radiation present around it toward the infrared detector. Our bodies generate infrared heat, and this heat is detected by the motion sensor. The sensor outputs a 5V signal for a period of one minute as soon as it detects the presence of a person. It offers a tentative range of detection of about 6–7m and is highly sensitive.

Circuit and Working



PIN CONNECTIONS

The connections required to interface the Arduino motion sensor and the piezo buzzer are very simple. Connect the motion sensor to your Arduino as per the following connection diagram. Connect the VCC and GND on the sensor to the Arduino's 5V and GND pins. Next, connect the output signal pin on the motion sensor to the Arduino's digital pin 2 (interrupt pin 0).

After hooking up the Arduino motion sensor, we have to connect the piezo buzzer to this system. To do this, connect the negative terminal of the buzzer (black wire) to the Arduino's GND pin and the positive terminal of the buzzer (red wire) to the Arduino's digital pin 9.

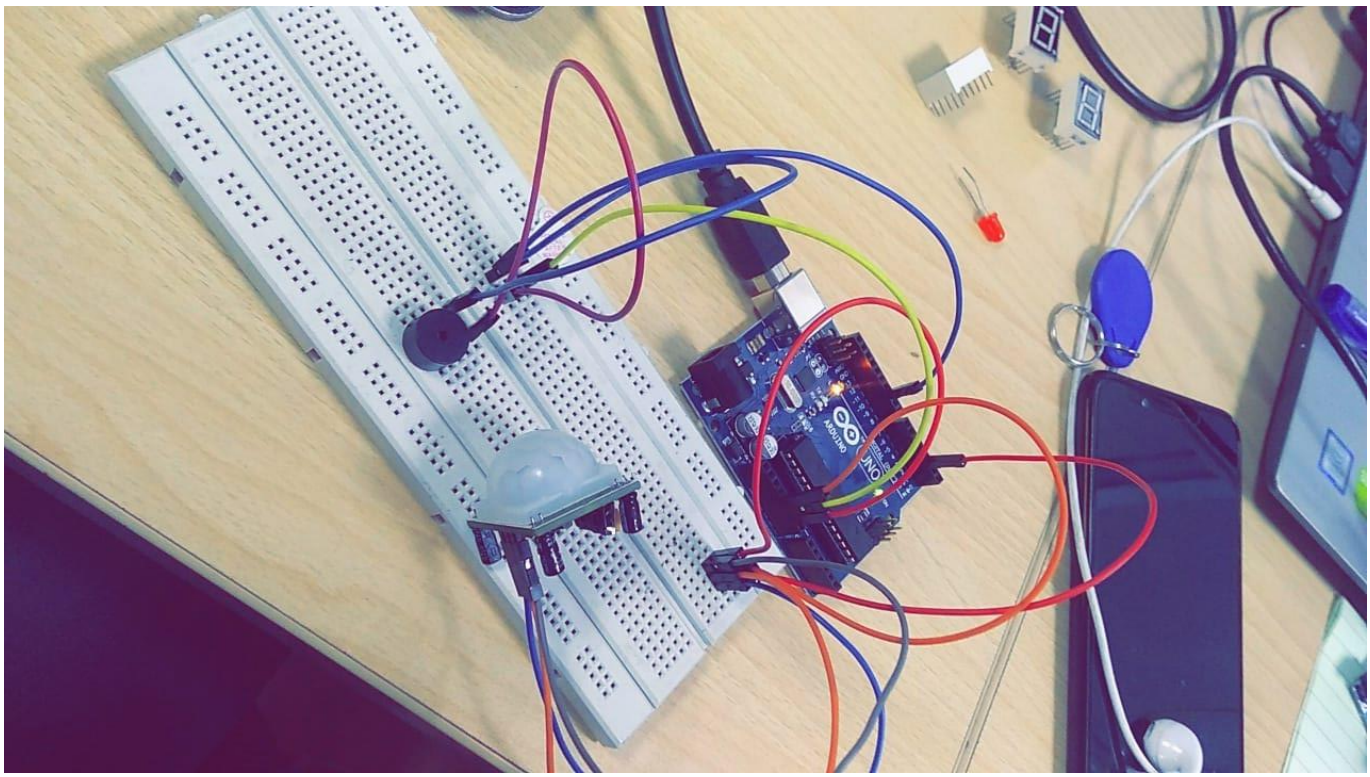
WORKING

When the PIR motion sensor detects a person, it outputs a 5V signal to the Arduino and triggers an interrupt. We define what the Arduino should do as it detects an intruder. Here, we are creating an alarm sound through a piezo buzzer. When the sensor detects an intruder, it triggers an alarm sound through the buzzer.
The piezo buzzer is activated through the Arduino using PWM (pulse width modulation) signals. The source code at the end of this tutorial will show you how to do this.

CODE

```
int speakerOut = 9;
byte names[] = {'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};
int tones[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
byte melody[] = "2d2a1f2c2d2a2d2c2f2d2a2c2d2a1f2c2d2a2a2g2p8p8p8p";
int count = 0;
int count2 = 0;
int count3 = 0;
int MAX_COUNT = 24;
int statePin = LOW;
void siren();

 volatile byte intruder;
 void setup()
 {
   Serial.begin(115200);
   attachInterrupt(0, intruder_detect, RISING);//Initialize the intterrupt pin for the motion sensor (Arduino digital pin 2)
   intruder = 0;
 }
 void loop()
 {

 }
 void intruder_detect()//This function is called whenever an intruder is detected by the arduino
 {
   intruder++;
   Serial.println("Intruder detected");
   for(int i=0; i<3; i++)//Play the alarm three times
   siren();
 }
 void siren()//This function will make the alarm sound using the piezo buzzer
{
for (count = 0; count < MAX_COUNT; count++) {
      for (count3 = 0; count3 <= (melody[count*2] - 48) * 30; count3++) {
      for (count2=0;count2<8;count2++) {
        if (names[count2] == melody[count*2 + 1]) {
          analogWrite(speakerOut,1023);
          delayMicroseconds(tones[count2]);
          analogWrite(speakerOut, 0);
          delayMicroseconds(tones[count2]);
        }
        if (melody[count*2 + 1] == 'p') {
          // make a pause of a certain size
```

```
        analogWrite(speakerOut, 0);
        delayMicroseconds(100);
      }
    }
  }
 }
}
```



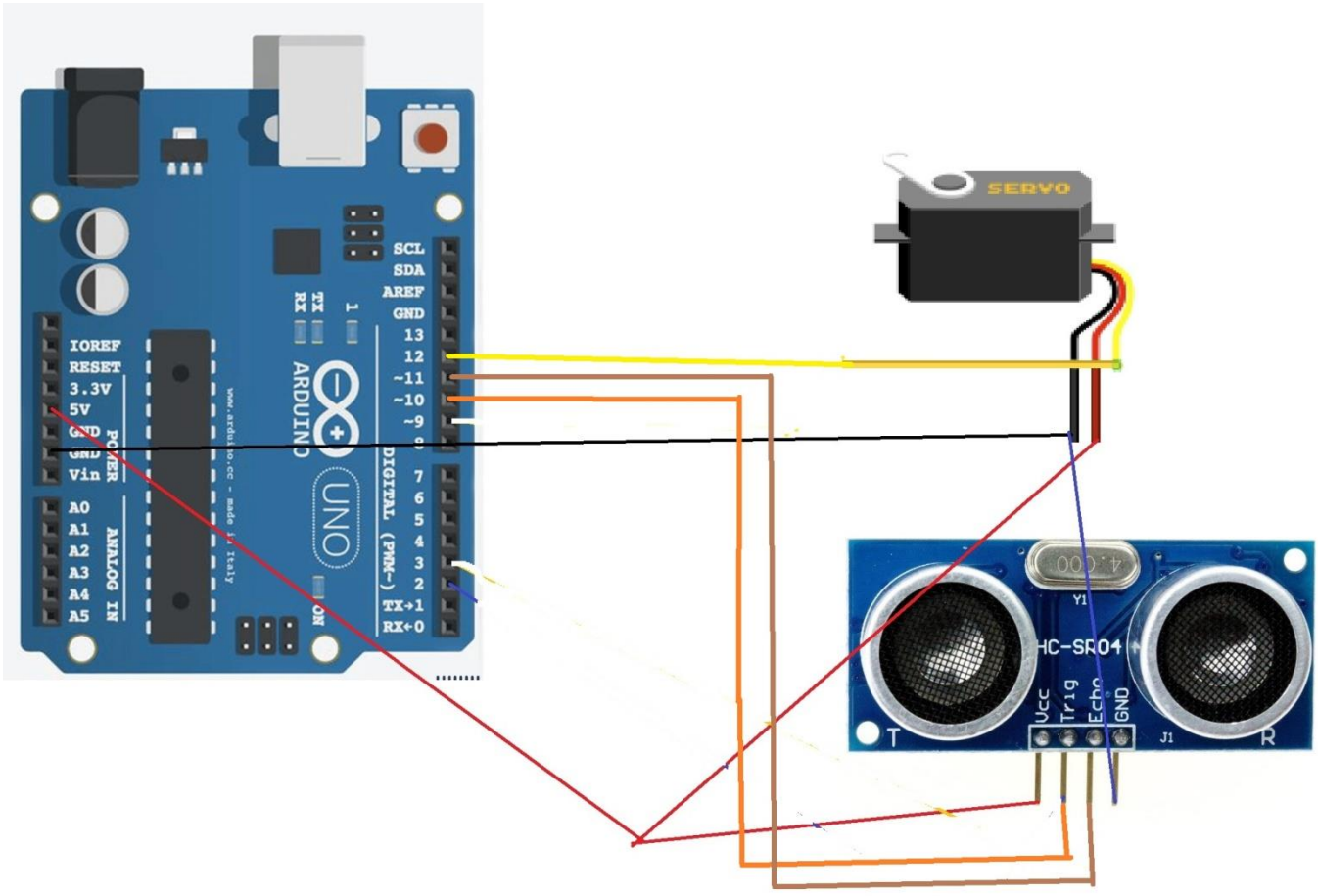
PROJECT

Radar (Radio Detection and Ranging) Using ARDUINO UNO

Radar is a device which transmit radio signals which are being travel and get returned from obstacles and received at the receiver.

It helps in finding the distance and location of the target or obstacle.

Circuit and Working



PIN CONNECTIONS

Connect the ultrasonic TRIG pin to arduino pin 10.

Connect the ultrasonic ECHO pin to arduino pin 11.

Take servo motor and connect signal pin to arduino pin 12.

Connect both ultrasonic sensor and servo Vcc pin to Arduino +5V pin.

Connect both ultrasonic sensor and servo GND pin to Arduino GND pin.

Upload the arduino code.

we use PROCESSING software to upload code to arduino

WORKING

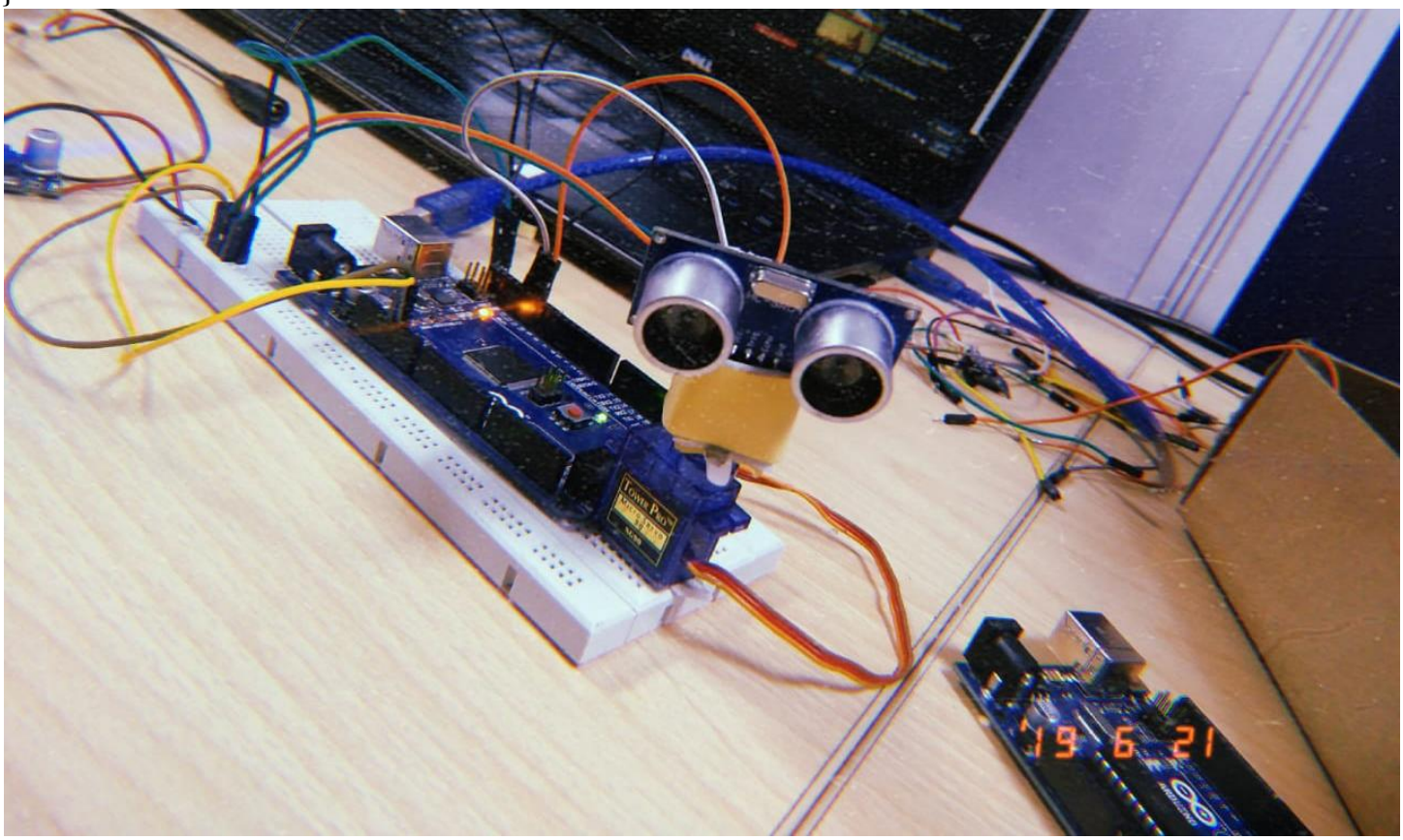
Radar is a long-range object detection system that uses radio waves to establish certain parameters of an object like its range, speed and position. Radar technology is used in aircrafts, missiles, marine, weather predictions and automobiles.

Even though the title says Arduino Radar Project, technically the project is based on Sonar technology as I will be using an Ultrasonic Sensor to determine the presence of any object in a particular range.

CODE

```
#include <Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo motor attached
}
void loop() {
  // rotates the servo motor from 15 to 165 degrees
  for(int i=15;i<=165;i++){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();// Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree

  Serial.print(i); // Sends the current degree into the Serial Port
  Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
  Serial.print(distance); // Sends the distance value into the Serial Port
  Serial.print("."); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
```

```
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
   }
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in microseconds
  distance= duration*0.034/2;
  return distance;
}
```

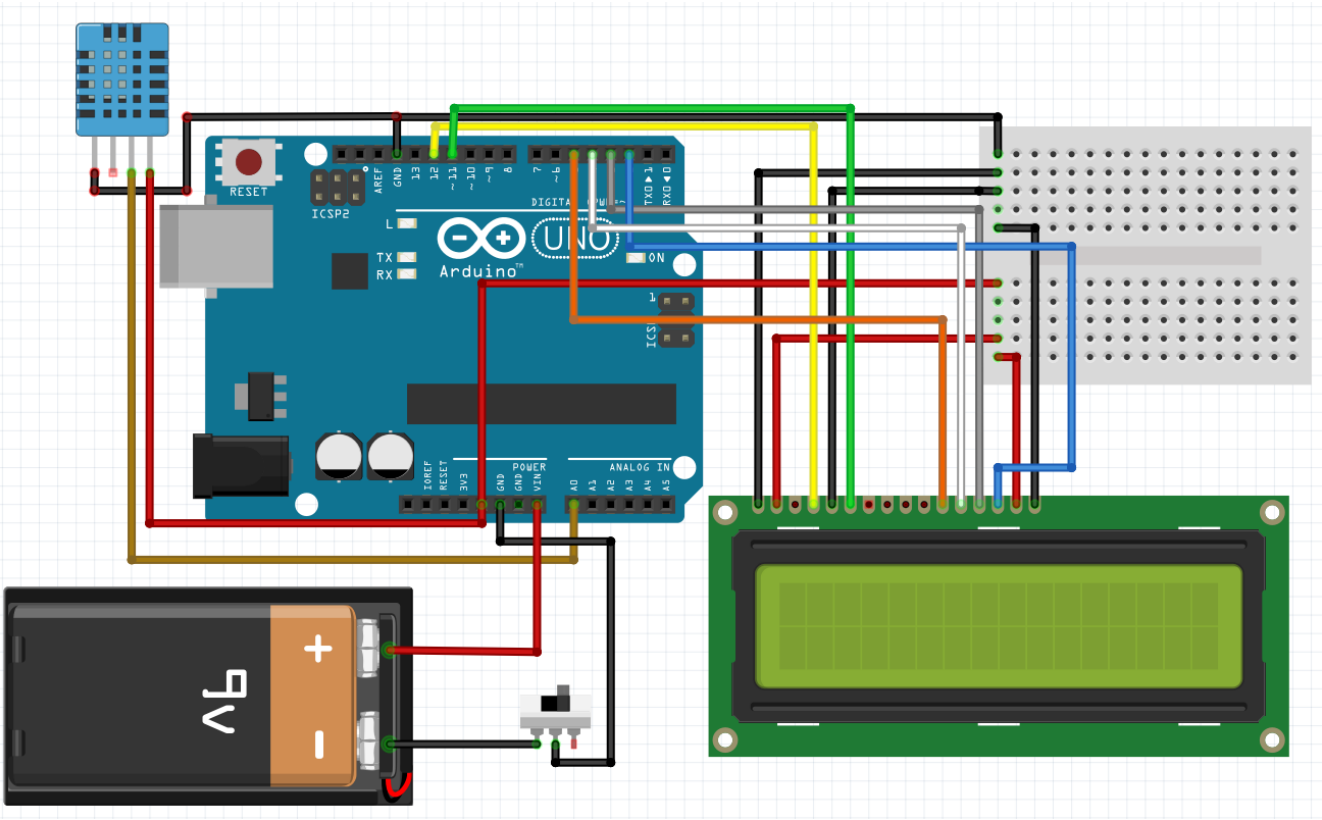


PROJECT

READING THE VALUES OF TEMPRATURE AND HUMIDITY USING DHT11 SENSOR ON ARDUINO.

In this project we have used a DHT11 sensor with Arduino Uno board. This sensor can read the values of temperature and humidity for 0-50 degree Celsius and 20-95% of humidity. It is very simple to connect as it has only three pins namely GND, Vcc and Data pin.
The printing of data from DHT11 sensor was done on 16x2 LCD display which was compatible with Arduino

Circuit and Working



PIN CONNECTIONS

For LCD, from left to right pins are defined as VSS, VDD, VO, RS, RW, E, DO, D1, D2, D3, D4, D5, D6, D7, Anode for backlight and (K)Cathode for backlight.
For DHT11 sensor left to right GND, Data and Vcc.
VSS, RW and K were connected to ground in LCD and RS, E, D4, D5, D6, D7 were used to transfer of data to print on LCD. Rest of the pins of LCD were remain unused,however some LCD requires contrast for which VO can be connected to ground through 5K ohm resistor.

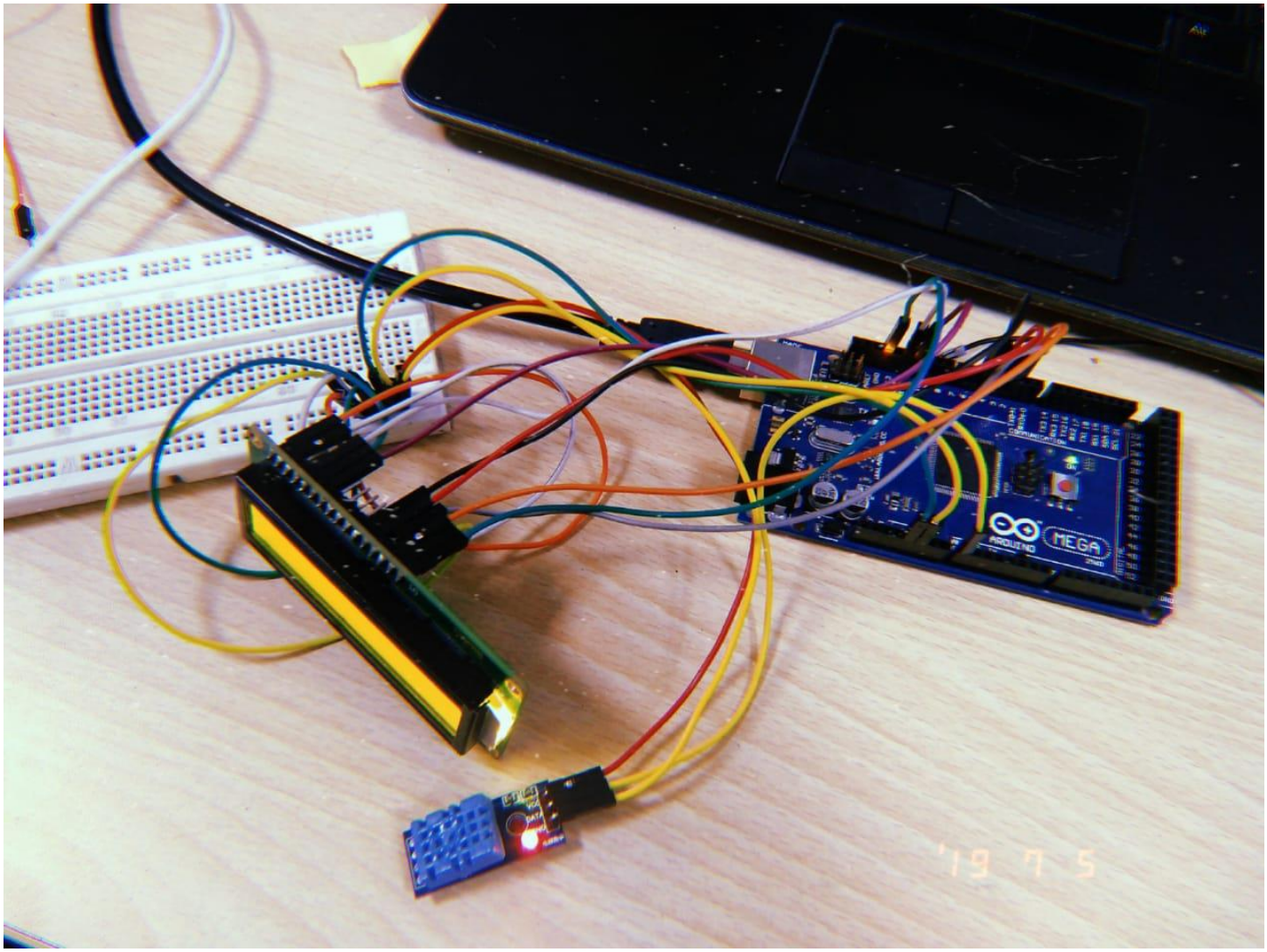
WORKING

Whenever some data is sensed by DHT11 sensor it is received by Arduino and sent to LCD where LCD displays the live

data of sensor. The delay time can be changed in the code to make LCD refresh faster and it may display the data more accurately .

CODE

```
#include<LiquidCrystal.h>
#include"dht.h"
intdhtPin=A0;
dhtDHT;
LiquidCrystallcd(12,11,5,4,3,2);

voidsetup(){

  Serial.begin(9600);
  lcd.begin(16,2);
  Serial.println("DHT11 Humidity & temperature Sensor\n\n");
  }

voidloop(){


    DHT.read11(dhtPin);

    Serial.print("Current humidity = ");
    Serial.print(DHT.humidity);
    Serial.print("%  ");
    Serial.print("temperature = ");
    Serial.print(DHT.temperature);
    Serial.println("C  ");

    lcd.setCursor(0,0);
    lcd.print("Humidity ");
    lcd.print(DHT.humidity);
    lcd.print(" %");
    lcd.setCursor(0,1);
    lcd.print("Temp ");
    lcd.print(DHT.temperature);
    lcd.print(" C");



    delay(500);


}
```

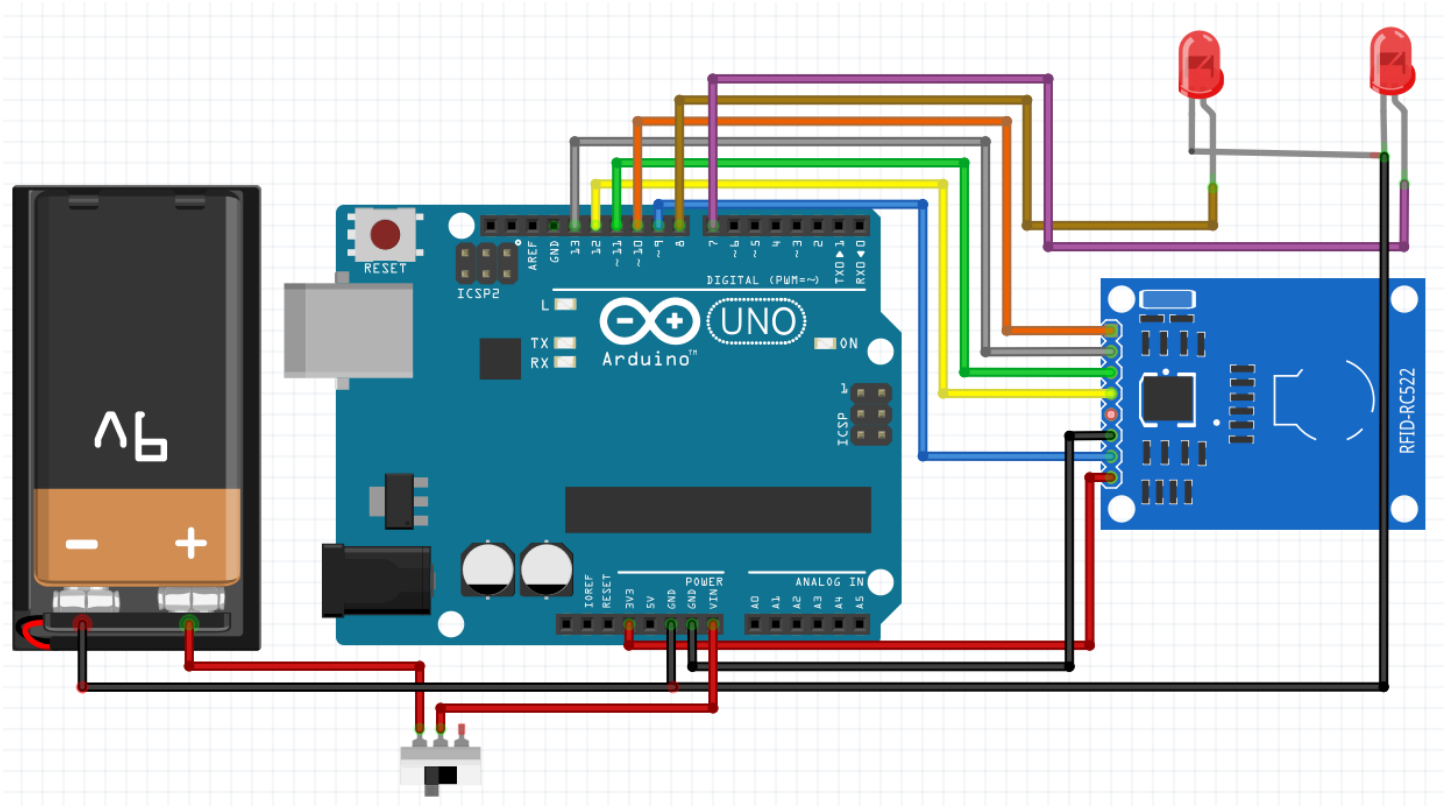

ARDUINO
MEGA
'19 7 5

PROJECT

RFID door lock Using ARDUINO UNO

RFID stands for Radio Frequency IDentification and it's a non-contact technology that's broadly used in many industries for tasks such as personnel tracking, access control, supply chain management, books tracking in libraries, tollgate systems and so on.

Circuit and Working



PIN CONNECTIONS

SDA is connected to pin 10 ,SCK is connected to pin 13, 3.3v connected to 3.3v arduino , GND to GND ,MOSI connected to 11 pin , MISO connected to pin 12 ,positive end of battery is connected to 1st pin of switch , 2nd pin of switch is connectes to VIN , negative leg of 1st led is connected to 8,7

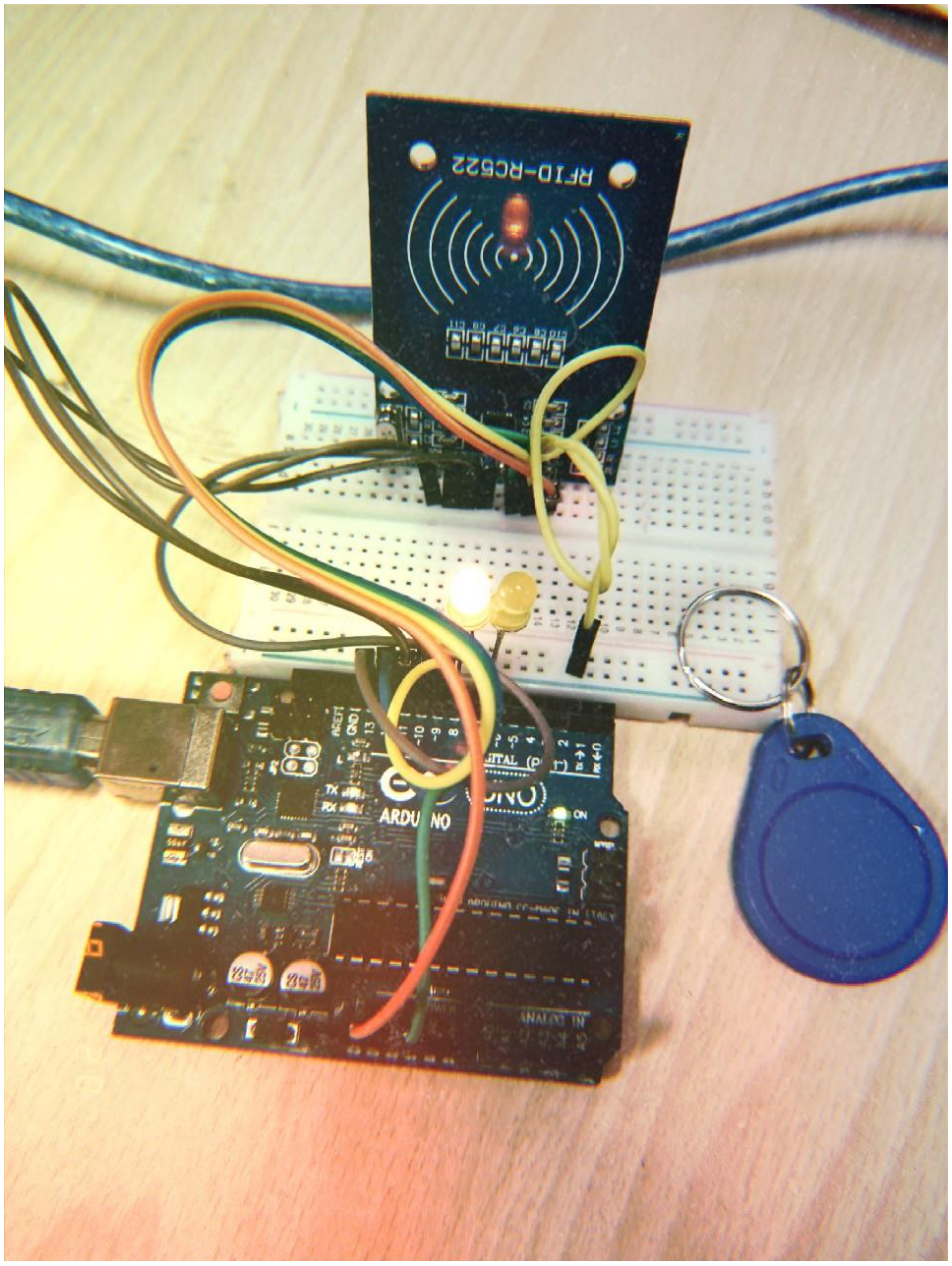
WORKING

An RFID system consists of two main components, a transponder or a tag which is located on the object that we want to be identified, and a transceiver or a reader. The RFID reader consist of a radio frequency module, a control unit and an antenna coil which generates high frequency electromagnetic field. On the other hand, the tag is usually a passive component, which consist of just an antenna and an electronic microchip, so when it gets near the electromagnetic field of the transceiver, due to induction, a voltage is generated in its antenna coil and this voltage

serves as power for the microchip.
Now as the tag is powered it can extract the transmitted message from the reader, and for sending message back to the reader, it uses a technique called load manipulation. Switching on and off a load at the antenna of the tag will affect the power consumption of the reader's antenna which can be measured as voltage drop. This changes in the voltage will be captured as ones and zeros and that's the way the data is transferred from the tag to the reader.

CODE

```
#include <SPI.h>
#include <MFRC522.h>
MFRC522 mfrc522(10, 9);
int granted = 8;
int denied = 7;

void setup(){
Serial.begin(9600);
SPI.begin();
mfrc522.PCD_Init();
pinMode(granted,OUTPUT);
pinMode(denied,OUTPUT);
Serial.println("Place the Card or Tag ");
}

void loop(){
if ( ! mfrc522.PICC_IsNewCardPresent()){
return;
}
if ( ! mfrc522.PICC_ReadCardSerial()) {
return;
}
Serial.print("Tag: ");
String content= "";
for (byte i = 0; i < mfrc522.uid.size; i++) {
Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
Serial.print(mfrc522.uid.uidByte[i], HEX);
content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
content.concat(String(mfrc522.uid.uidByte[i], HEX));
}
Serial.println();
content.toUpperCase();
content = content.substring(1);
if(content == "56 E0 67 AC"){
  digitalWrite(granted,HIGH);
  digitalWrite(denied,LOW);
Serial.println("Access Granted.");
Serial.println();
delay(900);
}else{
  digitalWrite(granted,LOW);
  digitalWrite(denied,HIGH);
Serial.println("Access Denied.");
delay(900);
}
}}
```

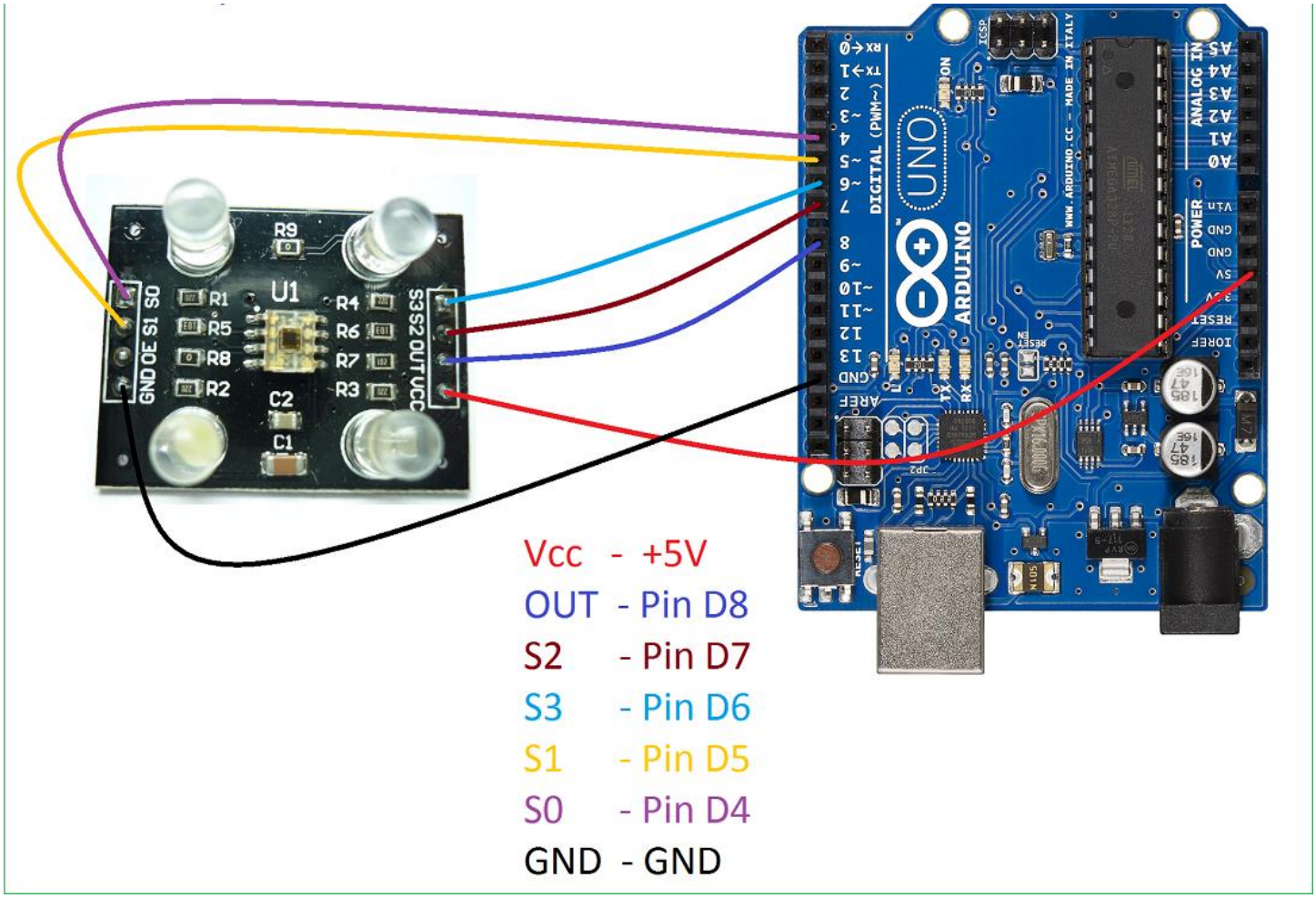

RFID-RC522
ARDUINO

PROJECT

RGB Color Detector Using TCS3200 Sensor Module

This project is used for detecting primary colors (red, green and blue, or RGB)—colors that are physically available in LEDs in one package; for example, common cathode or common-cathode RGB LED. We can display primary colors and also generate specific colors by modifying the Arduino code. The project demonstrates the basic interfacing of TCS3200 sensor, Arduino Uno and common-cathode RGB LED.

Circuit and Working



PIN CONNECTIONS

VCC to 5V
GND to GND
s0 to 4
s1 to 5
s2 to 6
s3 to 7
OUT to 8
OE to GND

WORKING

TCS3200 module has eight pins as shown in Fig. 4. This module consists of programmable color light-to-frequency converters that combine configurable silicon photodiodes and current-to-frequency converter on a single monolithic CMOS integrated circuit. Output is square-wave (50 per cent duty cycle) with frequency directly proportional to light intensity (irradiance).

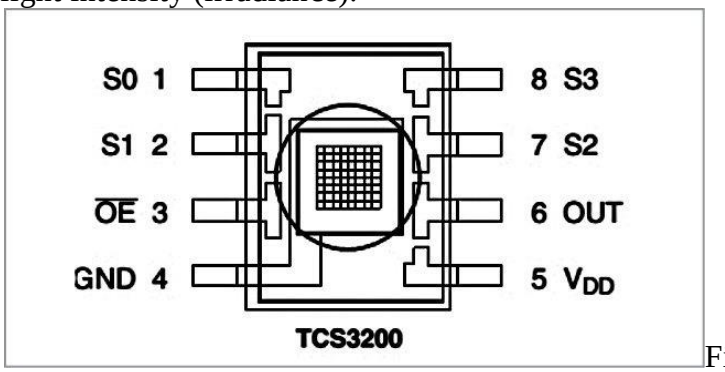


Fig. 4: Pin diagram of the TCS3200 color sensor module

Digital inputs and outputs allow direct interface to the MCU or other logic circuitry. Output enable (OE) places the output in high-impedance state for multiple units sharing an MCU input line. In TCS3200, the light-to-frequency converter reads an 8×8 array of photodiodes. Sixteen photodiodes have blue filters, another sixteen have green, yet another sixteen have red and remaining sixteen are clear with no filters.

TABLE I
TERMINAL FUNCTIONS

Pin name	Pin number	I/O	Description
GND	4		Power supply ground. All voltages referenced to GND
$\overline{OE}$	3	I	Enable for fo (active low)
OUT	6	O	Output frequency (fo)
S0, S1	1, 2	I	Output frequency scaling selection inputs
S2, S3	7, 8	I	Photodiode type selection inputs
V_{DD}	5		Supply voltage (5V)

All photodiodes of the same color are connected in parallel. Pins S2 and S3 of TCS3200 are used to select the group of photodiodes (red, green, blue and clear) that are active. The detailed pin description is shown in Tables I, II and III, respectively.

TABLE II
S0 AND S1 FUNCTIONS

S0	S1	Output Frequency (fo)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

Each sensor array in these three arrays is selected separately, depending on the requirement. Hence, it is known as a programmable sensor.

TABLE III
S2 AND S3 FUNCTIONS

S2	S3	Photodiode type
L	L	Red
L	H	Blue
H	L	Clear
H	H	Green

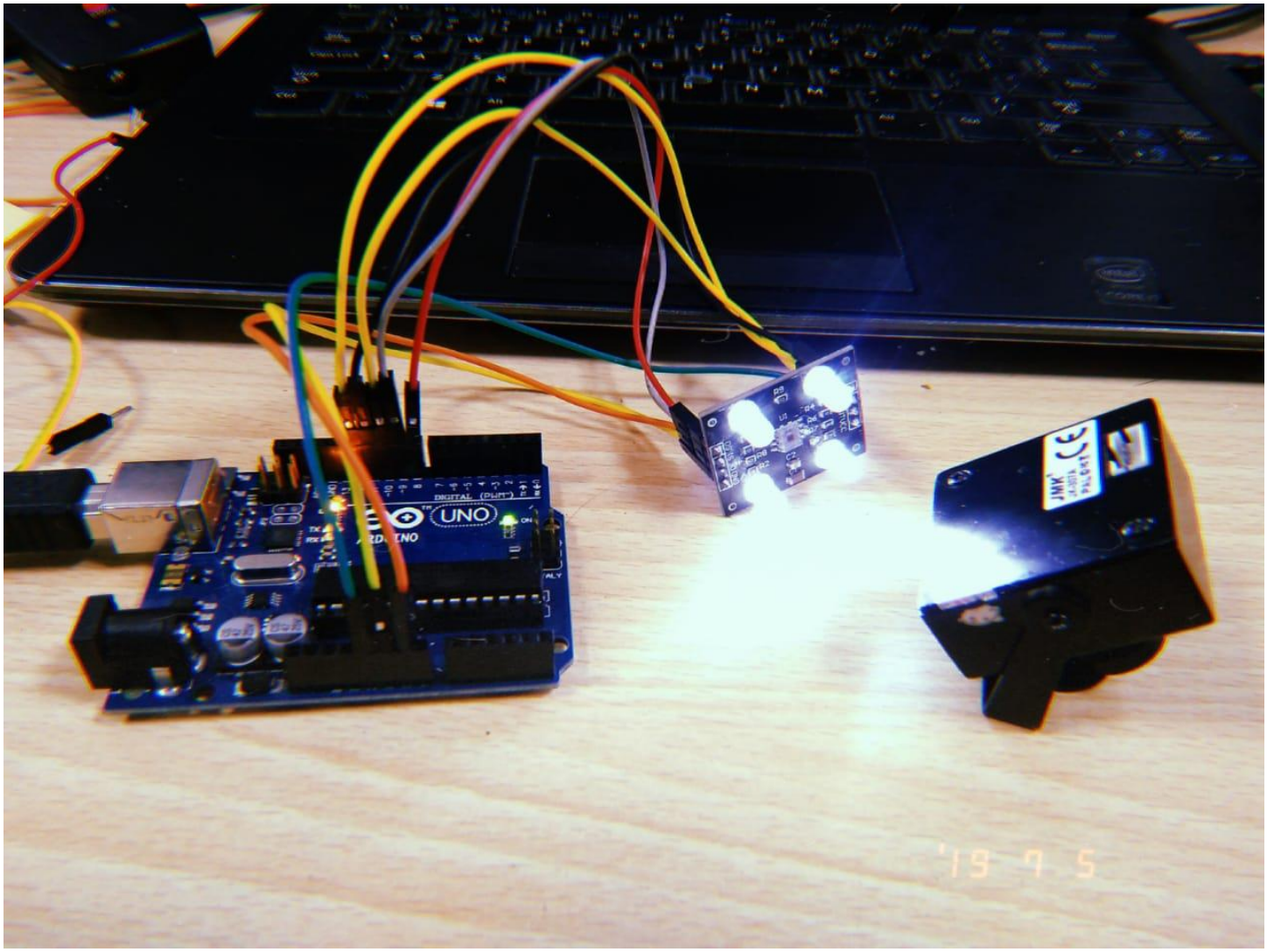
The module can be used to sense a particular color only. It contains filters for selection purpose. There is a fourth mode with no filter. With no filter, the sensor detects white light.

CODE

```
const int s0 = 4;
const int s1 = 5;
const int s2 = 6;
const int s3 = 7;
const int out = 8;
// LED pins connected to Arduino
```

```
int redLed = 2;
int greenLed = 3;
int blueLed = 4;
// Variables
int red = 0;
int green = 0;
int blue = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(out, INPUT);
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  digitalWrite(s0, HIGH);
  digitalWrite(s1, HIGH);
}

void loop()
{
  color();
  Serial.print("R Intensity:");
  Serial.print(red, DEC);
  Serial.print(" G Intensity: ");
  Serial.print(green, DEC);
  Serial.print(" B Intensity : ");
  Serial.print(blue, DEC);
  //Serial.println();

  if (red < blue && red < green && red < 20)
  {
   Serial.println(" - (Red Color)");
   digitalWrite(redLed, HIGH); // Turn RED LED ON
   digitalWrite(greenLed, LOW);
   digitalWrite(blueLed, LOW);
  }

  else if (blue < red && blue < green)
  {
   Serial.println(" - (Blue Color)");
   digitalWrite(redLed, LOW);
   digitalWrite(greenLed, LOW);
   digitalWrite(blueLed, HIGH); // Turn BLUE LED ON
  }
```

```
  else if (green < red && green < blue)
  {
   Serial.println(" - (Green Color)");
   digitalWrite(redLed, LOW);
   digitalWrite(greenLed, HIGH); // Turn GREEN LED ON
   digitalWrite(blueLed, LOW);
  }
  else{
  Serial.println();
  }
  delay(300);
  digitalWrite(redLed, LOW);
  digitalWrite(greenLed, LOW);
  digitalWrite(blueLed, LOW);
 }

void color()
{
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  //count OUT, pRed, RED
  red = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s3, HIGH);
  //count OUT, pBLUE, BLUE
  blue = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s2, HIGH);
  //count OUT, pGreen, GREEN
  green = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
}
```

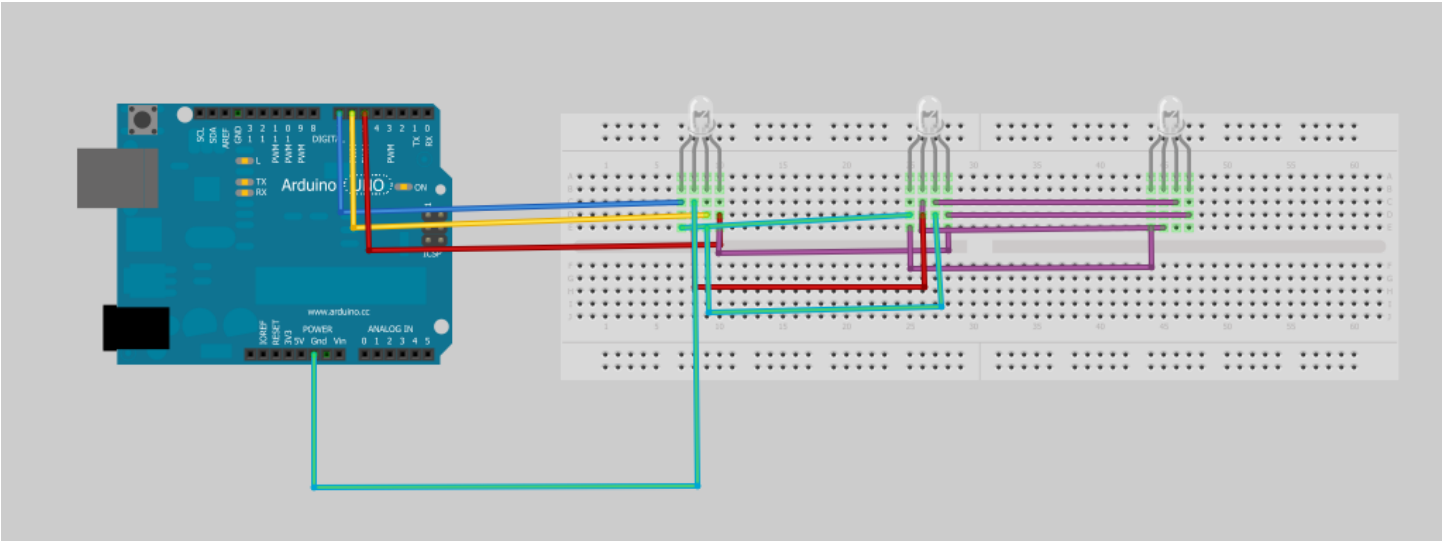

UNO
'19 9 5

PROJECT

Working of an RGB led Using ARDUINO UNO

The RGB LED can emit different colors by mixing the 3 basic colors red, green and blue. So it actually consists of 3 separate LEDs red, green and blue packed in a single case. That's why it has 4 leads, one lead for each of the 3 colors and one Common Cathode or Anode depending of the RGB LED type. The RGB LED used in this lab report is a Common Cathode RGB LED.

Circuit and Working



PIN CONNECTIONS

connect red Pin to 7,
connect green Pin = 6,
connect blue Pin = 5,
negative to ground

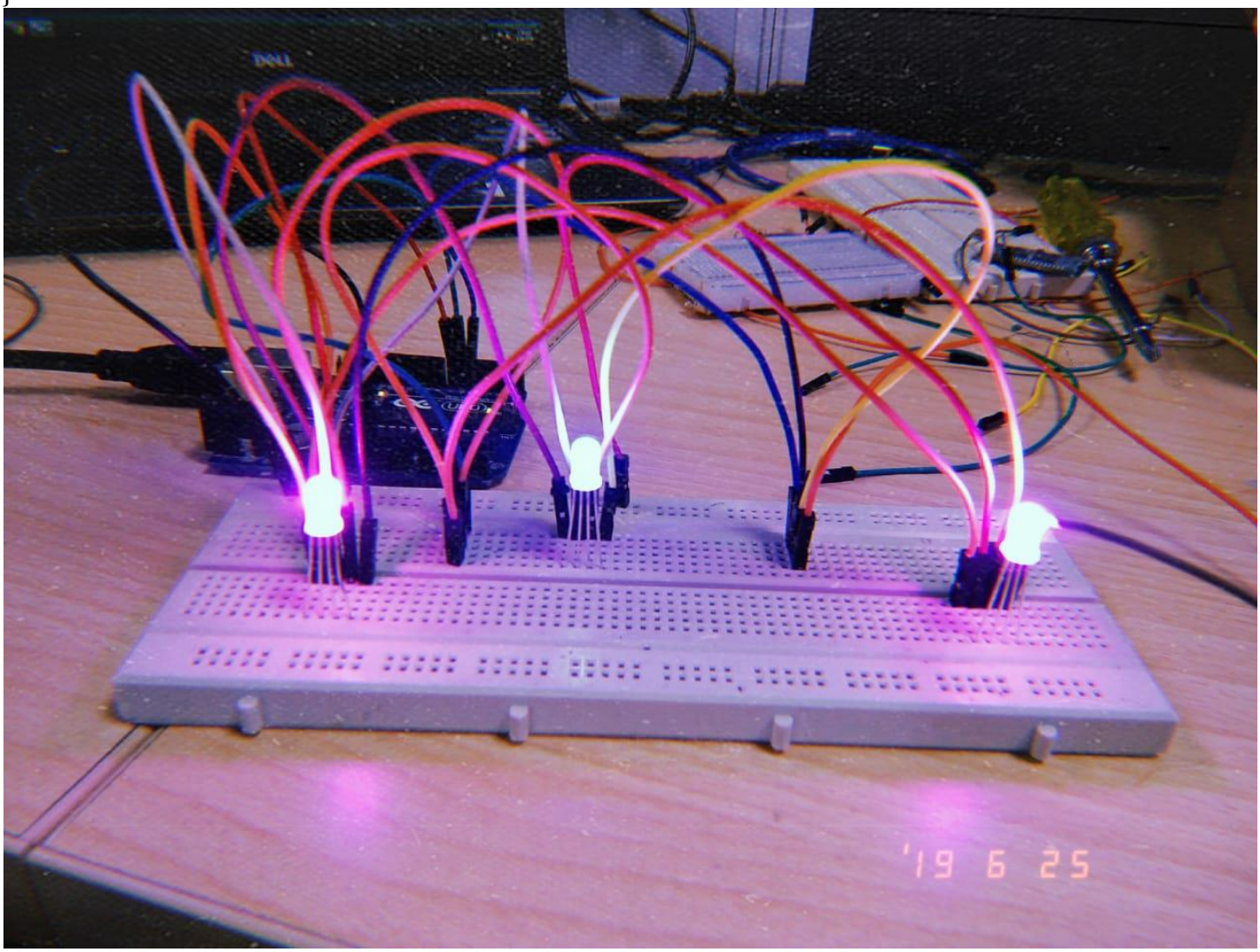
WORKING

The cathode will be connected to the ground and the 3 anodes will be connected through to 3 digital pins on the Arduino Board that can provide PWM (Pulse Width Modulator) signal. We will use PWM, Pulse Width Modulation for simulating analog output which will provide different voltage levels to the LEDs so we can get the desired colors.

CODE

```
int redPin= 7;
int greenPin = 6;
int bluePin = 5;
void setup()
{
 pinMode(redPin, OUTPUT);
 pinMode(greenPin, OUTPUT);
 pinMode(bluePin, OUTPUT);
```

```
}
void loop()
{
 setColor(255, 0, 0); // Red Color
 delay(1000);

setColor(0, 255, 0); // Green Color
 delay(1000);

setColor(0, 0, 255); // Blue Color
delay(1000);
setColor(255, 255, 255); // White Color
delay(1000);
setColor(170, 0, 255); // Purple Color
delay(1000);
}
```

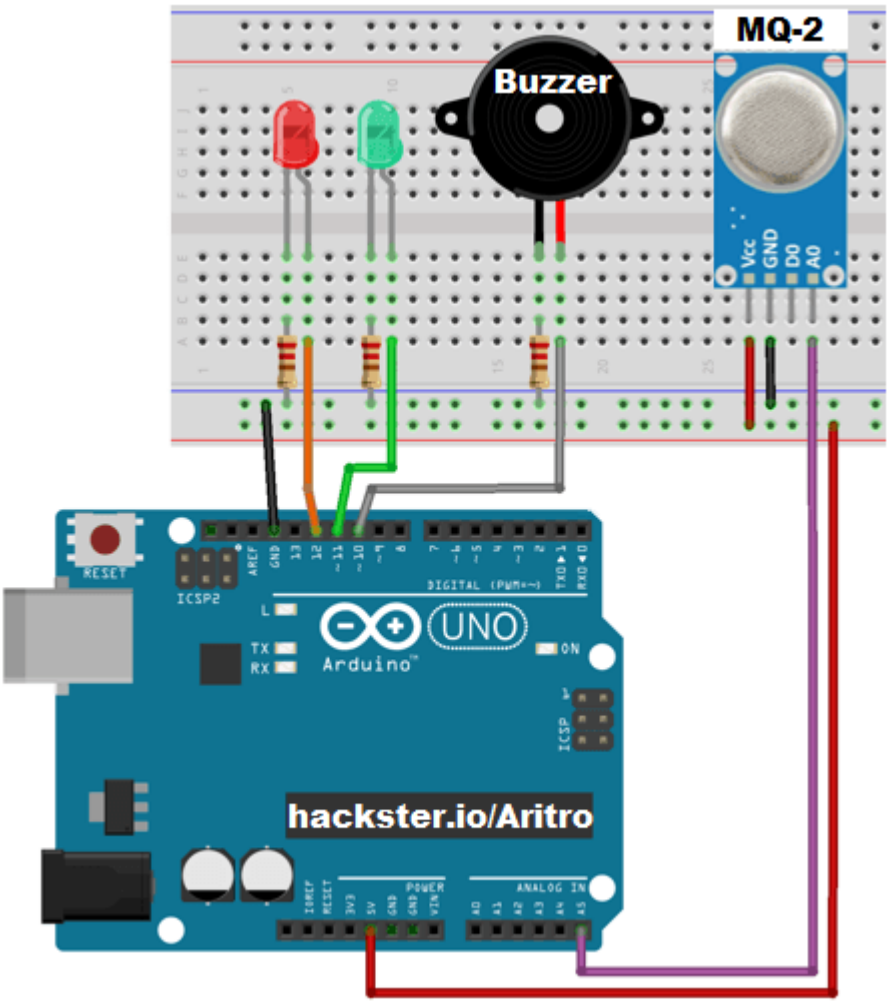


PROJECT

Smoke Sensor with buzzer Using ARDUINO UNO

In this project The gas sensor module mq-135 consists of a steel exoskeleton under which a sensing element is housed. This sensing element is subjected to current through connecting leads. This current is known as heating current through it, the gases coming close to the sensing element get ionized and are absorbed by the sensing element. This changes the resistance of the sensing element which alters the value of the current going out of it. .

Circuit and Working



PIN CONNECTIONS

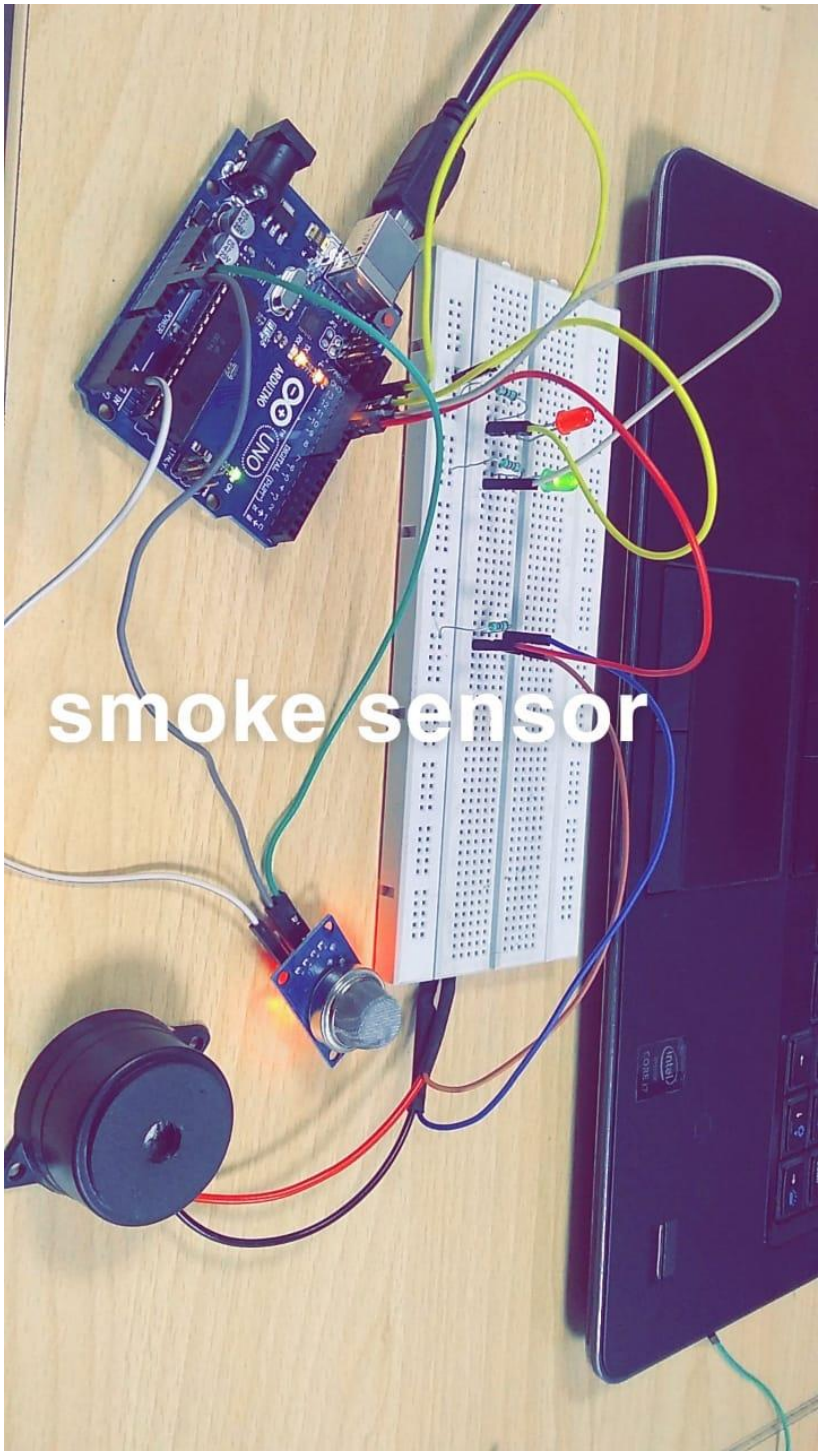
connect red Led to pin 12 arduino;
connect green Led to 11;
connect buzzer to 10;
connect smoke A0 to A5;
and ground to GND and 5v to Vcc

WORKING

The **MQ-135 smoke sensor** consists of a tin dioxide (SnO2), a perspective layer inside aluminium oxide micro tubes (measuring electrodes) and a heating element inside a tubular casing. The end face of the sensor is enclosed by a stainless steel net and the back side holds the connection terminals. Smoke is emitted from the source by burning anything. With the smoke cascade on the tin dioxide sensing layer, the resistance decreases. By using the external load resistance the resistance variation is converted into a suitable voltage variation.

CODE

```
int redLed = 12;
int greenLed = 11;
int buzzer = 10;
int smokeA0 = A5;
// Your threshold value
int sensorThres = 400;

void setup() {
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(smokeA0, INPUT);
  Serial.begin(9600);
}

void loop() {
  int analogSensor = analogRead(smokeA0);

  Serial.print("Pin A0: ");
  Serial.println(analogSensor);
  // Checks if it has reached the threshold value
  if (analogSensor > sensorThres)
  {
    digitalWrite(redLed, HIGH);
    digitalWrite(greenLed, LOW);
    tone(buzzer, 1000, 200);
  }
  else
  {
    digitalWrite(redLed, LOW);
    digitalWrite(greenLed, HIGH);
    noTone(buzzer);
  }
  delay(100);
}
```


smoke sensor

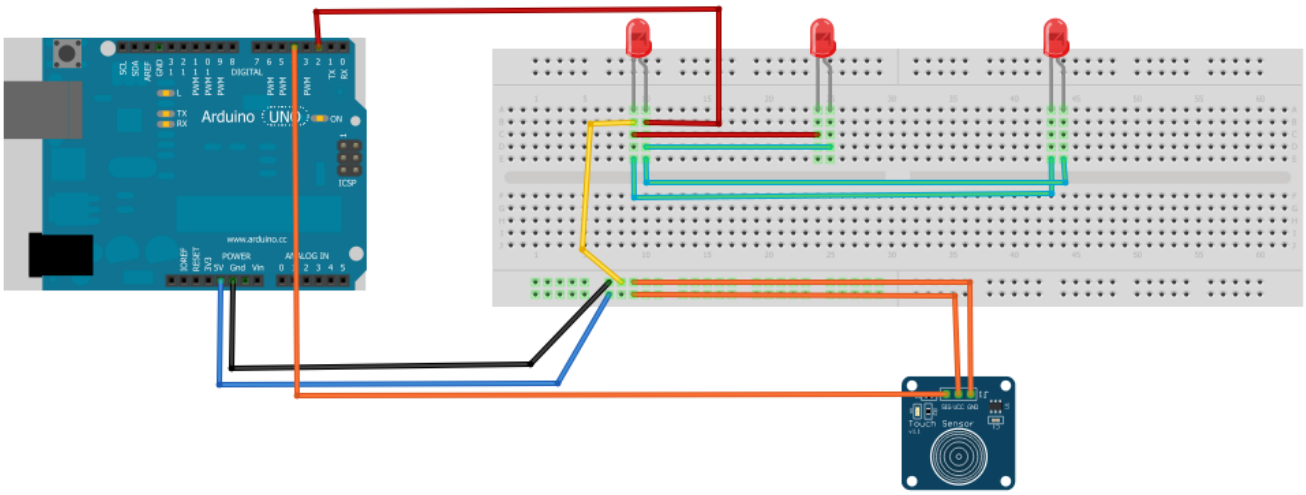
PROJECT

Touch Controlled Light Using ARDUINO UNO

In this project I made a touch sensor using a coin that works based on arduino Capacitive Sensing Library. I used this sensor to turn on and turn off LED by touching on it. Capacitive sensors can detect anything that is conductive or that has a significantly different permitivity than air, like a human body or hand.

The capacitive Sensor library turns two or more Arduino pins into a capacitive sensor, which can sense the electrical capacitance of the human body.

Circuit and Working



PIN CONNECTIONS

Negative pin of led connects with digital pin 2 and positive pin connects with the ground and same for other two. 5volt pin from arduino connects with the bread board and ground with bread board as shown and VCC pin of touch sensor connects with the 5v on bread board and GND pin to ground on bread board SIG pin is connect with digital pin 2 on arduino.

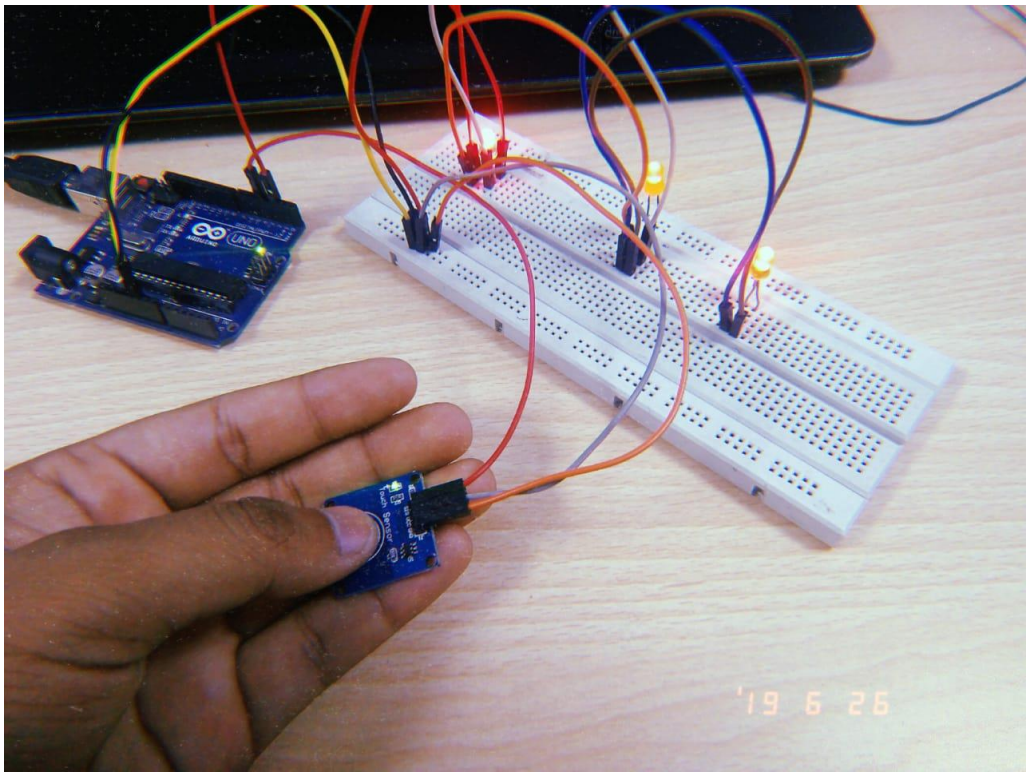
WORKING

Whenever you touches the sensor it changes the previous state, that is turning on or off the LED.

CODE

```
#define touchpin 4
int ledPin = 2;
void setup() {
  pinMode(touchpin, INPUT);
  pinMode(ledPin, OUTPUT);
```

```
}
void loop() {
  int touchValue = digitalRead(touchpin);
  if (touchValue == HIGH){
    digitalWrite(ledPin, HIGH);
  }
  else{
    digitalWrite(ledPin,LOW);
  }
  delay(300);
}
```

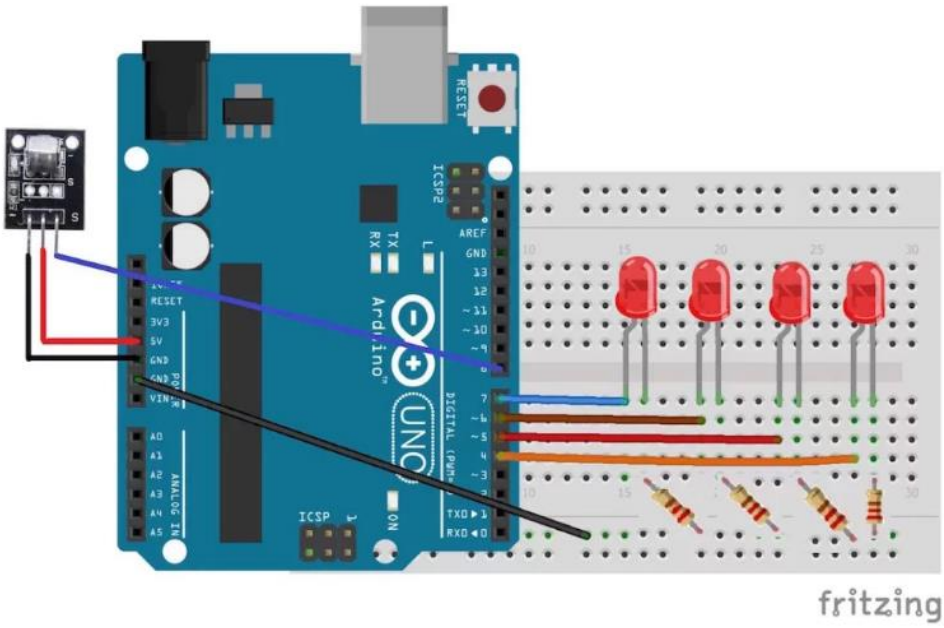


PROJECT

Control LEDs using an IR sensor and a remote

In this project, we are going to control LEDs using an IR sensor and a remote. The IR sensor is a 1838B IR receiver. Whenever a button on the remote is pressed, it will send an infrared signal to the IR sensor in the coded form. The IR sensor will then receive this signal and will give it to the Arduino.

Circuit and Working



PIN CONNECTIONS

First, connect the four LEDs to the Arduino. Connect the positives of the four LEDs to the pins 7, 6, 5, and 4. Connect the negative of the four LEDs to GND on the Arduino through the 220 ohm resistors. The longer wires on the LEDs are positive and the shorter wires are negative.

Then connect the IR sensor to the Arduino. The connections for the IR sensor with the Arduino are as follows:

Connect the negative wire on the IR sensor to GND on the Arduino.
Connect the middle of the IR sensor which is the VCC to 5V on the Arduino.
Connect the signal pin on the IR sensor to pin 8 on the Arduino.

WORKING

Whenever a button is pressed on the remote, it sends an infrared signal in encoded form. This signal is then

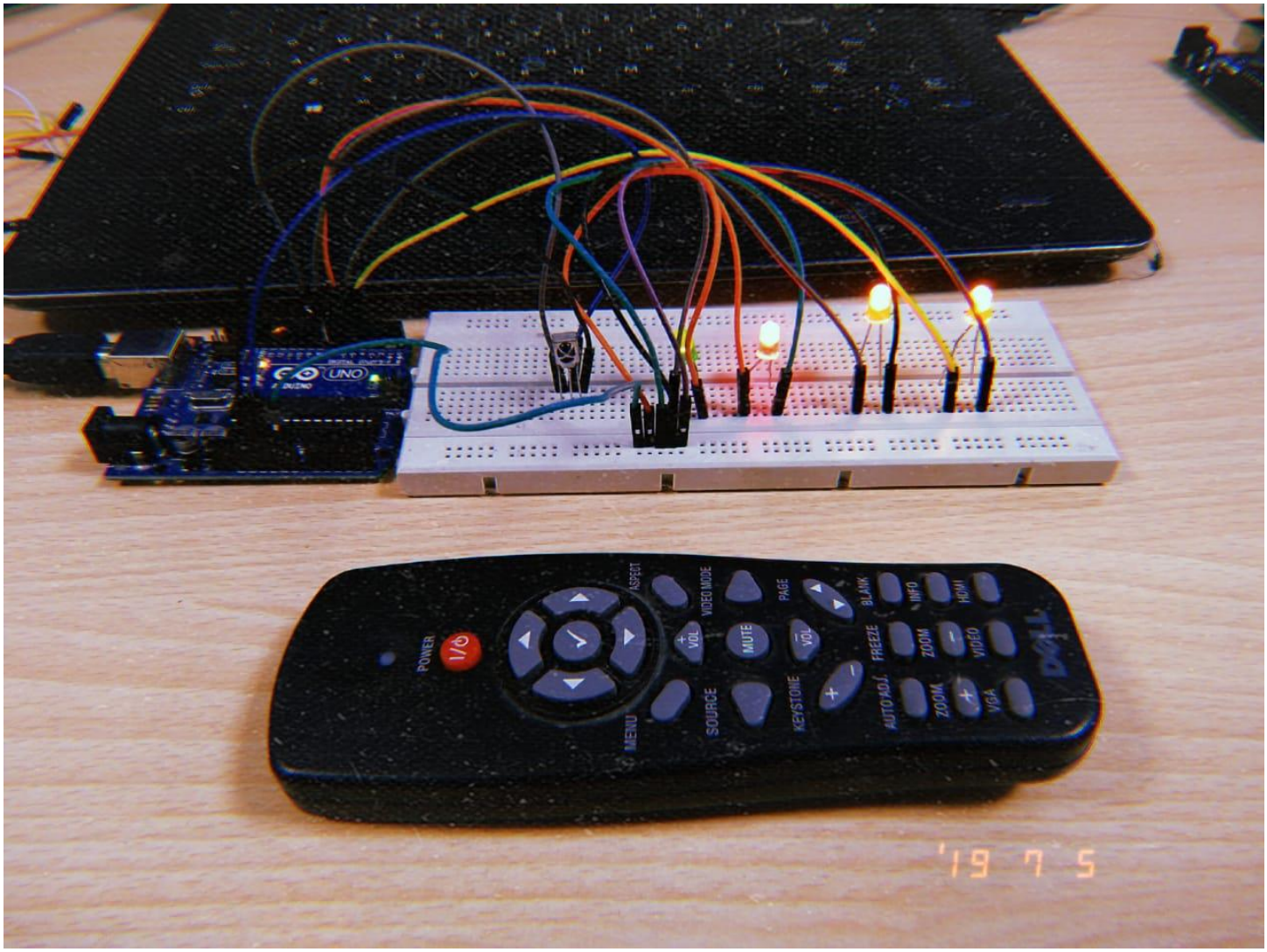
received by the IR receiver and given to the Arduino.

We will save the code for the buttons that we want to control the LEDs in the Arduino code. Whenever a button on the remote is pressed, the Arduino receives a code. The Arduino will compare this code with the codes already saved, and if any of them match, the Arduino will turn on the LED connected to that button.

CODE

```
#include <IRremote.h>
#define first_key  26775
#define second_key  51255
#define third_key  43095
#define fourth_key  10455
int receiver_pin = 8;

int first_led_pin = 7;
int second_led_pin = 6;
int third_led_pin = 5;
int fourth_led_pin = 4;

int led[] = {0,0,0,0};
IRrecv receiver(receiver_pin);
decode_results output;

void setup()
{
  Serial.begin(9600);
  receiver.enableIRIn();
  pinMode(first_led_pin, OUTPUT);
  pinMode(second_led_pin, OUTPUT);
  pinMode(third_led_pin, OUTPUT);
  pinMode(fourth_led_pin, OUTPUT);
  }

void loop() {
  if (receiver.decode(&output)) {
    unsigned int value = output.value;
    switch(value) {
       case first_key:
         if(led[1] == 1) {
            digitalWrite(first_led_pin, LOW);
            led[1] = 0;
         } else {
             digitalWrite(first_led_pin, HIGH);
             led[1] = 1;
         }
          break;
       case second_key:

         if(led[2] == 1) {
```

```
        digitalWrite(second_led_pin, LOW);
        led[2] = 0;
      } else {
        digitalWrite(second_led_pin, HIGH);
        led[2] = 1;
      }
      break;
    case third_key:

      if(led[3] == 1) {
        digitalWrite(third_led_pin, LOW);
        led[3] = 0;
      } else {
        digitalWrite(third_led_pin, HIGH);
        led[3] = 1;
      }
      break;
      case fourth_key:

      if(led[4] == 1) {
        digitalWrite(fourth_led_pin, LOW);
        led[4] = 0;
      } else {
        digitalWrite(fourth_led_pin, HIGH);
        led[4] = 1;
      }
      break;
  }
  Serial.println(value);
  receiver.resume();
 }
}
```


UNO
POWER
ASPECT
VIDEO MODE
PAGE
BLANK
INFO
HDMI
VOL
MUTE
VOL
FREEZE
ZOOM
VIDEO
MENU
SOURCE
KEYSTONE
AUTO ADJ.
ZOOM
VGA
DELL

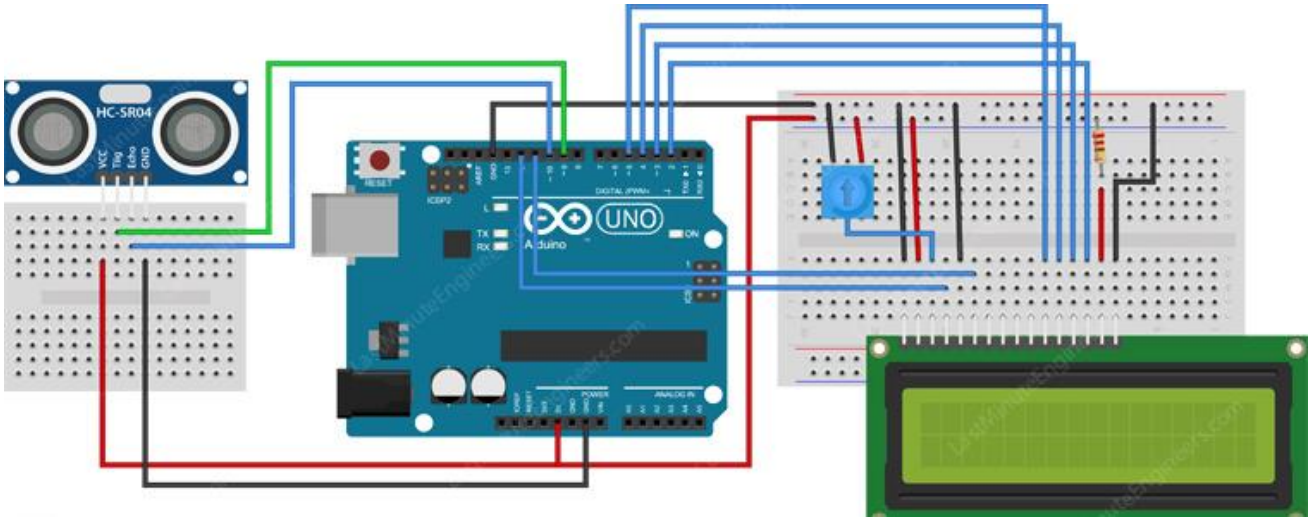
PROJECT

HC-SR04 Ultrasonic distance sensor Using ARDUINO UNO

HC-SR04 Ultrasonic distance sensor consists of two ultrasonic transducers. The one acts as a transmitter which converts electrical signal into 40 KHz ultrasonic sound pulses. The receiver listens for the transmitted pulses. If it receives them it produces an output pulse whose width can be used to determine the distance the pulse travelled. As simple as pie!

The sensor is small, easy to use in any robotics project and offers excellent non-contact range detection between 2 cm to 400 cm (that's about an inch to 13 feet) with an accuracy of 3mm. Since it operates on 5 volts, it can be hooked directly to an Arduino or any other 5V logic microcontrollers.

Circuit and Working



PIN CONNECTIONS

vss = ground
vdd = +5v
rw = ground
rs = 12pin ard
e/en = 11pin ard
d4 = 5 pin ard
d5 = 4 pin
d6 = 3 pin
d7 = 2 pin
k = ground
trig Pin = 9

echo Pin = 10

WORKING

It all starts, when a pulse of at least 10 µS (10 microseconds) in duration is applied to the Trigger pin. In response to that the sensor transmits a sonic burst of eight pulses at 40 KHz. This 8-pulse pattern makes the “ultrasonic signature” from the device unique, allowing the receiver to differentiate the transmitted pattern from the ambient ultrasonic noise.

The eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the Echo pin goes HIGH to start forming the beginning of the echo-back signal.

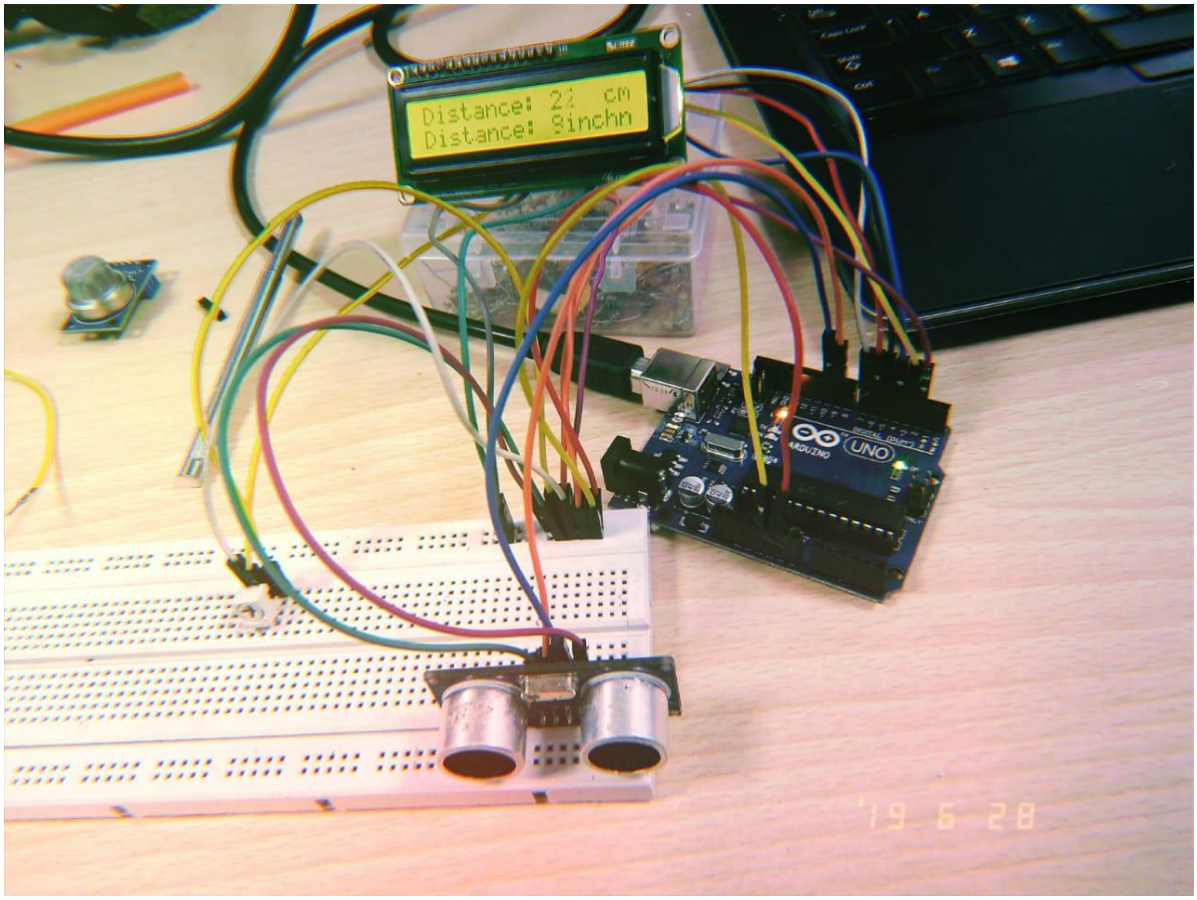
In case, If those pulses are not reflected back then the Echo signal will timeout after 38 mS (38 milliseconds) and return low. Thus a 38 mS pulse indicates no obstruction within the range of the sensor

If those pulses are reflected back the Echo pin goes low as soon as the signal is received. This produces a pulse whose width varies between 150 µS to 25 mS, depending upon the time it took for the signal to be received..

CODE

```
#include <LiquidCrystal.h>

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

const int trigPin = 9;
const int echoPin = 10;
long duration;
int distanceCm, distanceInch;

void setup() {

lcd.begin(16,2);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

}

void loop() {

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

duration = pulseIn(echoPin, HIGH);
distanceCm= duration*0.034/2;
distanceInch = duration*0.0133/2;

lcd.setCursor(0,0); // Sets the location at which subsequent text written to the LCD will be displayed
lcd.print("Distance: "); // Prints string "Distance" on the LCD
lcd.print(distanceCm); // Prints the distance value from the sensor
lcd.print("  cm");
delay(10);
lcd.setCursor(0,1);
```

```
lcd.print("Distance: ");
lcd.print(distanceInch);
lcd.print("inch");
delay(10);

}
```

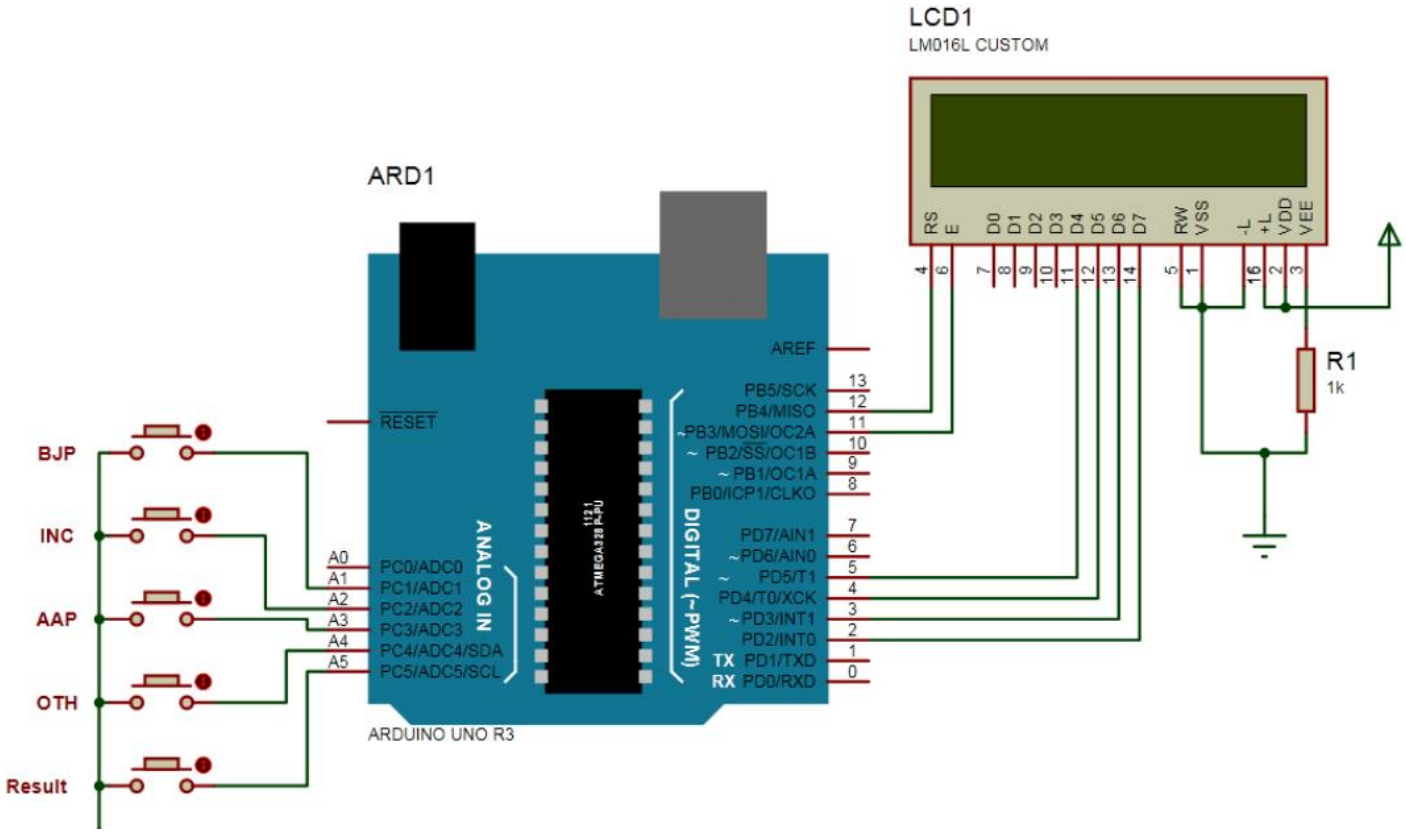


PROJECT

VOTING MACHINE with Arduino

In this project we have used four push buttons for four different candidates. We can increase the number of candidate but for better understanding we have limited it to four. When any voter press any of four button then respecting voting value will increment by one each time. After whole voting we will press result button to see the results. As the "result" button is pressed, arduino calculates the total votes of each candidate and show it on LCD display.

Circuit and Working



PIN CONNECTIONS

Control pin RS, RW and En are directly connected to arduino pin 12, GND and 11. And data pin D4-D7 is connected to pins 5, 4, 3 and 2 of arduino.

The five push buttons are directly connected with pin 15-19(A1-A5) of Arduino with respect to ground.

WORKING

Circuit of this project is quite easy which contains Arduino, push buttons and LCD. Arduino controls the complete processes like reading button, incrementing vote value,

generating result and sending vote and result to LCD. Here we have added five buttons in which first button is for BJP, second for INC, third is for AAP, forth is for OTH means others and last button is used for calculating or displaying results.

CODE

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
#define sw1 15
#define sw2 16
#define sw3 17
#define sw4 18
#define sw5 19
int vote1=0;
int vote2=0;
int vote3=0;
int vote4=0;
void setup()
{
  pinMode(sw1, INPUT);
  pinMode(sw2,INPUT);
  pinMode(sw3,INPUT);
  pinMode(sw4,INPUT);
  pinMode(sw5,INPUT);
  lcd.begin(16, 2);
  lcd.print("Voting Machine");
  lcd.setCursor(0,1);
  lcd.print("Circuit Digest");
  delay(3000);
  digitalWrite(sw1, HIGH);
  digitalWrite(sw2, HIGH);
  digitalWrite(sw3, HIGH);
  digitalWrite(sw4, HIGH);
  digitalWrite(sw5, HIGH);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("BJP");
  lcd.setCursor(4,0);
  lcd.print("INC");
  lcd.setCursor(8,0);
  lcd.print("AAP");
  lcd.setCursor(12,0);
  lcd.print("OTH");
```

```
}
void loop()
{
  lcd.setCursor(0,0);
  lcd.print("BJP");
  lcd.setCursor(1,1);
  lcd.print(vote1);
  lcd.setCursor(4,0);
  lcd.print("INC");
  lcd.setCursor(5,1);
  lcd.print(vote2);
  lcd.setCursor(8,0);
  lcd.print("AAP");
   lcd.setCursor(9,1);
  lcd.print(vote3);
  lcd.setCursor(12,0);
  lcd.print("OTH");
  lcd.setCursor(13,1);
  lcd.print(vote4);
  if(digitalRead(sw1)==0)
    vote1++;
    while(digitalRead(sw1)==0);
  if(digitalRead(sw2)==0)
    vote2++;
    while(digitalRead(sw2)==0);
  if(digitalRead(sw3)==0)
    vote3++;
    while(digitalRead(sw3)==0);
   if(digitalRead(sw4)==0)
    vote4++;
    while(digitalRead(sw4)==0);
   if(digitalRead(sw5)==0)
   {
     int vote=vote1+vote2+vote3+vote4;
     if(vote)
     {
      if((vote1 > vote2 && vote1 > vote3 && vote1 > vote4))
      {
       lcd.clear();
       lcd.print("BJP Wins");
       delay(2000);
       lcd.clear();
```

```
 }
 else if((vote2 > vote1 && vote2 > vote3 && vote2 > vote4))
 {
  lcd.clear();
  lcd.print("INC Wins");
  delay(2000);
  lcd.clear();
 }
 else if((vote3 > vote1 && vote3 > vote2 && vote3 > vote4))
 {
  lcd.clear();
  lcd.print("AAP Wins");
  delay(2000);
  lcd.clear();
 }
else if(vote4 > vote1 && vote4 > vote2 && vote4 > vote3)
{
  lcd.setCursor(0,0);
  lcd.clear();
  lcd.print("OTH Wins");
  delay(2000);
  lcd.clear();
}

 else if(vote4 > vote1 && vote4 > vote2 && vote4 > vote3)
{
  lcd.setCursor(0,0);
  lcd.clear();
  lcd.print("OTH Wins");
  delay(2000);
  lcd.clear();
}

else
{
  lcd.clear();
  lcd.print("   Tie Up Or   ");
  lcd.setCursor(0,1);
  lcd.print("   No Result   ");
  delay(1000);
  lcd.clear();
}
```

```
    }
     else
     {
       lcd.clear();
       lcd.print("No Voting....");
       delay(1000);
       lcd.clear();
     }
     vote1=0;vote2=0;vote3=0;vote4=0,vote=0;
     lcd.clear();
   }

}
```