UNIVERSITY OF MUMBAI

PRACTICAL JOURNAL ON

TNTERNET OF THINGS

SUBMITTED BY

ARSLAN MOMIN(202113)

UNDER THE GUIDANCE OF

PROF.RASHMITA PRADHAN

LATE BHAUSAHEB HIRAY S.S. TRUST'S
INSTITUTE OF COMPUTER APPLICATION

MUMBAI - 400051

MAHARASHTRA

MCA SEM II [2021-2022]

Name: Harshal Chavan Division: A
Roll No: 202124

LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE OF COMPUTER APPLICATION

ISO 9001-2008 CERTIFIED

S.N. 341, Next to New English School, Govt. Colony, Bandra (East), Mumbai – 400051, Tel: 91-22-26570892/3181

Date:

CERTIFICATE

This is to certify that Mr./Ms. Harshal Jaywant Chavan Roll No. 202124 is a student of FYMCA Semester-II has completed successfully full-semester practical/assignments of subject Internet Of Things for the academic year 2021 – 22.

Subject In-Charge **Director**

External Examiner

Name: Harshal Chavan Division: A
Roll No: 202124

INDEX

Name: Harshal Chavan Division: A
Roll No: 202124

9.	To interface LCD, potentiometer with Arduino and write a program to display message on LCD	
10.	To interface LCD, push button, potentiometer with Arduino and write a program to display the no. of times (count) the push button is pressed on LCD and display message on LCD when push button is pressed.	
11.	To interface LED's, potentiometer with Arduino and write a program to turn on or off more of the LEDs by turning the potentiometer knob.	
12.	To interface LED's, potentiometer with Arduino and write a program to implement all Analog signal functions.	
13.	To interface LED, Photo resistor (LDR) with Arduino and write a program to increase and decrease the brightness of the LED based on the amount of light present.	
14.	: To interface LED's with Arduino and write a program to show the fading effect on LED's.	
15.	To interface LM35 sensor with Arduino and write a program to display temperature data on serial monitor	
16.	To interface LM35 sensor with Arduino and write a program to display temperature data on LCD.	
17.	To interface PIR sensor with Arduino and write a program to turn on and off LED depending on motion detection	
18.	To interface Ultrasonic sensor with Arduino and write a program to display object distance(in inch/cm) in serial monitor depending on sound detection.	
19.	To interface servo motor with Arduino and write a program to sweep a servo back and forth through its full range of motion	
20.	To interface servo motor, Photo resistor (LDR) with Arduino and write a program to sweep a servo back and forth through its full range of motion depending on light.	

Name: Harshal Chavan Division: A
Roll No: 202124

21.	Write a program to control two DC motors with Arduino	
22.	To interface IR remote with Arduino Write a program to create useful commands from the IR remote control.	
23.	To interface RGB led and IR remote, write a program to control RGB led with IR remote	
24.	Write a program to control DC motor speed with an IR remote.	
25.	Upload data on Thingspeak cloud manually	
26.	To update readings to Thingspeak from Arduino using Tinkercad.	
27.	To interface Temperature sensor and ESP8266 with Arduino and update temperature reading to Thingspeak	
28.	To interface LDR sensor, LED and ESP8266 with Arduino and update light intensity values to Thingspeak and tweet “LIGHT ON” message on tweeter when light intensity value is less than 300	
29.	To interface Temperature sensor and ESP8266 with Arduino and update temperature values to Thingspeak and tweet “High Temp” message on tweeter when temperature value is greater than 40C.	

Name: Harshal Chavan Division: A
Roll No: 202124

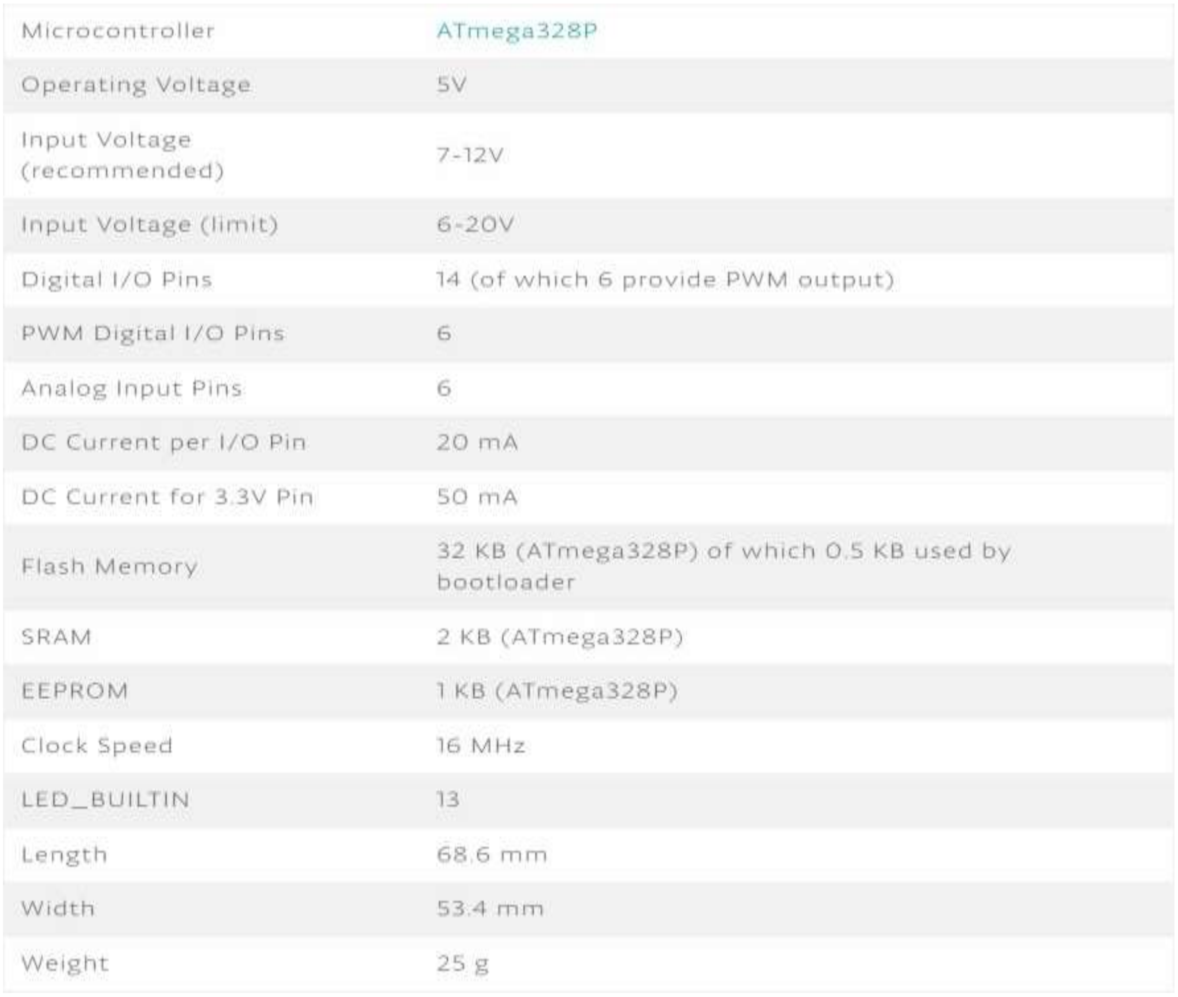
Practical No. 1

Aim: Program to blink Arduino onboard LED and to interface external LED with Arduino and write a program to turn ON LED for 1 sec after every 2 seconds.

Components: Arduino Uno, 1 Green LED, 1 Resistor of 100 Ω

Theory: Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button.

Name: Harshal Chavan Division: A
Roll No: 202124



Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

`digitalWrite():`

`Description:`

Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin.

Name: Harshal Chavan Division: A
Roll No: 202124

If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim. Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

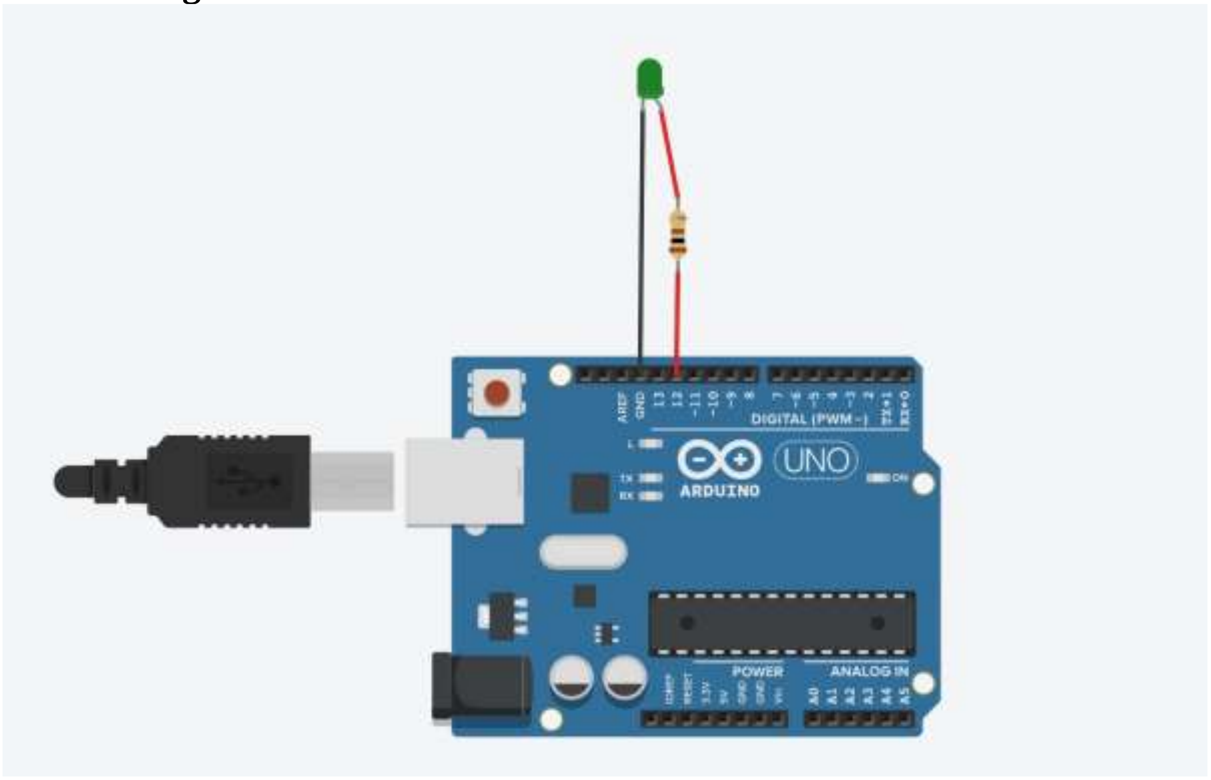
`Syntax:`

`digitalWrite(pin, value)`

`Parameters:`

`pin`: the Arduino pin number.
`value`: `HIGH` or `LOW`.

Circuit Diagram:

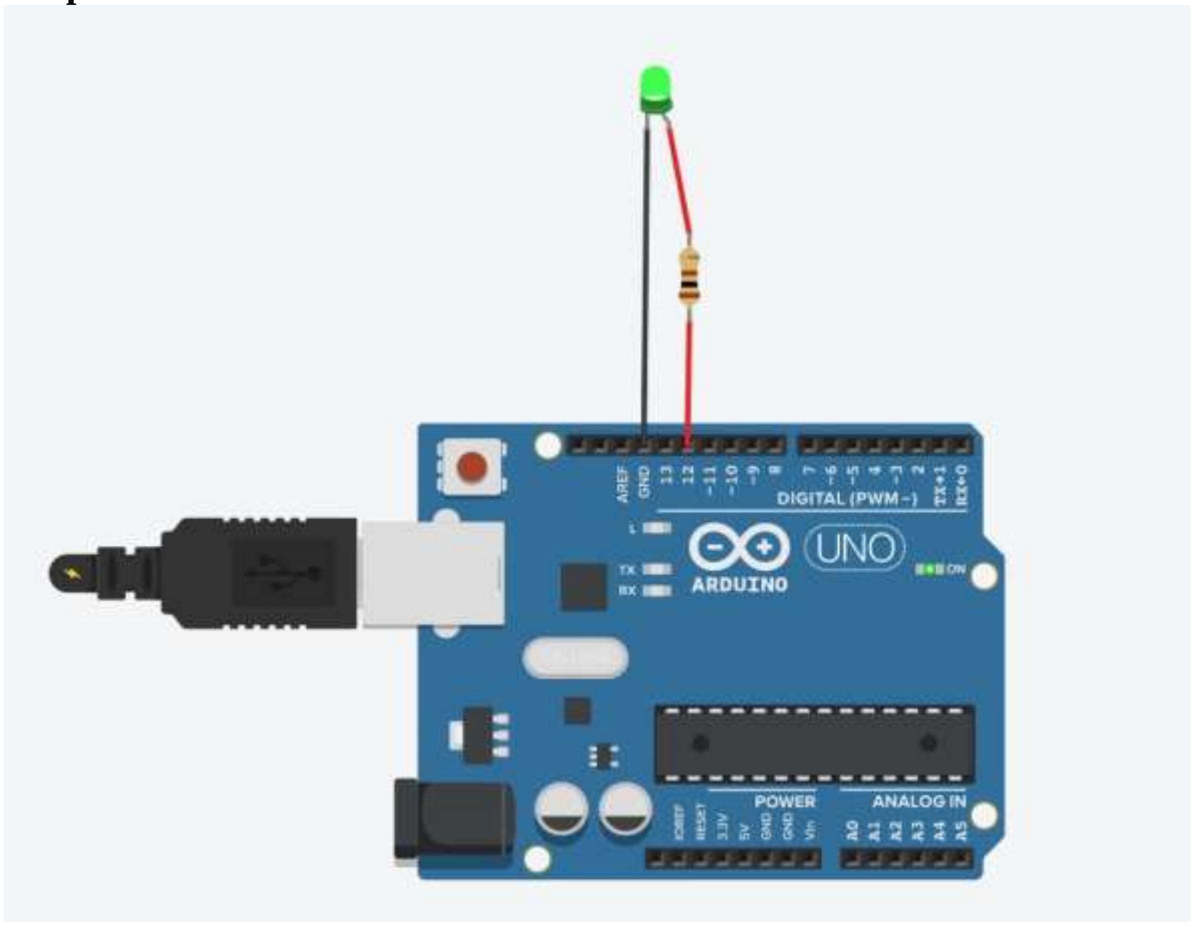


Program:

```
void setup()
{
  pinMode(12, OUTPUT);
}

void loop()
{
  digitalWrite(12, HIGH);
  delay(1000);
  digitalWrite(12, LOW);
  delay(2000);
}
```

Name: Harshal Chavan Division: A
Roll No: 202124

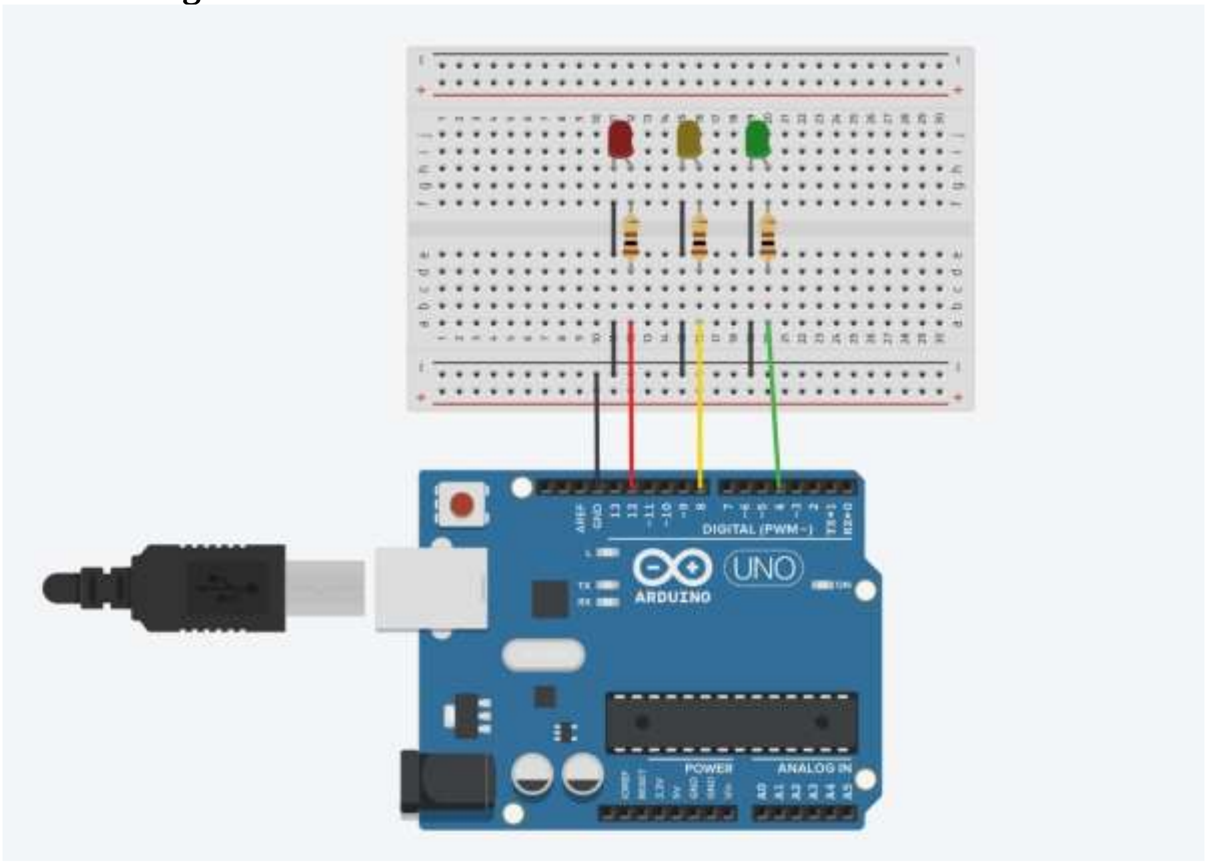
Output:



Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 2

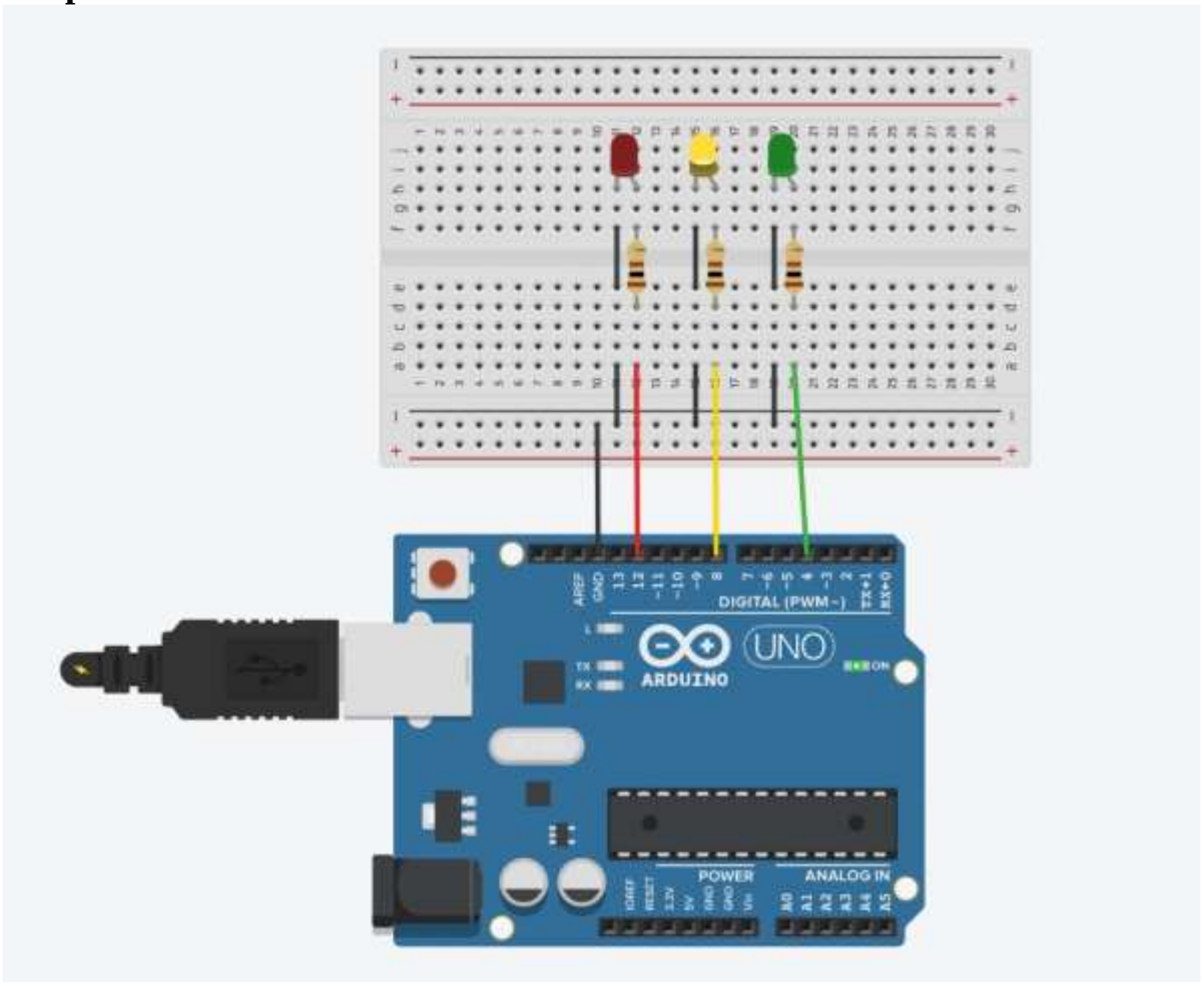
Aim: Program to build an Arduino Traffic Light Controller.
Component: Arduino Uno, 3 LEDs (Red, Yellow, Green), 3 Resistors of 100-ohm
Theory: For traffic light controller we have used 3 LEDs and turning ON using some dalay
Circuit Diagram:



Program:

```
void setup()
{
  pinMode(4,   OUTPUT);
  pinMode(8,   OUTPUT);
  pinMode(12, OUTPUT);
}
void loop()
{
  digitalWrite(4, LOW);
  digitalWrite(8, LOW);
  digitalWrite(12, HIGH);
  delay(3000);
  digitalWrite(4, LOW);
  digitalWrite(8, HIGH);
  digitalWrite(12, LOW);
  delay(1000);
  digitalWrite(4, HIGH);
  digitalWrite(8, LOW);
  digitalWrite(12, LOW);
  delay(2000);
}
```

Output:



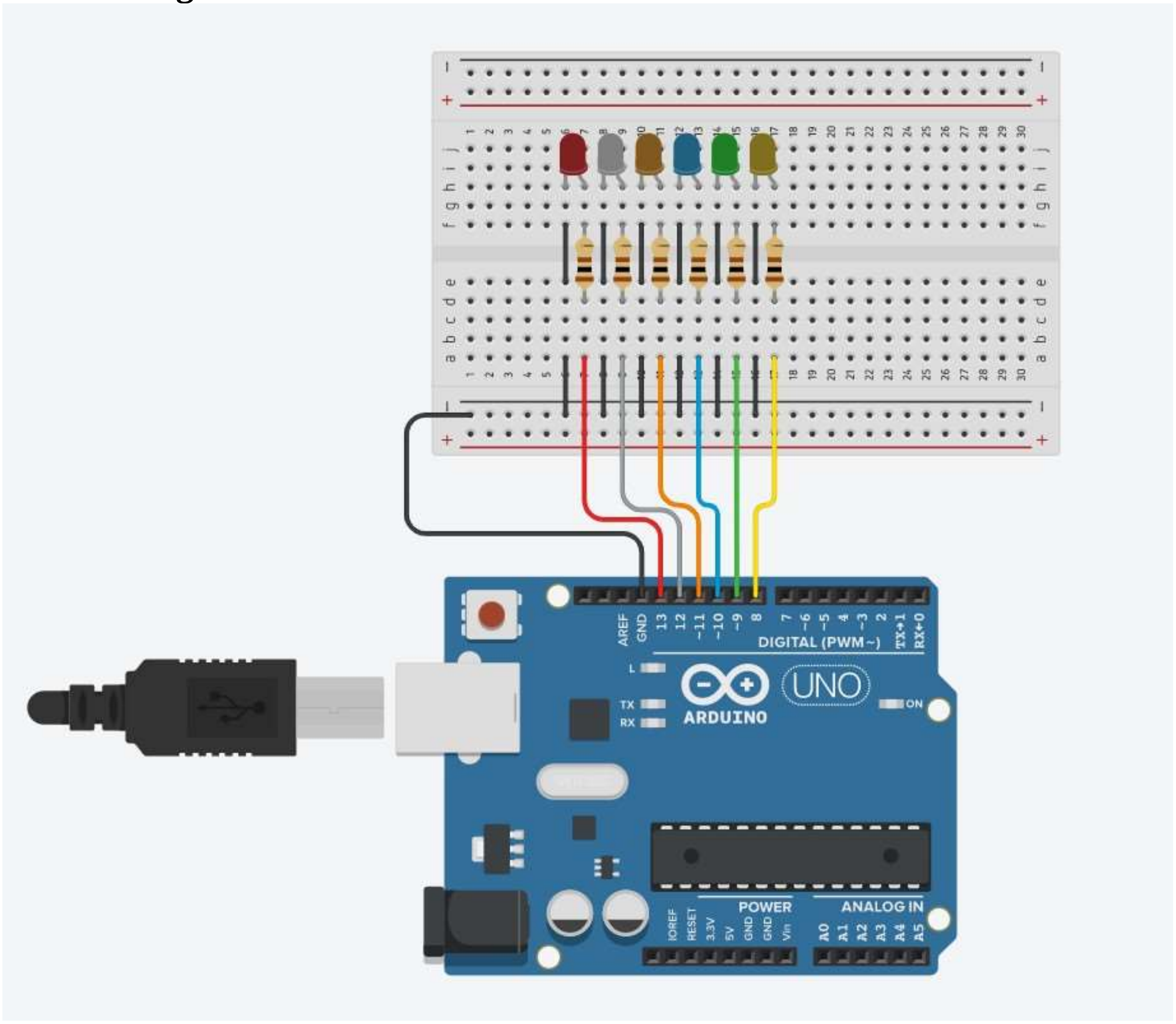
Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 3

Aim: To interface 5 LEDs with Arduino and write a program to blink 6 LEDs, one at a time, in a back-and-forth formation

Component: Arduino Uno, 6 different colour LEDs, 6 Resistors of 100 Ω

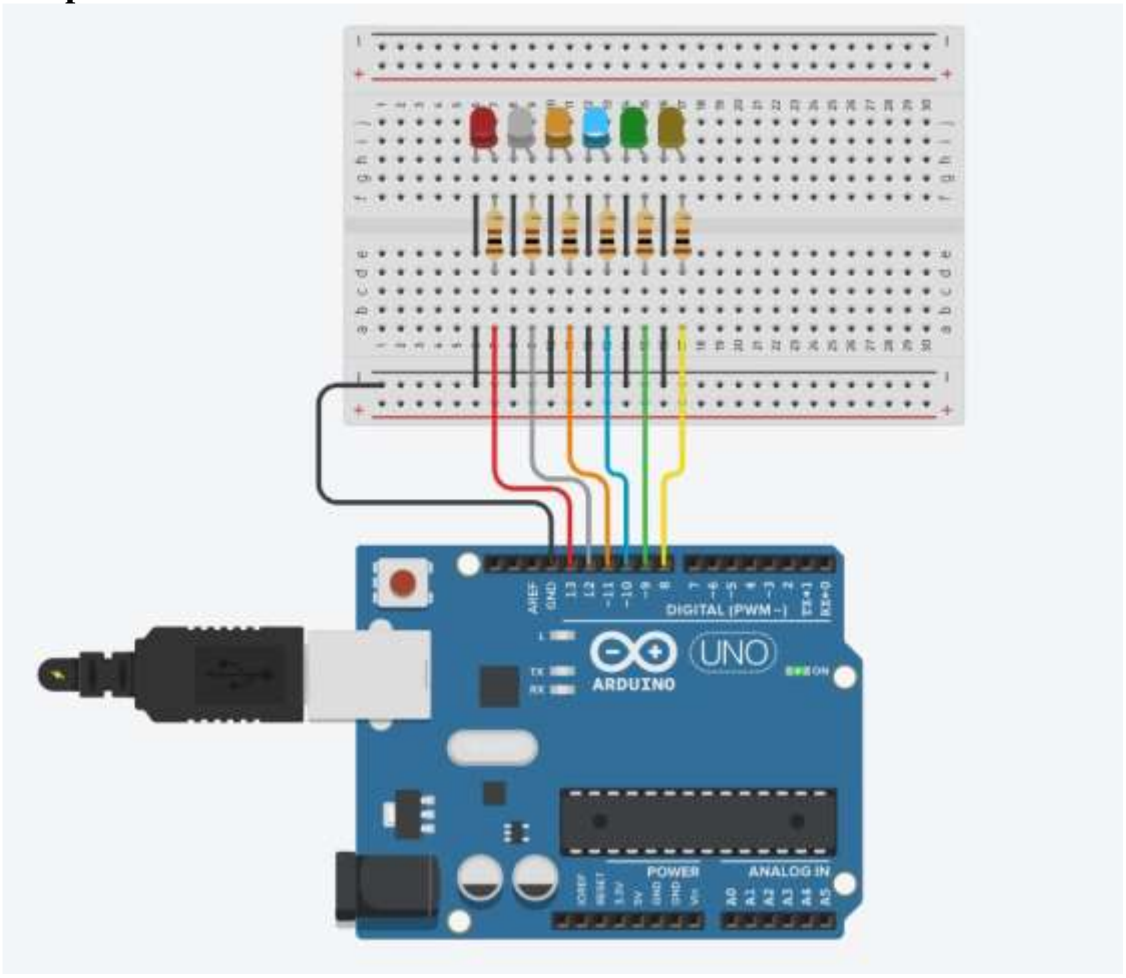
Circuit Diagram:



Program:

```
void setup()
{
  for(int pin = 8; pin <= 13; pin++)
          pinMode(pin, OUTPUT);
}

void loop()
{
  for(int pin = 8; pin <=13; pin++){
          digitalWrite(pin, HIGH);
          delay(100);
          digitalWrite(pin, LOW);
  }
  for(int pin = 13; pin >= 8; pin--){
    digitalWrite(pin, HIGH);
            delay(100);
            digitalWrite(pin, LOW);
  }
}
```

Output:



Name: Harshal Chavan   Division: A
Roll No: 202124

Practical No. 4

Aim: To interface Push button with Arduino and write a program to turn ON LED when push button is pressed.

Component: Arduino Uno, 2 Pushbutton, 2 different LEDs, 4 Resistors of 100 Ω

Theory:

Pushbutton: - The pushbutton is a component that connects two points in a circuit when you press it. The example turns on an LED when you press the button.

Resistors: - Resistors are electronic components, which offer resistance against the current flow, or speaking at a deeper level, against the electrons' flow. Resistors, denoted by R, are passive components, which means that they don't generate any electricity at all, but rather reduce voltage and current by dissipating power in the form of heat.
The unit of resistance is ohms (Ω) and resistors are usually built using carbon or metal wire.

digitalRead()
Reads the value from a specified digital pin, either HIGH or LOW.

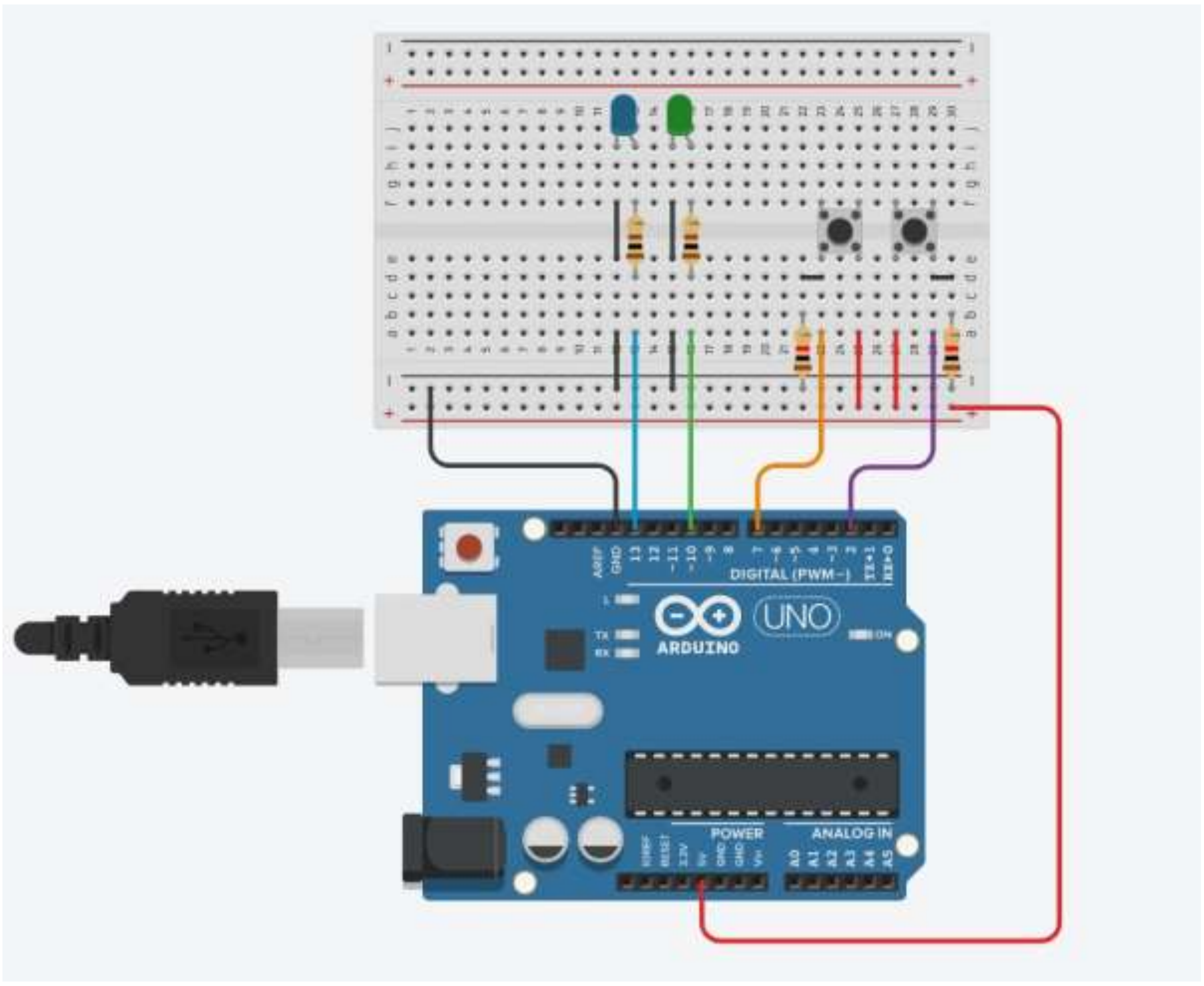
Syntax
digitalRead(pin)

Parameters
pin: the Arduino pin number you want to read

Returns
HIGH or LOW

Circuit Diagram:

Name: Harshal Chavan Division: A
Roll No: 202124



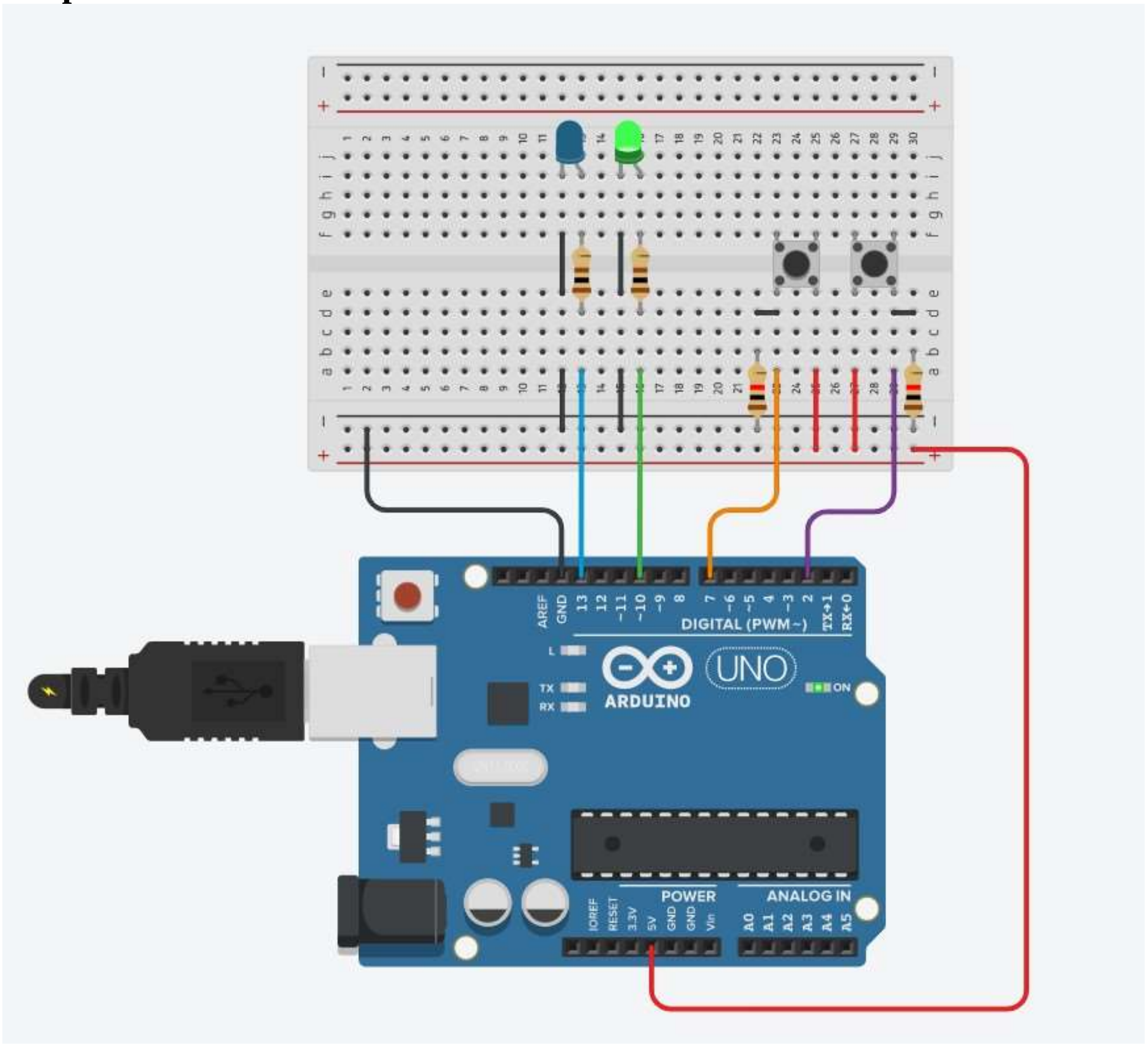
Program:

```
int button1 = 2;
int button2 = 7;
void setup()
{
  pinMode(10, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(button1, INPUT);
  pinMode(button2, INPUT);
}
void loop()
{
  int value1 = digitalRead(button1);
  int value2 = digitalRead(button2);
  if(value1 == HIGH)
  digitalWrite(10, HIGH);
  else
    digitalWrite(10, LOW);

  if(value2 == HIGH)
  digitalWrite(13, HIGH);
  else
    digitalWrite(13, LOW);
```

Name: Harshal Chavan Division: A
Roll No: 202124

}

Output:



Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 5

Aim: To interface Push button, Speaker/buzzer with Arduino and write a program to turn ON LED and generate a note or tone when push button is pressed.

Component: Arduino Uno, 1 Pushbutton, 1 Buzzer, 3 Resistors

Theory:

Buzzer: - A piezo buzzer is pretty sweet. It's not liked a regular speaker that you might think of. It uses a material that's piezoelectric, it actually changes shape when you apply electricity to it. By adhering a piezo-electric disc to a thin metal plate, and then applying electricity, we can bend the metal back and forth, which in turn creates noise.
The faster you bend the material, the higher the pitch of the noise that's produced. This rate is called frequency. Again, the higher the frequency, the higher the pitch of the noise we hear.

tone():- Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.
Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to tone() will have no effect. If the tone is playing on the same pin, the call will set its frequency.
Use of the tone() function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).
It is not possible to generate tones lower than 31Hz.

Syntax
tone(pin, frequency)
tone(pin, frequency, duration)

Parameters
pin: the Arduino pin on which to generate the tone.
frequency: the frequency of the tone in hertz. Allowed data types: unsigned int.
duration: the duration of the tone in milliseconds (optional).

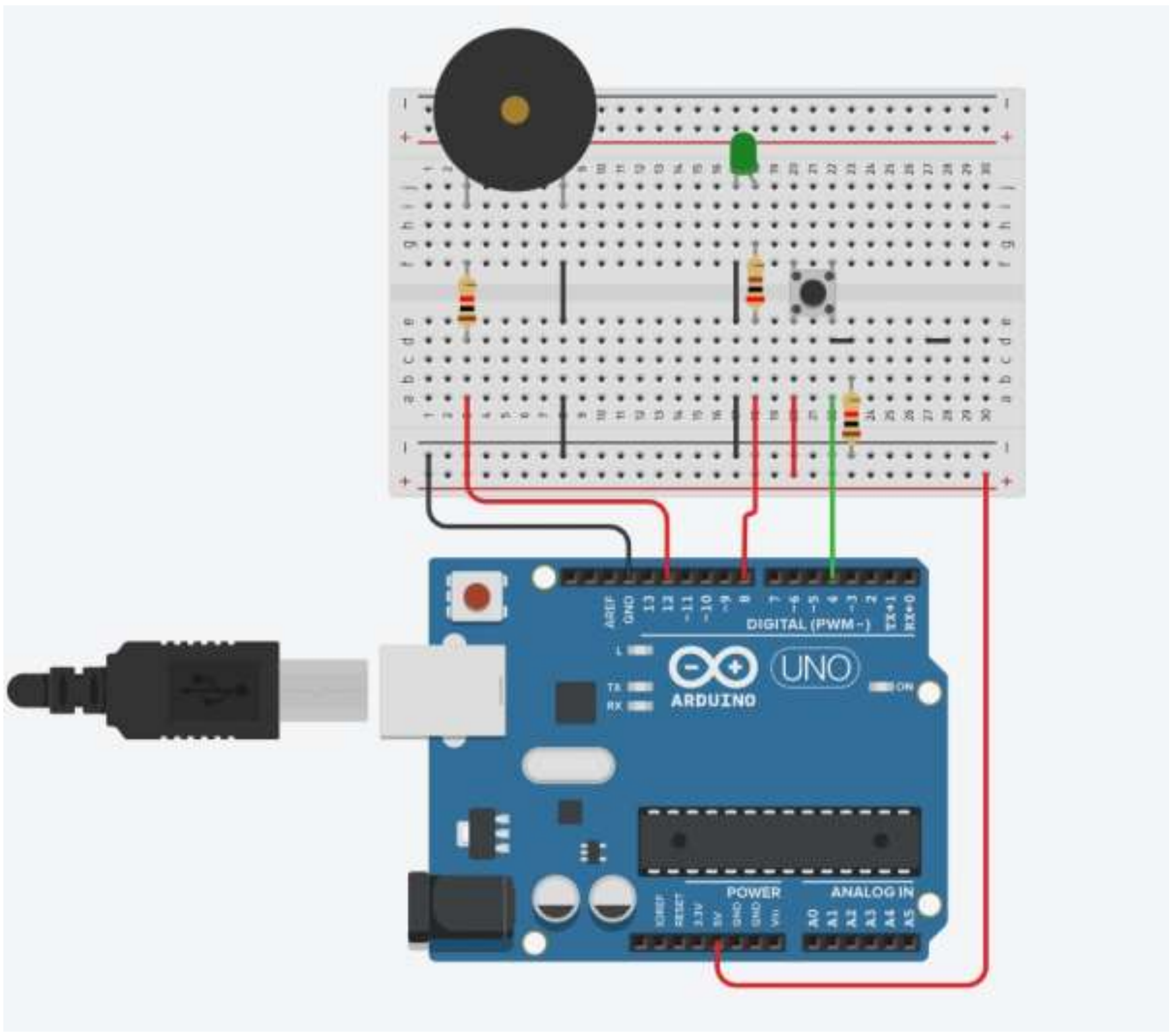
noTone(): - Stops the generation of a square wave triggered by tone(). Has no effect if no tone is being generated.

Syntax
noTone(pin)

Parameters
pin: the Arduino pin on which to stop generating the tone

Circuit Diagram:

Name: Harshal Chavan Division: A
Roll No: 202124



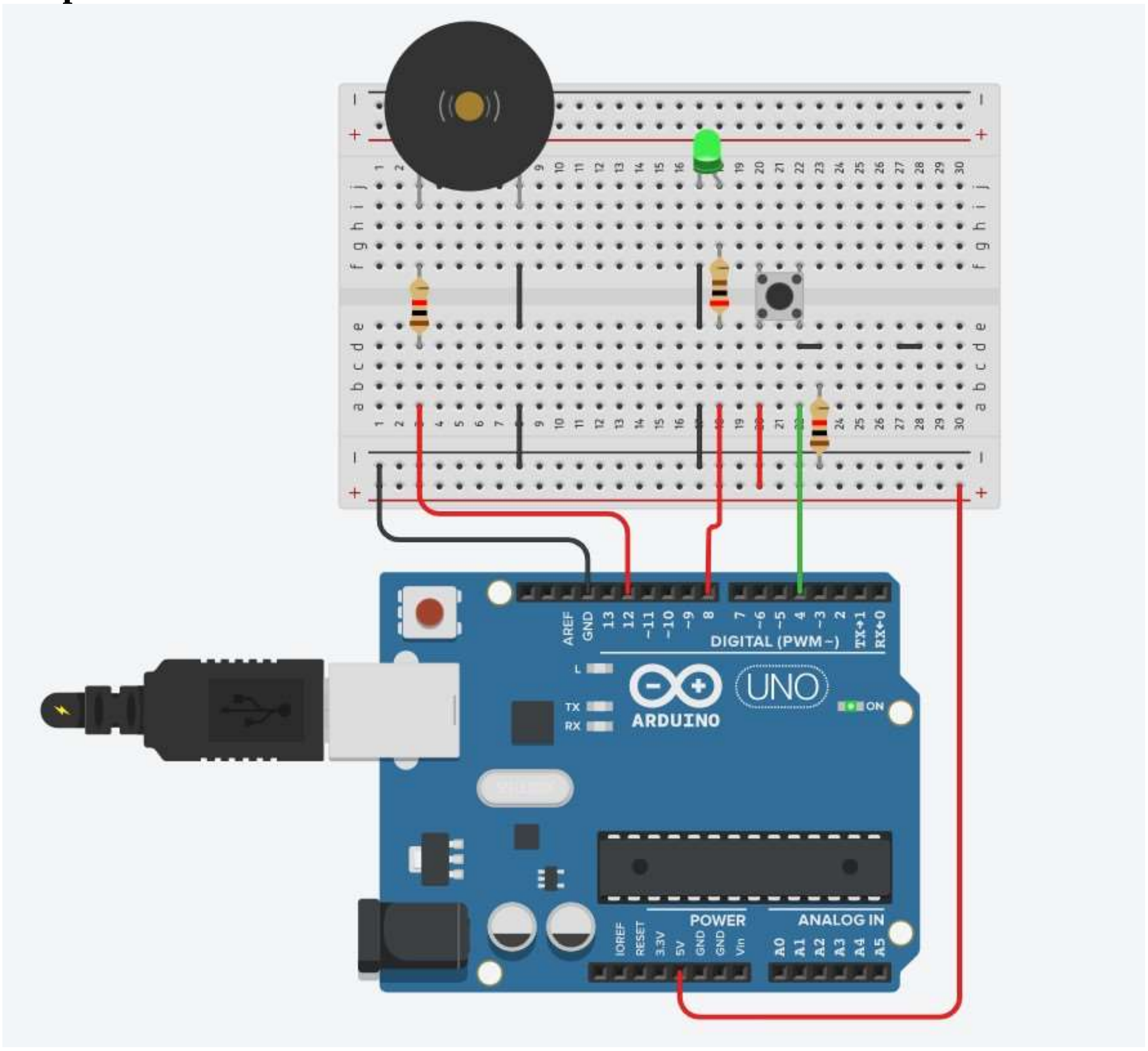
Program:

```
int button1 = 4;
void setup()
{
  pinMode(8, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(button1, INPUT);
}
void loop()
{
  int buttonStatus1 = digitalRead(button1);
   if(buttonStatus1 == HIGH)
  {
    digitalWrite(8, HIGH);
    tone(12,1000);
  }
  else
  {
    digitalWrite(8, LOW);
    noTone(12);
  }
```

Name: Harshal Chavan Division: A
Roll No: 202124

}

Output:



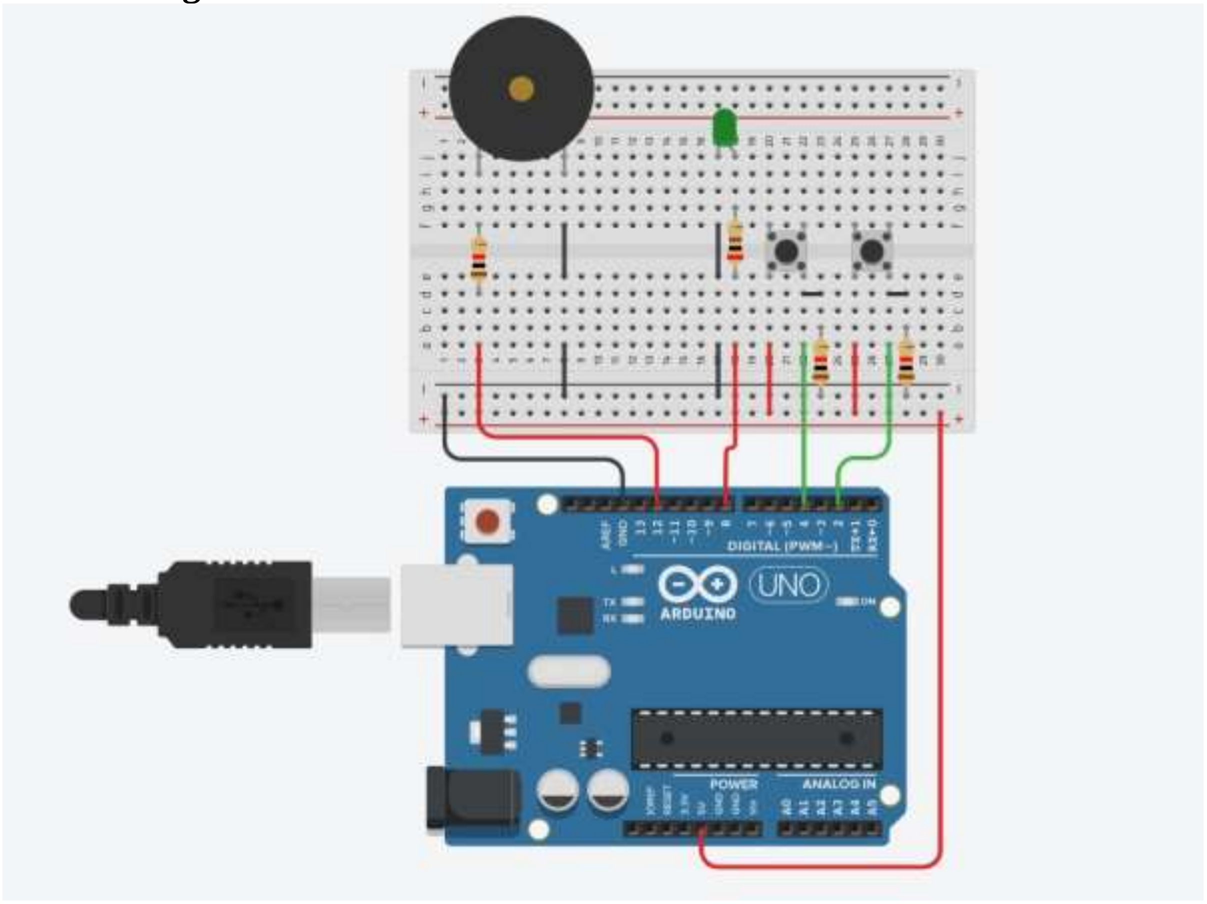
Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 6

Aim: To interface 2 Push buttons, a Speaker with Arduino and write a program to turn ON LED and generate 2 different notes on two button keyboards.

Component: Arduino Uno, 1 LED, 2 Pushbuttons, 1 Buzzer, 4 Resistors.

Circuit Diagram:



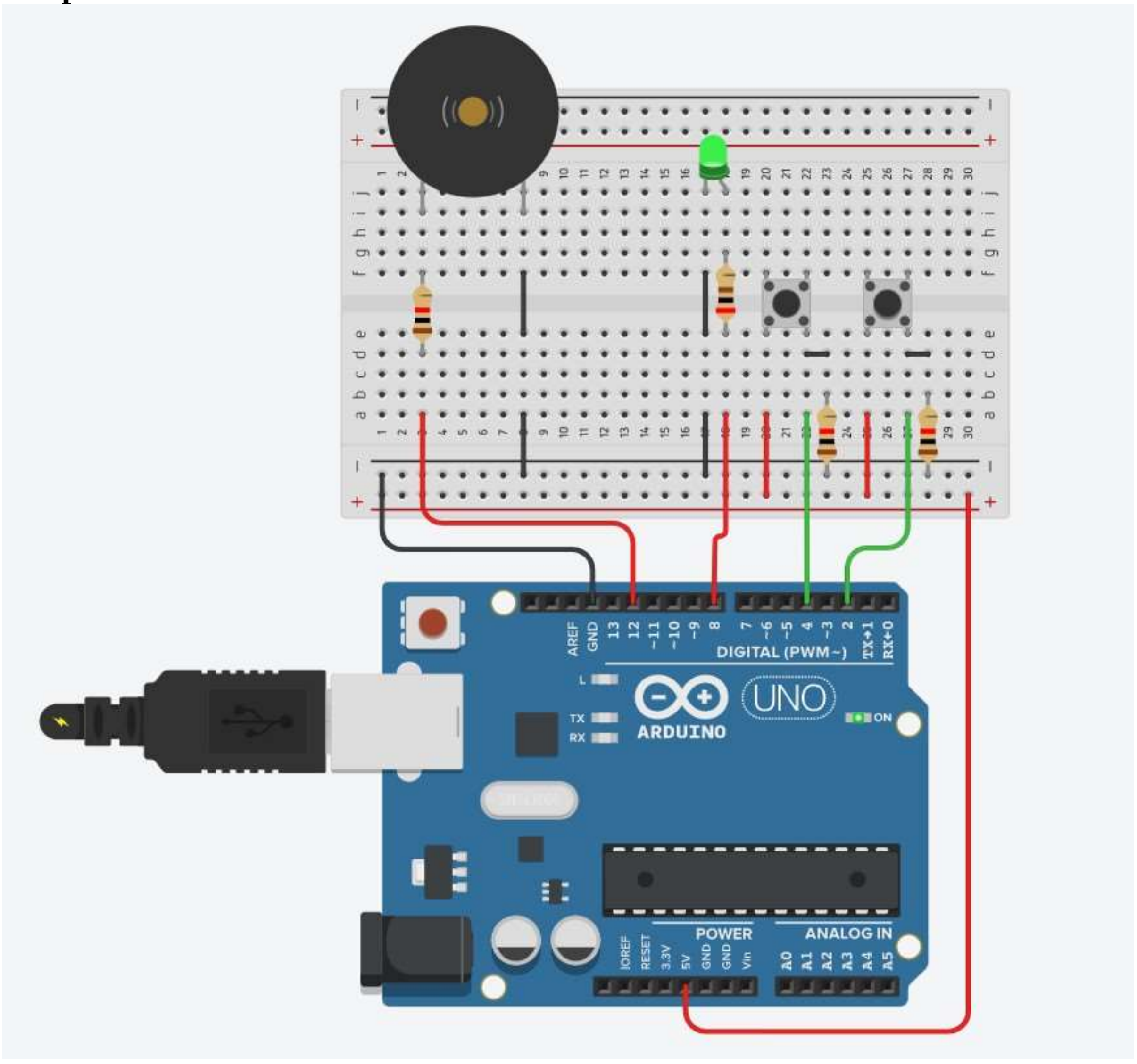
Program:

```
int button1 = 4;
int button2 = 2;
void setup()
{
  pinMode(8, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(button1, INPUT);
  pinMode(button2, INPUT);
}
void loop()
{
  int buttonStatus1 = digitalRead(button1);
  int buttonStatus2 = digitalRead(button2);
  if(buttonStatus1 == HIGH)
  {
    digitalWrite(8, HIGH);
    tone(12,1000);
  }
```

```
 else if(buttonStatus2 == HIGH)
 {
   digitalWrite(8, HIGH);
   tone(12,2000);
 }
 else
 {
   digitalWrite(8, LOW);
   noTone(12);
 }
}
```

Name: Harshal Chavan Division: A
Roll No: 202124

Output:



Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 7

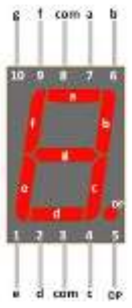
Aim: To interface Seven Segment Display (SSD) with Arduino and write a program to blink SSD.

Component: Arduino Uno, Cathode 7 Segment Display, 7 Resistors of 115 Ω

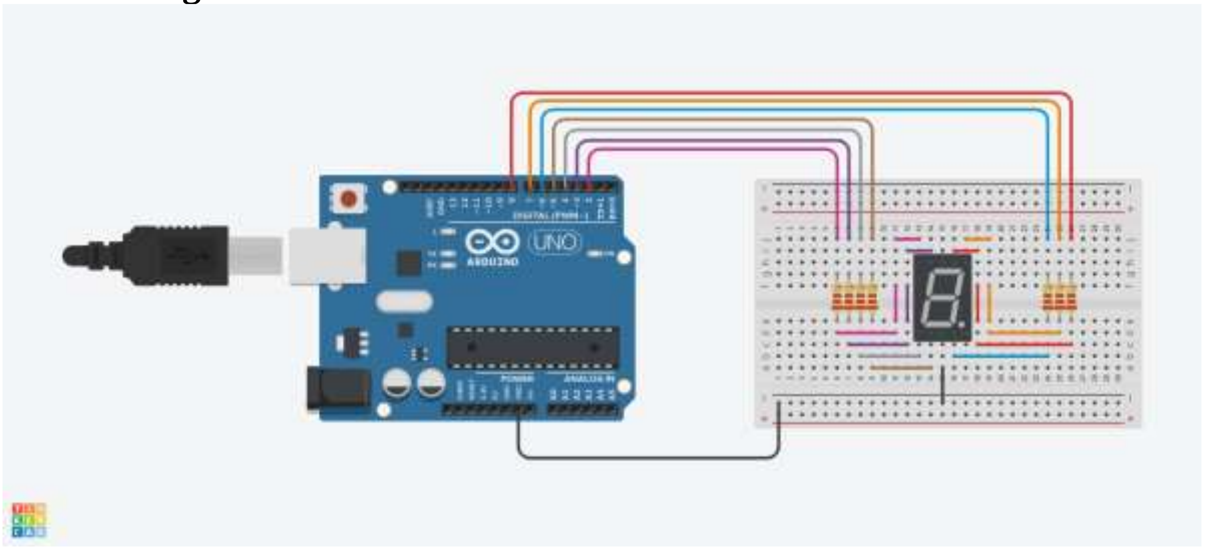
Theory:

The 7 Segment display: - 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.

Each one of the seven LEDs in the display is given a positional segment with one of its connection pins being brought straight out of the rectangular plastic package. These individually LED pins are labelled from a through to g representing each individual LED. The other LED pins are connected together and wired to form a common pin.



Circuit Diagram:



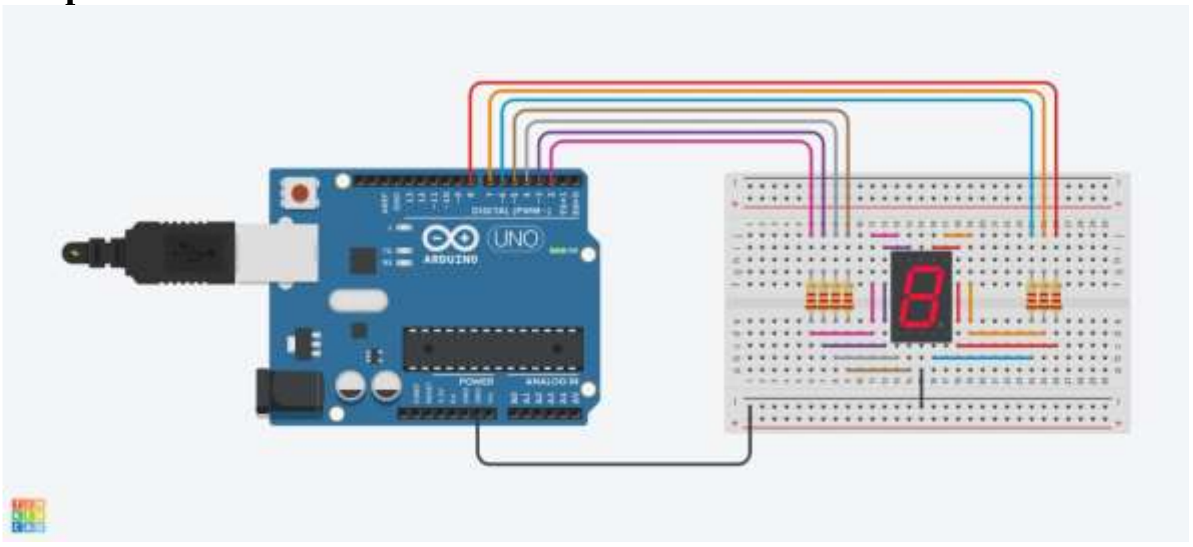
Program:

```
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
```

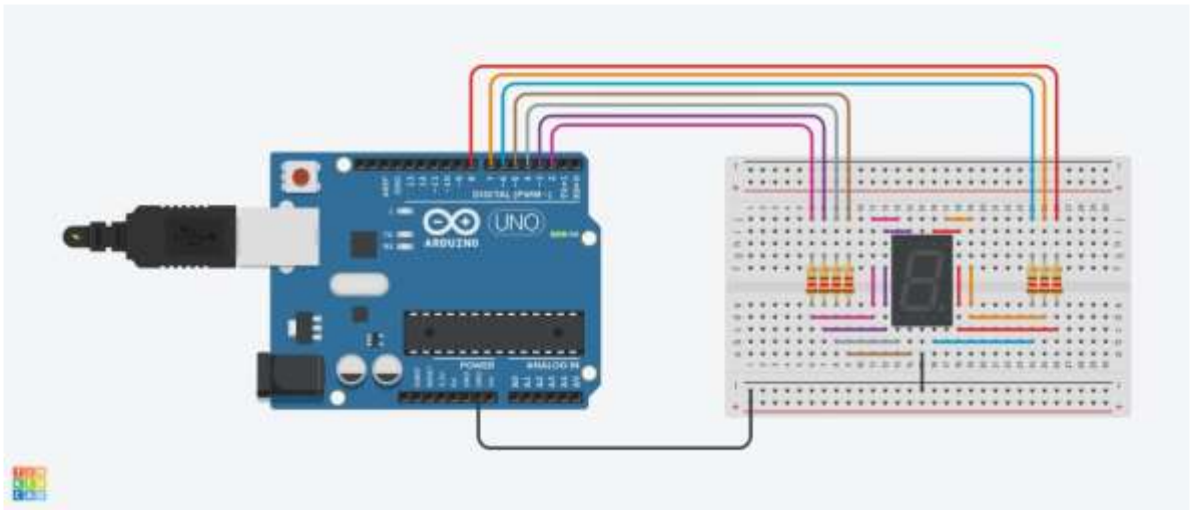
Name: Harshal Chavan Division: A
Roll No: 202124

```
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  delay(500); // Wait for 500 millisecond(s)
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  delay(300); // Wait for 300 millisecond(s)}
```

Output:



Name: Harshal Chavan Division: A
Roll No: 202124



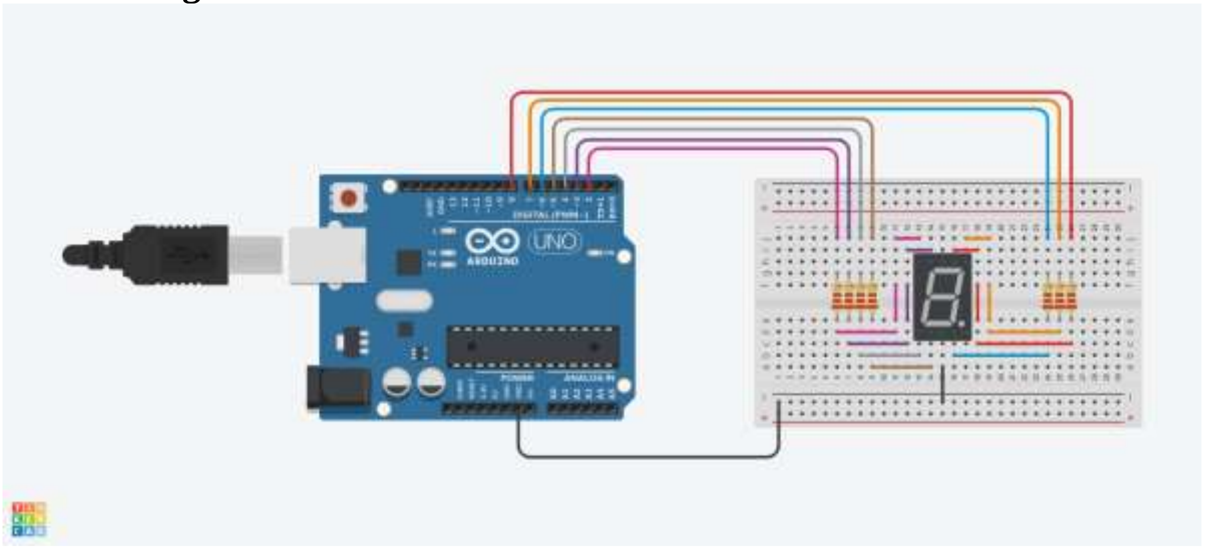
Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 8

Aim: To interface Seven Segment Display (SSD) with Arduino and write a program to print numbers from 1 to 4 on SSD.

Component: Arduino Uno, Cathode 7 Segment Display, 7 Resistors of 115 Ω

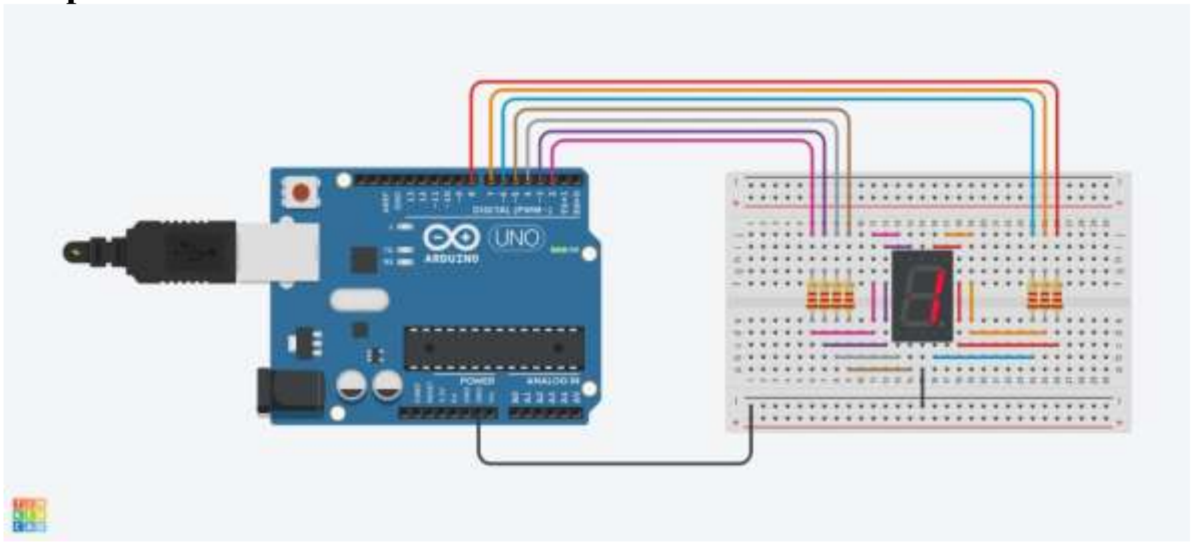
Circuit Diagram:



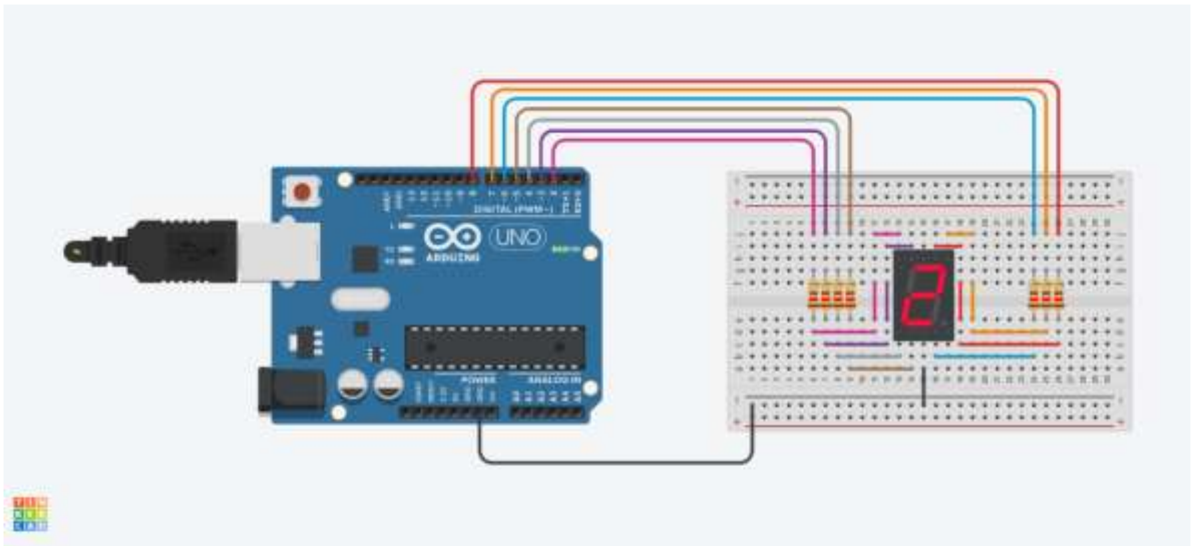
Program:

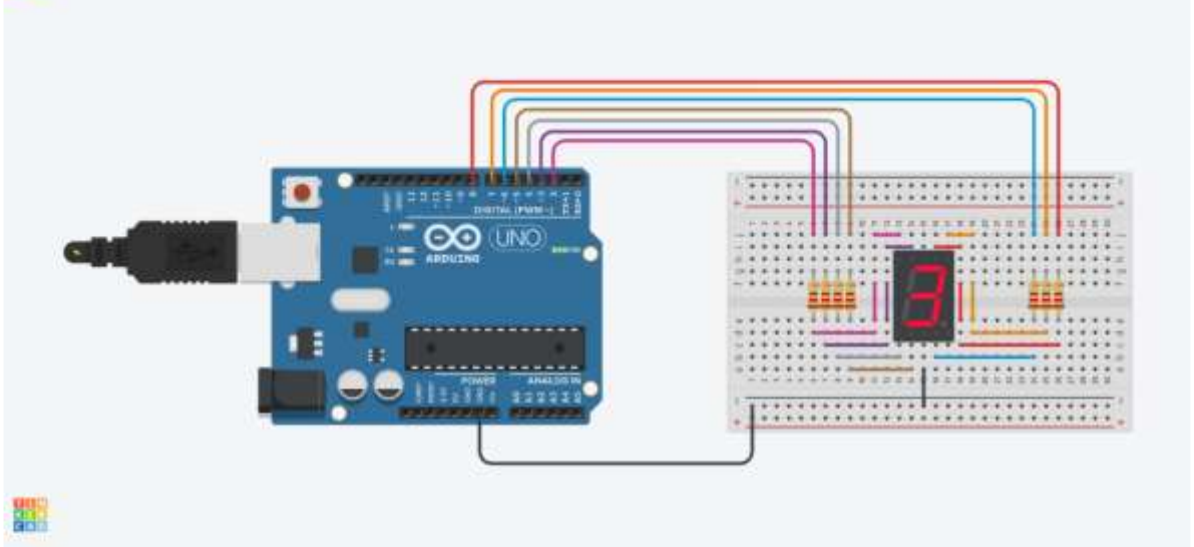
```
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop()
{
  digitalWrite(2,  LOW);
  digitalWrite(3,  HIGH);
  digitalWrite(4,  HIGH);
  digitalWrite(5,  HIGH);
  digitalWrite(6,  HIGH);
  digitalWrite(7,  HIGH);
  digitalWrite(8, HIGH);
  delay(2000); // Wait for 2000 millisecond(s)
  digitalWrite(2, LOW);
  digitalWrite(3,  LOW);
  digitalWrite(4,  LOW);
  digitalWrite(5,  LOW);
  digitalWrite(6,  HIGH);
  digitalWrite(7,  HIGH);
  digitalWrite(8, LOW);
  delay(2000); // Wait for 2000 millisecond(s)
  digitalWrite(2, HIGH);
  digitalWrite(3,  LOW);
  digitalWrite(4,  HIGH);
  digitalWrite(5,  HIGH);
  digitalWrite(6,  LOW);
  digitalWrite(7,  HIGH);
  digitalWrite(8, HIGH);
  delay(2000); // Wait for 2000 millisecond(s)
  digitalWrite(2, HIGH);
  digitalWrite(3,  LOW);
  digitalWrite(4,  LOW);
  digitalWrite(5,  HIGH);
  digitalWrite(6,  HIGH);
  digitalWrite(7,  HIGH);
  digitalWrite(8, HIGH);
  delay(2000); // Wait for 2000 millisecond(s)
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4,  LOW);
  digitalWrite(5,  LOW);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
  delay(2000); // Wait for 2000 millisecond(s)
}
```

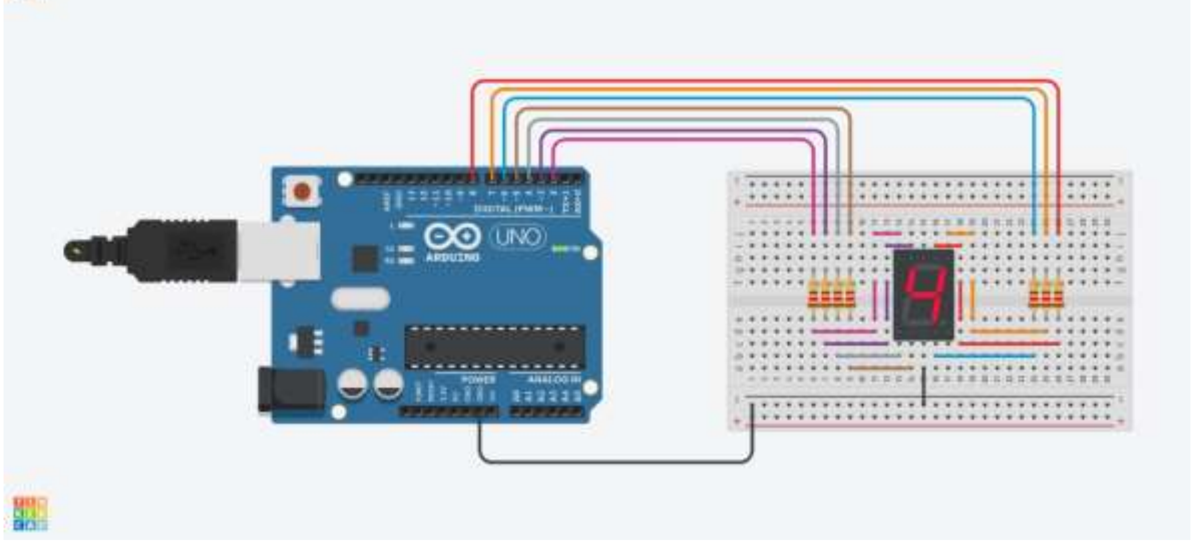
Output:



Name: Harshal Chavan Division: A
Roll No: 202124







Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 9

Aim: To interface LCD, potentiometer with Arduino and write a program to display message on LCD.

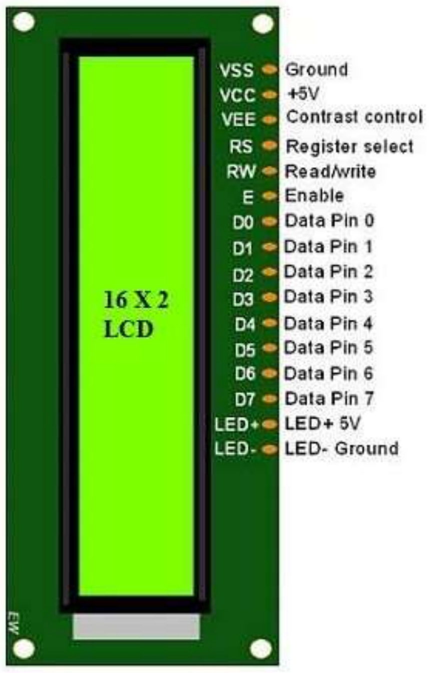
Components: Arduino Uno, LCD 16 x 2, 100 Ω Potentiometer, 1 kΩ Resistor

Theory:

LCD:

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

LCD 16×2 Pin Diagram



The 16×2 LCD pinout is shown below.

Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.

Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.

Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.

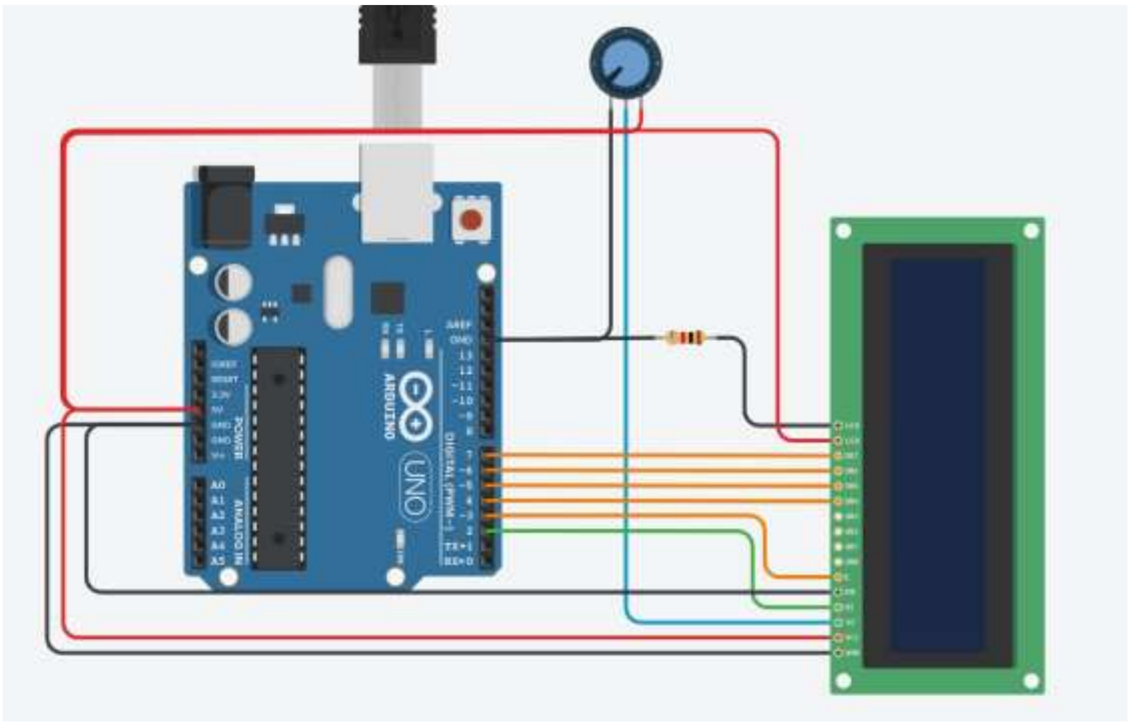
Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).

Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).

Name: Harshal Chavan Division: A
Roll No: 202124

Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.
Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
Pin15 (+ve pin of the LED): This pin is connected to +5V
Pin 16 (-ve pin of the LED): This pin is connected to GND.

Circuit Diagram:

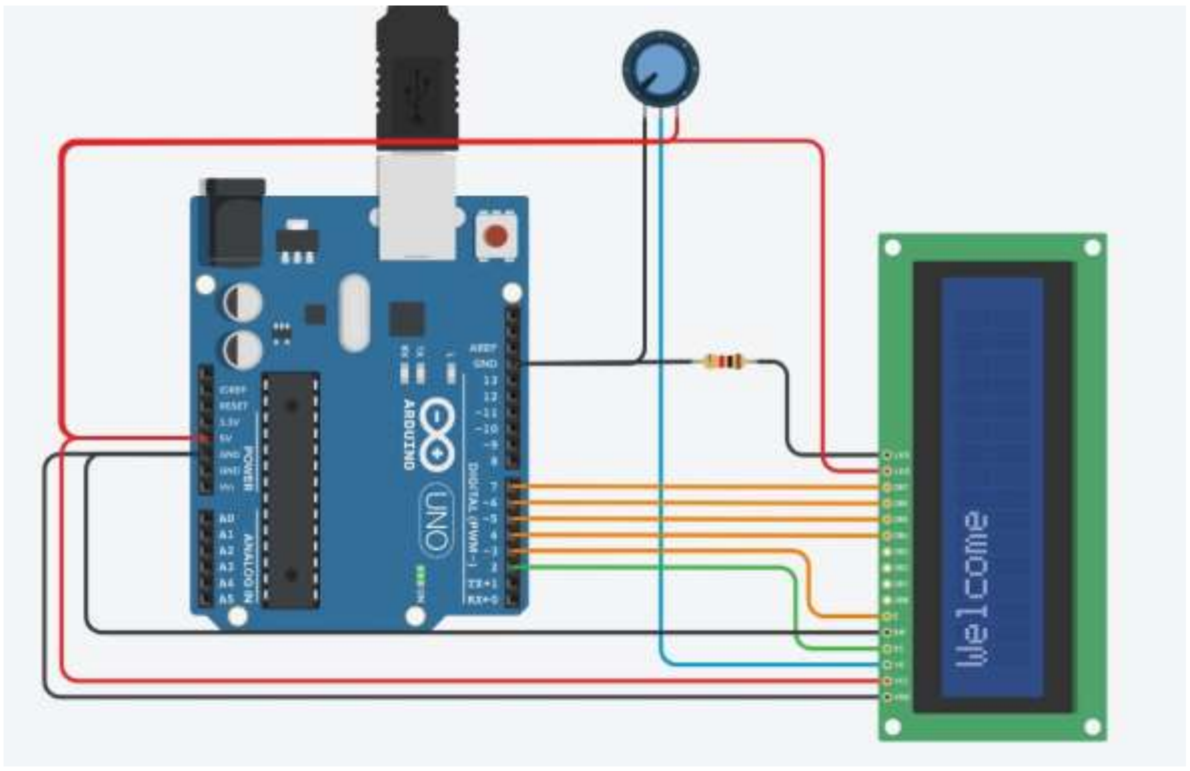


Program:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(2,3,4,5,6,7);
void setup()
{
  lcd.begin(16,2);
  lcd.clear();//start with a blank screen
  lcd.setCursor(0,0);//set the cursor to col 0 and row 0
  lcd.print("Welcome");
}

void loop()
{
}
```

Output:

Name: Harshal Chavan Division: A
Roll No: 202124



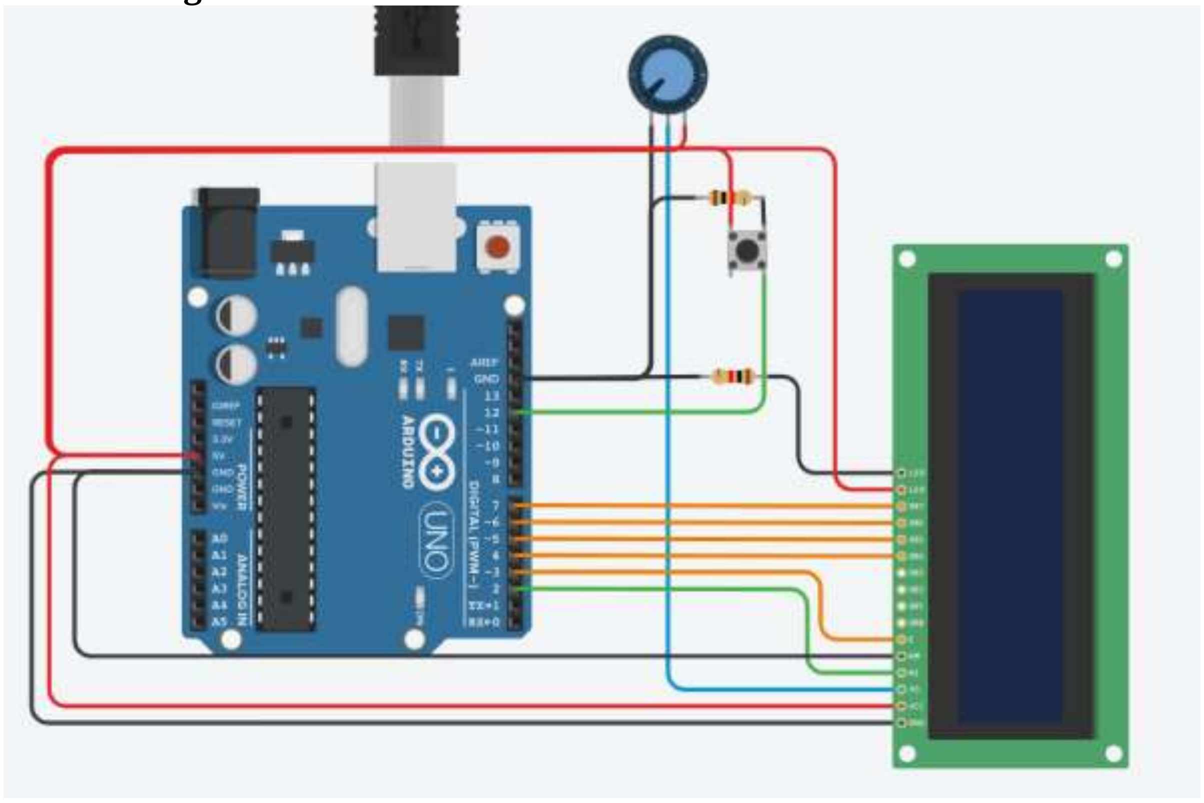
Name: Harshal Chavan   Division: A
Roll No: 202124

Practical No. 10

Aim: To interface LCD, push button, potentiometer with Arduino and write a program to display the no. of times (count) the push button is pressed on LCD and display message on LCD when push button is pressed.

Components: Arduino Uno, LCD 16 x 2, 100 Ω Potentiometer, 2pcs. 1 kΩ Resistor, Pushbutton

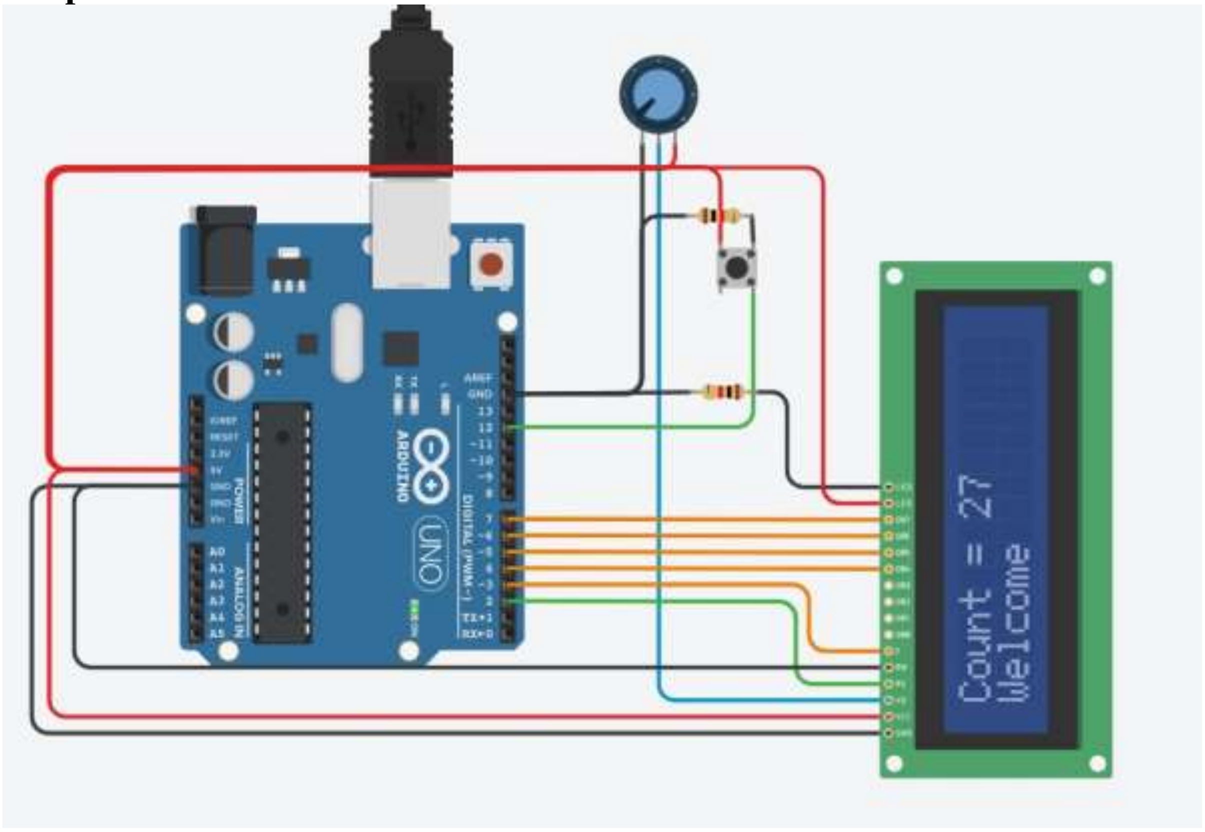
Circuit Diagram:



Program:

```
#include <LiquidCrystal.h>
int button1 = 12;
int count = 0;
LiquidCrystal lcd(2,3,4,5,6,7);
void setup()
{
  lcd.begin(16,2);
}
void loop()
{
  int buttonStatus1 = digitalRead(button1);
  if(buttonStatus1 == HIGH)  {
    count = count+1;
    lcd.setCursor(0,1);
    lcd.print("Welcome");
  }
  else {
    lcd.clear();  }
  lcd.setCursor(0,0);
  lcd.print("Count = ");
  lcd.print(count);
  delay(125);
}
```

Output:



Name: Harshal Chavan   Division: A
Roll No: 202124

Practical No. 11

Aim: To interface LED's, potentiometer with Arduino and write a program to turn on or off more of the LEDs by turning the potentiometer knob.

Components: Arduino Uno, 250 kΩ Potentiometer, 100 Ω Resistor, Red LED.

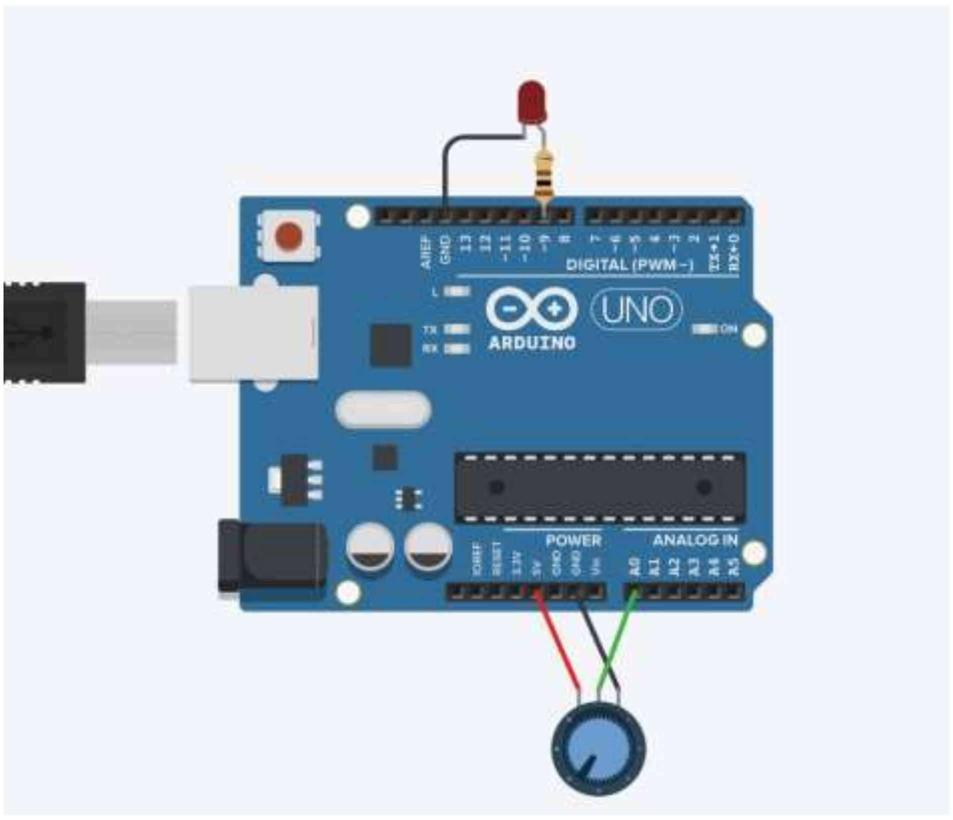
Theory:

A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider.[1] If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.
The measuring instrument called a potentiometer is essentially a voltage divider used for measuring electric potential (voltage); the component is an implementation of the same principle, hence its name.

Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment. Potentiometers operated by a mechanism can be used as position transducers, for example, in a joystick. Potentiometers are rarely used to directly control significant power (more than a watt), since the power dissipated in the potentiometer would be comparable to the power in the controlled load.

Circuit Diagram:

Name: Harshal Chavan Division: A
Roll No: 202124

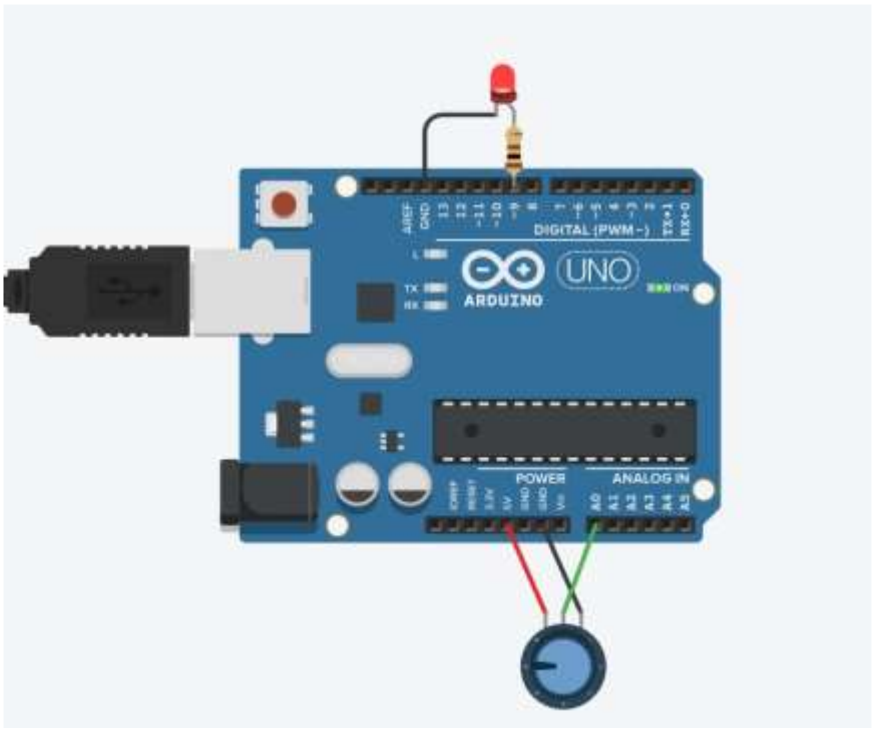


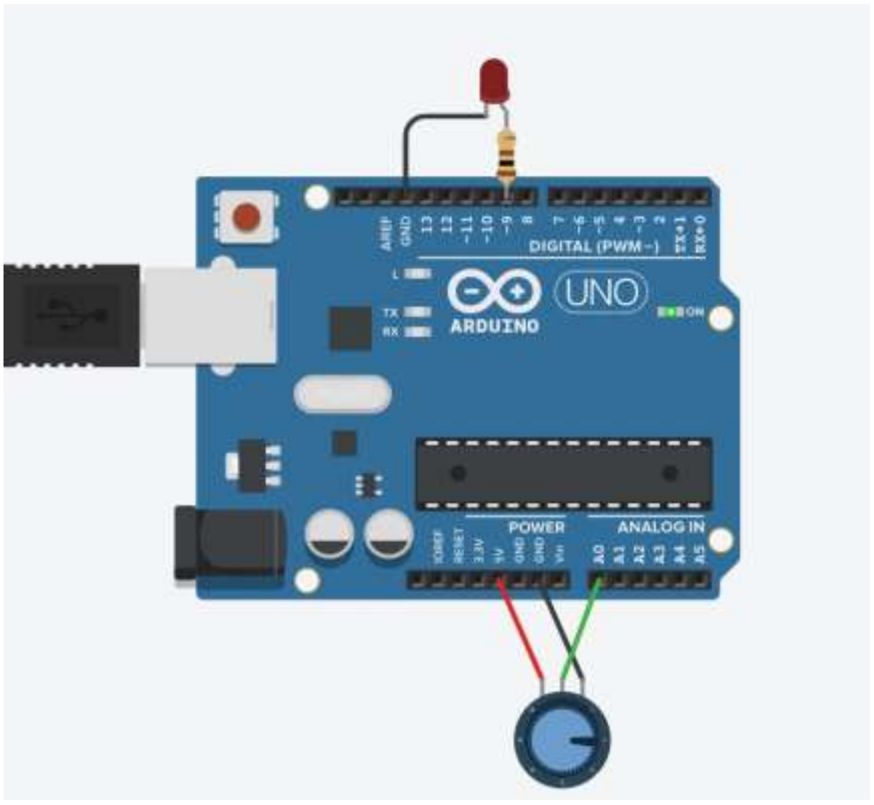
Program:

```
int sensorValue = 0;
void setup() {
pinMode(A0, INPUT);
pinMode(9, OUTPUT);
Serial.begin(9600);
}
void loop()
{
  sensorValue = analogRead(A0);
  if (sensorValue >= 500)
  digitalWrite(9,HIGH);
  else
    digitalWrite(9,LOW);
  delay(2); // Wait for 2 millisecond(s)
}
```

Output:

Name: Harshal Chavan Division: A
Roll No: 202124





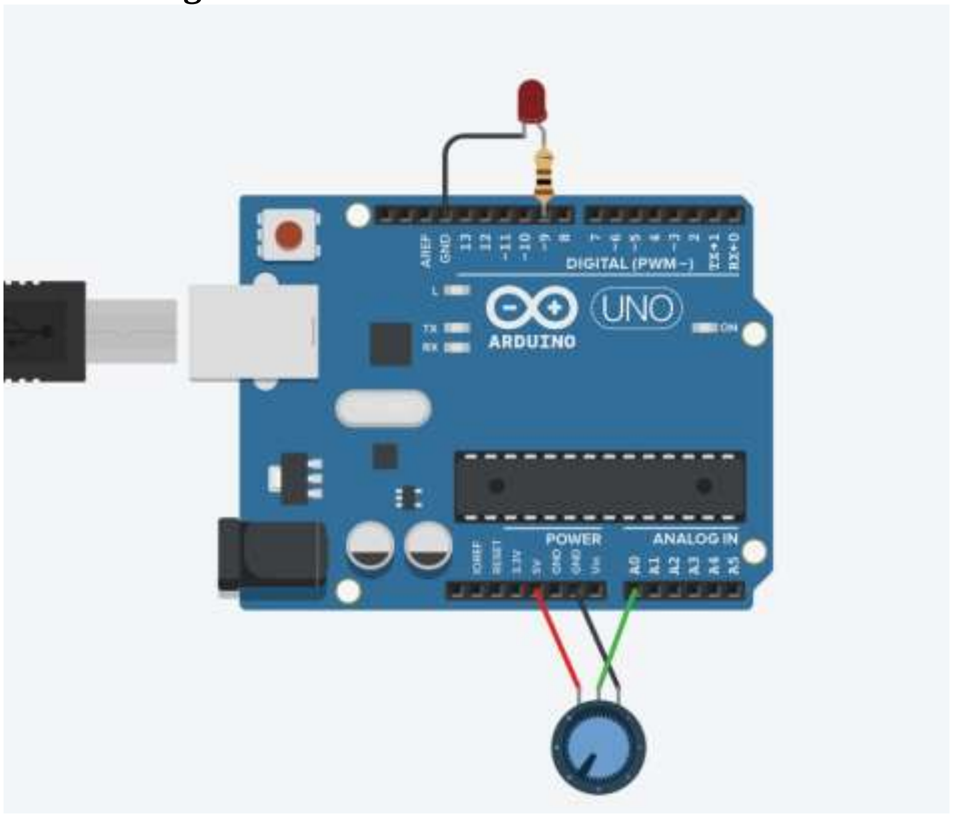
Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 12

Aim: To interface LED's, potentiometer with Arduino and write a program to implement all Analog signal functions.

Components: Arduino Uno, LCD 16 x 2, 100 Ω Potentiometer, 2pcs. 1 kΩ Resistor, Pushbutton

Circuit Diagram:

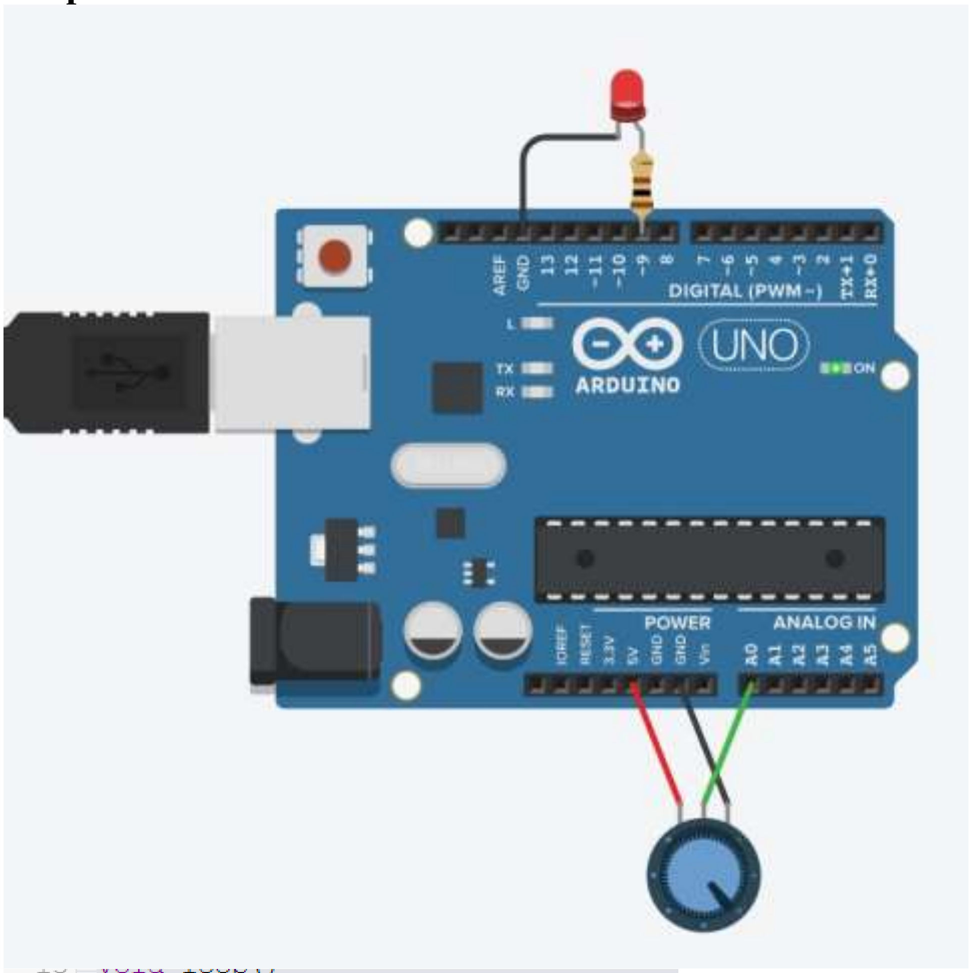


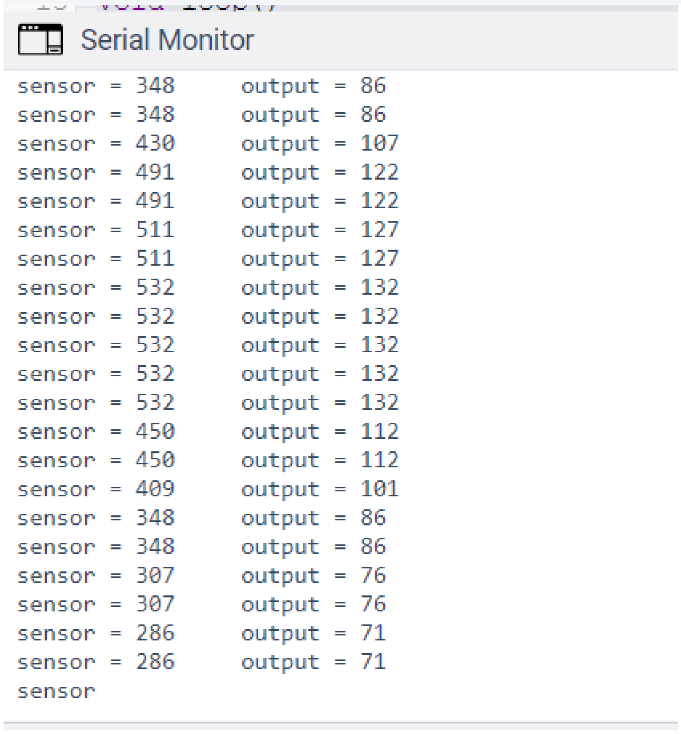
Program:

```
int sensorValue = 0;
int outputValue = 0;

void setup()
{
  pinMode(A0,  INPUT);
  pinMode(9, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  sensorValue = analogRead(A0);
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(9, outputValue);
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);
  delay(2);
```

}

Output:




Serial Monitor

```
sensor = 348     output = 86
sensor = 348     output = 86
sensor = 430     output = 107
sensor = 491     output = 122
sensor = 491     output = 122
sensor = 511     output = 127
sensor = 511     output = 127
sensor = 532     output = 132
sensor = 532     output = 132
sensor = 532     output = 132
sensor = 532     output = 132
sensor = 532     output = 132
sensor = 450     output = 112
sensor = 450     output = 112
sensor = 409     output = 101
sensor = 348     output = 86
sensor = 348     output = 86
sensor = 307     output = 76
sensor = 307     output = 76
sensor = 286     output = 71
sensor = 286     output = 71
sensor
```

Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 13

Aim: To interface LED, Photo resistor (LDR) with Arduino and write a program to increase and decrease the brightness of the LED based on the amount of light present.

Components: Photoresistor, Arduino Uno, Red LED, 1 kΩ Resistor, 100 Ω Resistor.

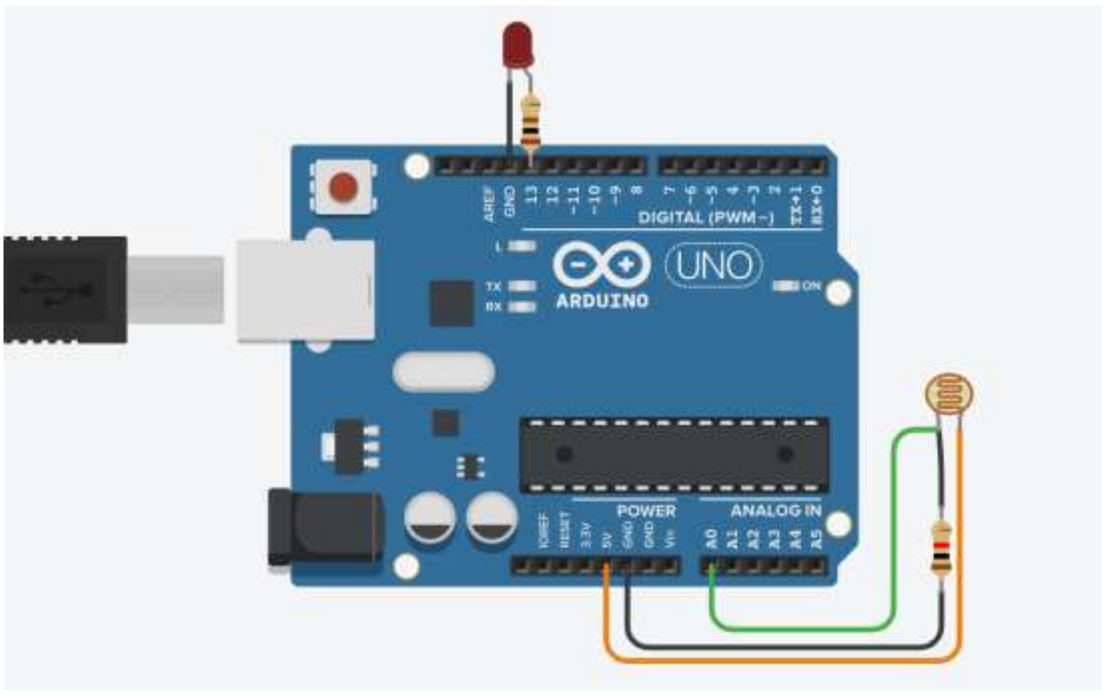
Theory:

A photoresistor (also known as a light-dependent resistor, LDR, or photo-conductive cell) is a passive component that decreases resistance with respect to receiving luminosity (light) on the component's sensitive surface. The resistance of a photoresistor decreases with increase in incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits and light-activated and dark-activated switching circuits acting as a resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several megaohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a photoresistor can substantially differ among dissimilar devices. Moreover, unique photoresistors may react substantially differently to photons within certain wavelength bands.

A photoelectric device can be either intrinsic or extrinsic. An intrinsic semiconductor has its own charge carriers and is not an efficient semiconductor, for example, silicon. In intrinsic devices, the only available electrons are in the valence band, and hence the photon must have enough energy to excite the electron across the entire bandgap. Extrinsic devices have impurities, also called dopants, added whose ground state energy is closer to the conduction band; since the electrons do not have as far to jump, lower energy photons (that is, longer wavelengths and lower frequencies) are sufficient to trigger the device. If a sample of silicon has some of its atoms replaced by phosphorus atoms (impurities), there will be extra electrons available for conduction. This is an example of an extrinsic semiconductor.

Circuit Diagram:

Name: Harshal Chavan Division: A
Roll No: 202124

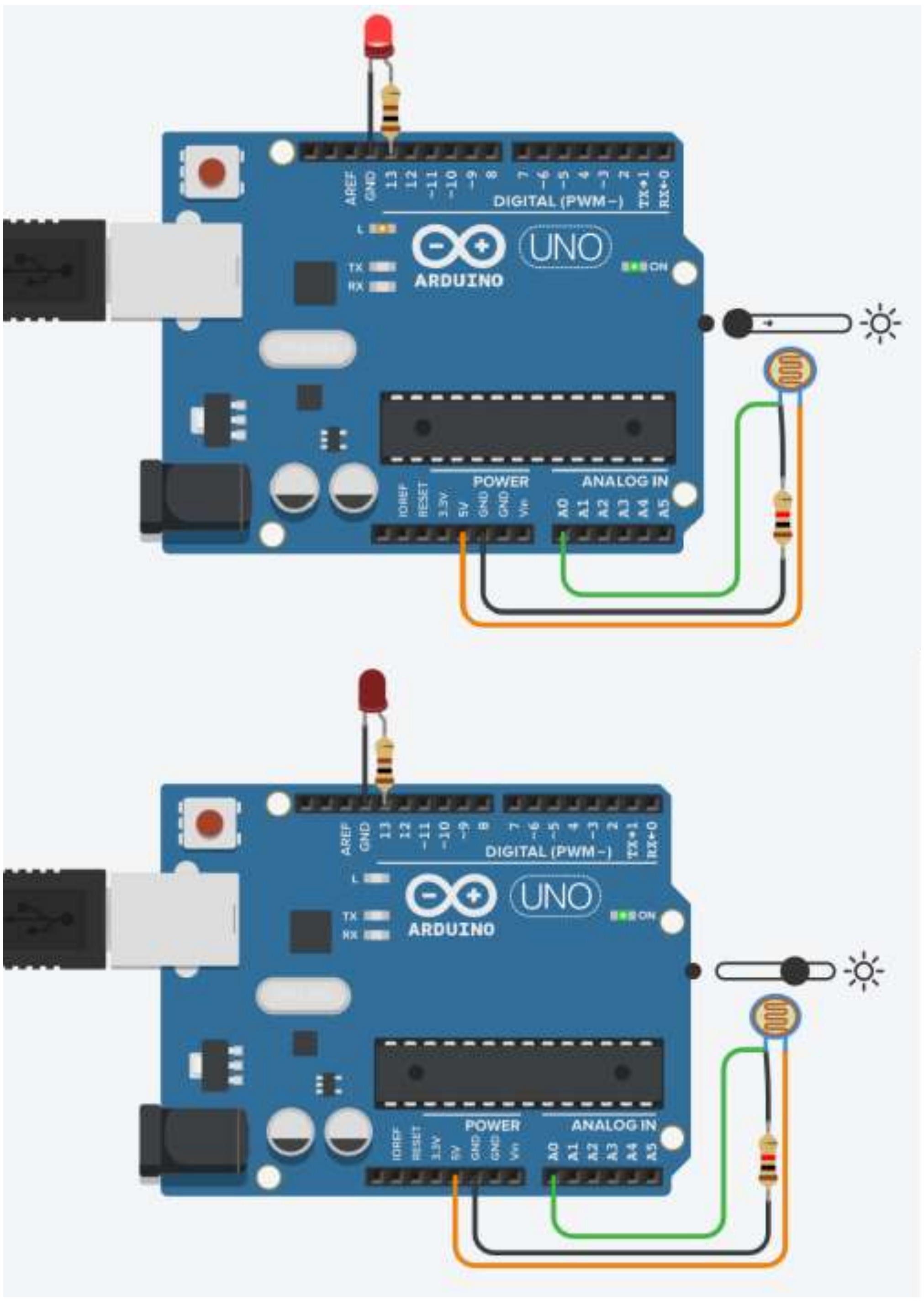


Program:

```
int ldrPin = A0;
int ledPin = 13;
int sensorValue = 0;
void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  sensorValue = analogRead(ldrPin);
  Serial.println(sensorValue);

  if(sensorValue <= 500)
  digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}
```

Output:

Name: Harshal Chavan   Division: A
Roll No: 202124



Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 14

Aim: To interface LED's with Arduino and write a program to show the fading effect on LED's.

Component: Arduino Uno, 0.2 kΩ Resistor, Red LED

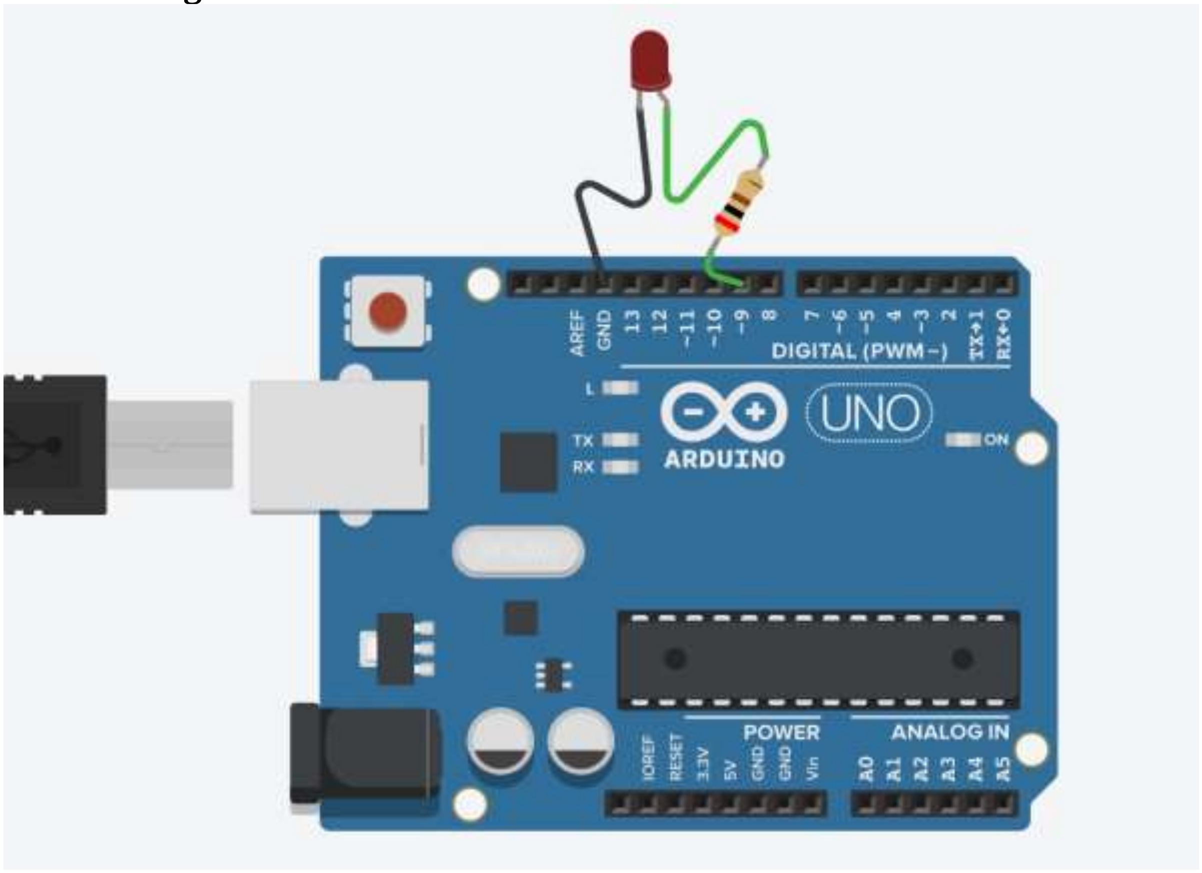
Theory:

In order to fade your LED off and on, gradually increase your PWM value from 0 (all the way off) to 255 (all the way on), and then back to 0 once again to complete the cycle. In the sketch below, the PWM value is set using a variable called brightness. Each time through the loop, it increases by the value of the variable fadeAmount.

If brightness is at either extreme of its value (either 0 or 255), then fadeAmount is changed to its negative. In other words, if fadeAmount is 5, then it is set to -5. If it's -5, then it's set to 5. The next time through the loop, this change causes brightness to change direction as well.

analogWrite() can change the PWM value very fast, so the delay at the end of the sketch controls the speed of the fade. Try changing the value of the delay and see how it changes the fading effect.

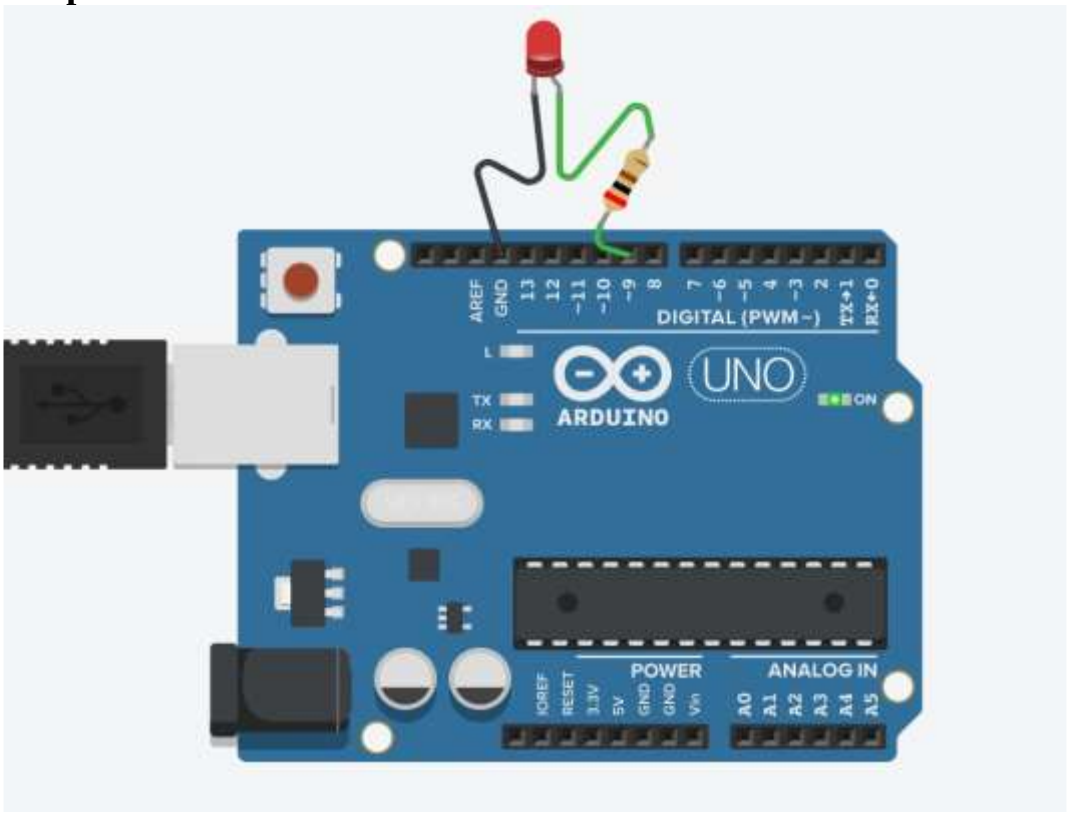
Circuit Diagram:



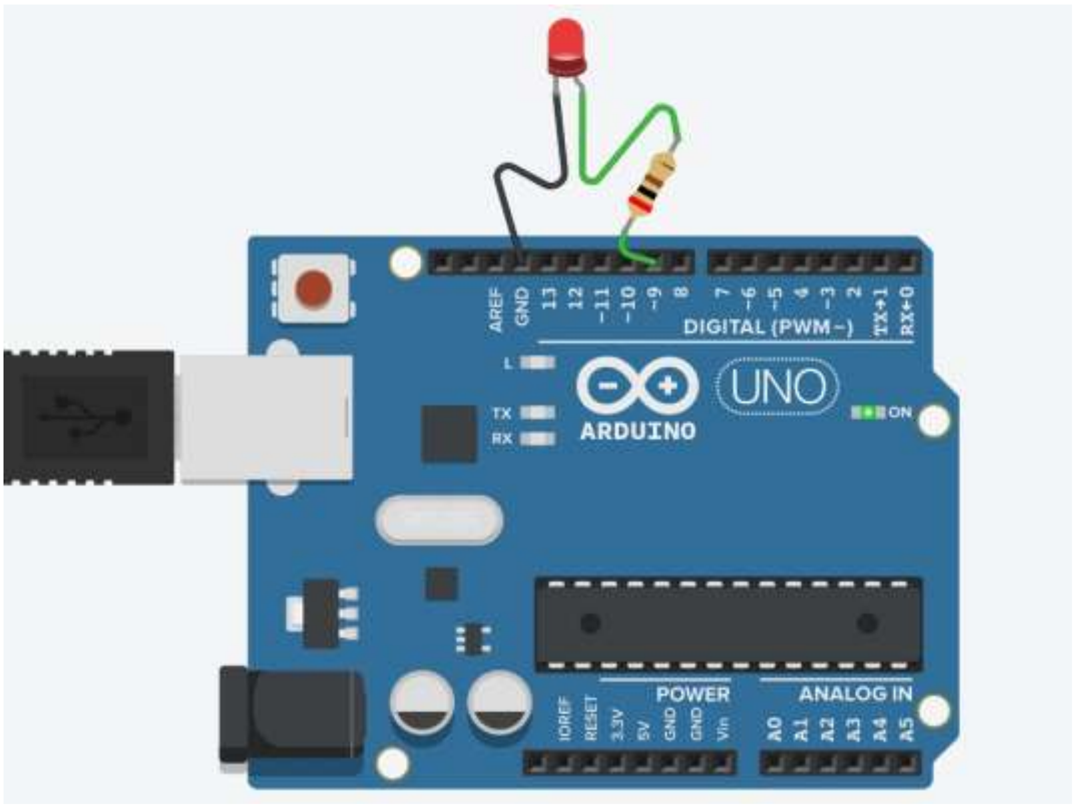
Program:

```
int led = 9;
int bright = 255;
int fed = 5;
void setup()
{
  pinMode(9, OUTPUT);
}

void loop()
{
  analogWrite(led,bright);
  bright = bright + fed;
  delay(30);
  if(bright <= 0){
    for (int i=0;i<=255;i=i+fed)
    {
      bright = i;
      analogWrite(led,bright);
      delay(30);
    }
  }
  else if(bright >= 255){
    for (int i=255;i >=0;i=i-fed)
    {
      bright = i;
      analogWrite(led,bright);
      delay(30);
    }
  }}
```

Output:



Name: Harshal Chavan Division: A
Roll No: 202124



Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 15

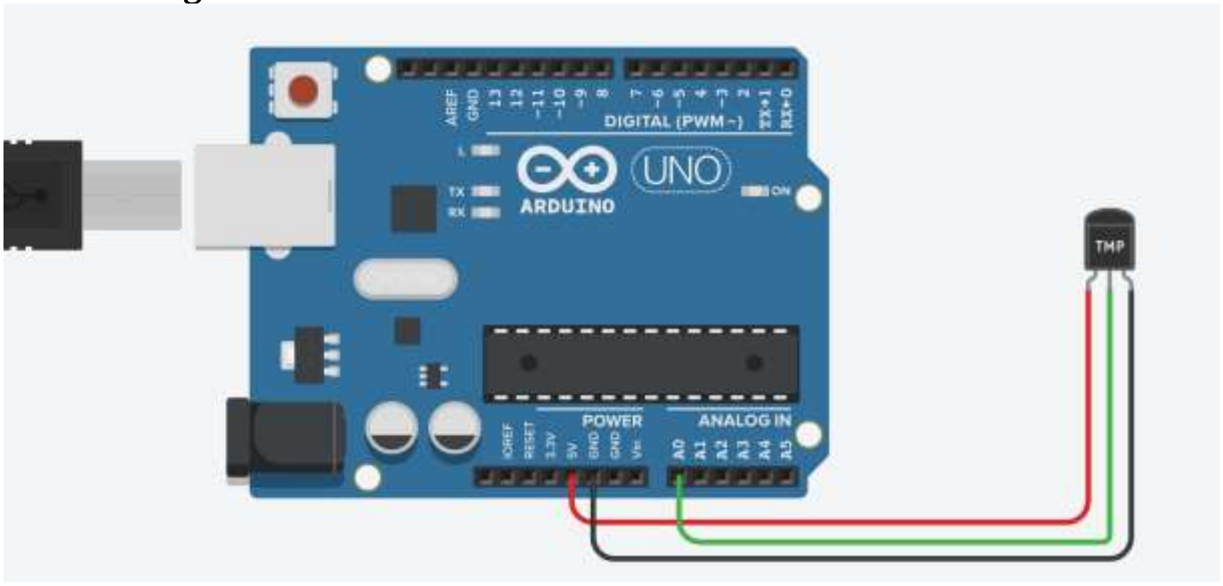
Aim: To interface LM35 sensor with Arduino and write a program to display temperature data on serial monitor.

Component: Arduino Uno, Temperature Sensor [TMP36]

Theory:

TMP36 is a temperature sensor chip which generates an analog voltage at the output which is linearly proportional to the Celsius temperature. Then convert this voltage into temperature based on a 10 mV/°C scale factor. It has a shutdown capability which limits the output current to less than 0.5 µA. It provides a supply current of up to 50 µA.
This sensor provides a highly precise temperature in centigrade. Most importantly, it produces output in dc voltage that we can measure easily with the help of any bare metal microcontrollers such as Arduino Uno, STM32F4, PIC16F877A. On top of that, Celsius's temperature and an output voltage change linearly which makes it easy to compensate temperature/Voltage variations. Having a linear relationship is helpful. Because we will not require any external calibration circuit. Furthermore, it offers a very low output impedance. In short, it is very easy to interface this sensor with ADCs or microcontrollers having built-in ADCs.

Circuit Diagram:

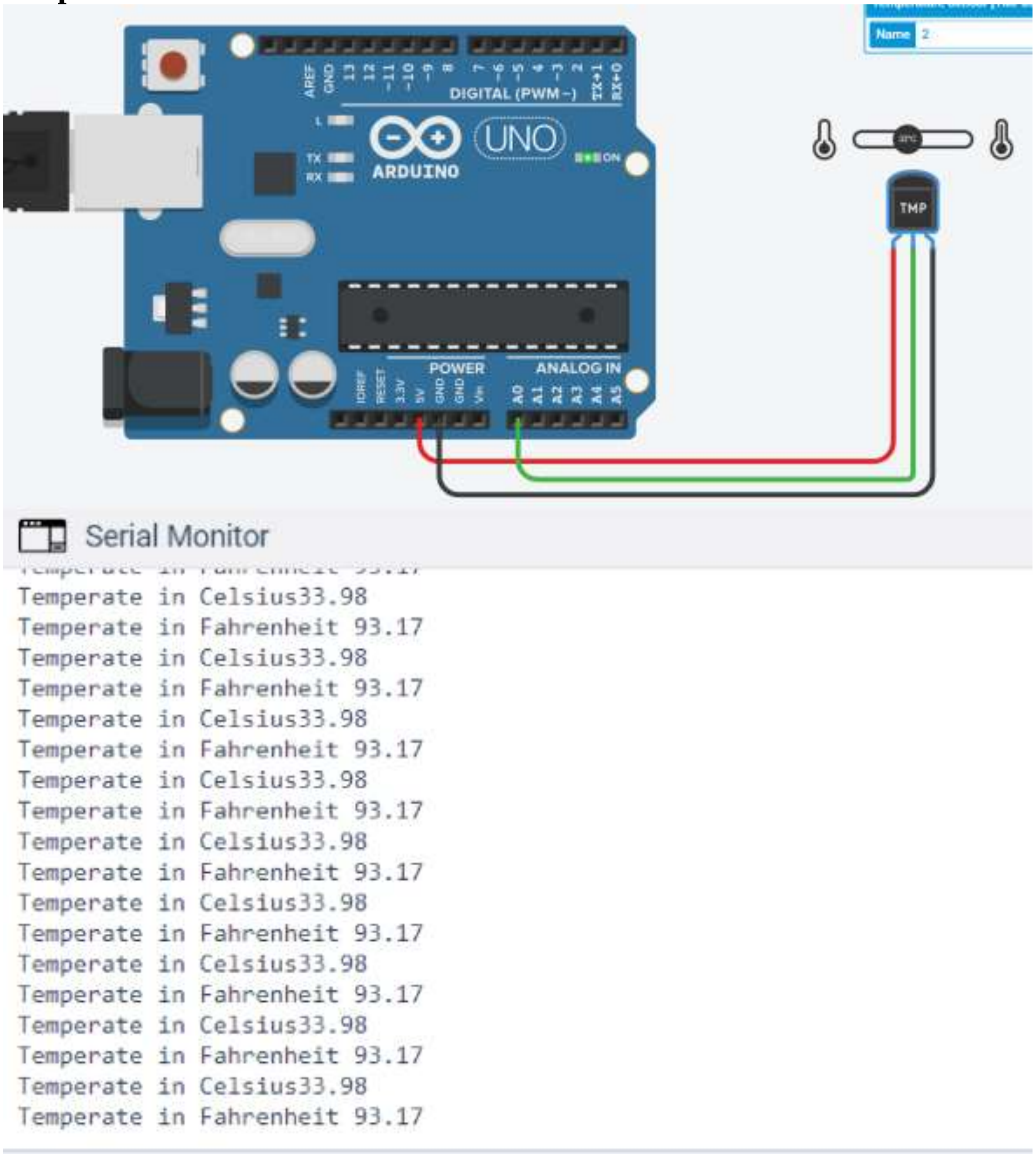


Program:

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  int sensorValue = analogRead(A0);
  float volt = (5.0 / 1024) * sensorValue;
  delay(2000);

  float tempC = (volt - 0.5) * 100;
  Serial.print("Temperate in Celsius");
  Serial.print(tempC);

  float tempF = (tempC * 9/5) + 32;
  Serial.print("\nTemperate in Fahrenheit ");
  Serial.println(tempF);
}
```

Name: Harshal Chavan   Division: A
Roll No: 202124

Output:



```
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
Temperate in Celsius33.98
Temperate in Fahrenheit 93.17
```

Name: Harshal Chavan Division: A
Roll No: 202124

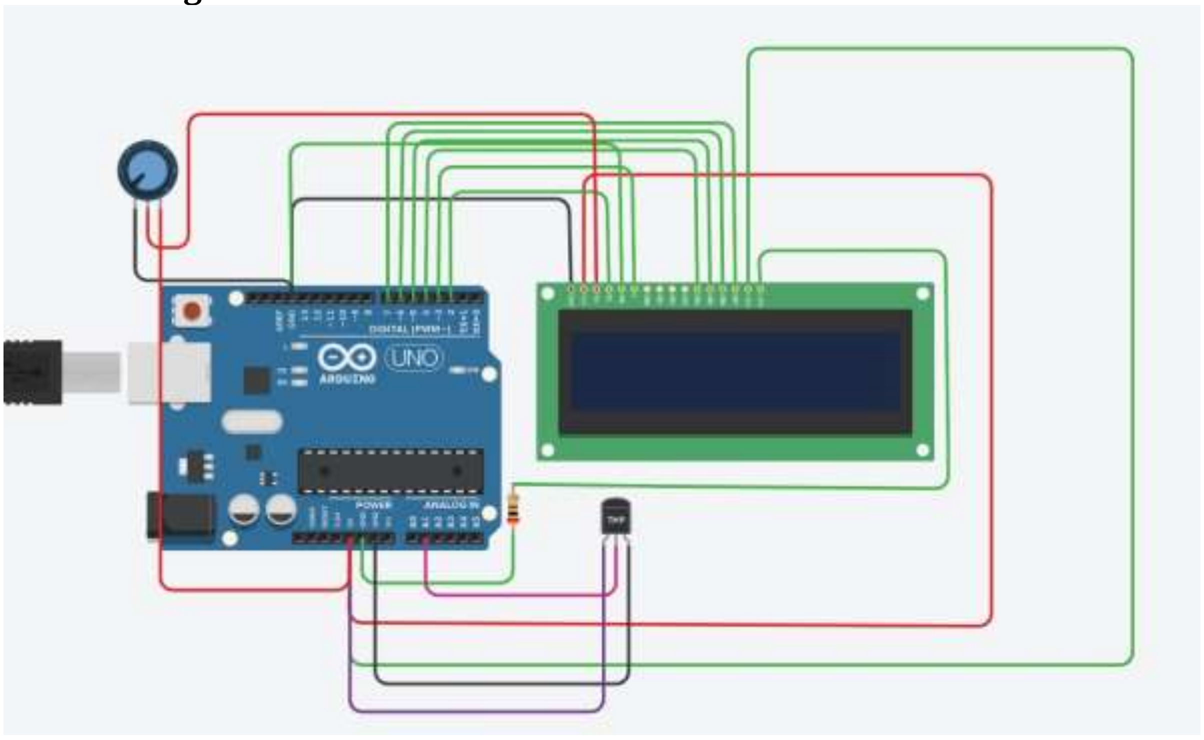
Practical No. 16

Aim: To interface LM35 sensor with Arduino and write a program to display temperature data on LCD.

Component: LCD 16 x 2, 0.2 kΩ Resistor, Temperature Sensor [TMP36], 100 Ω Potentiometer

Theory:

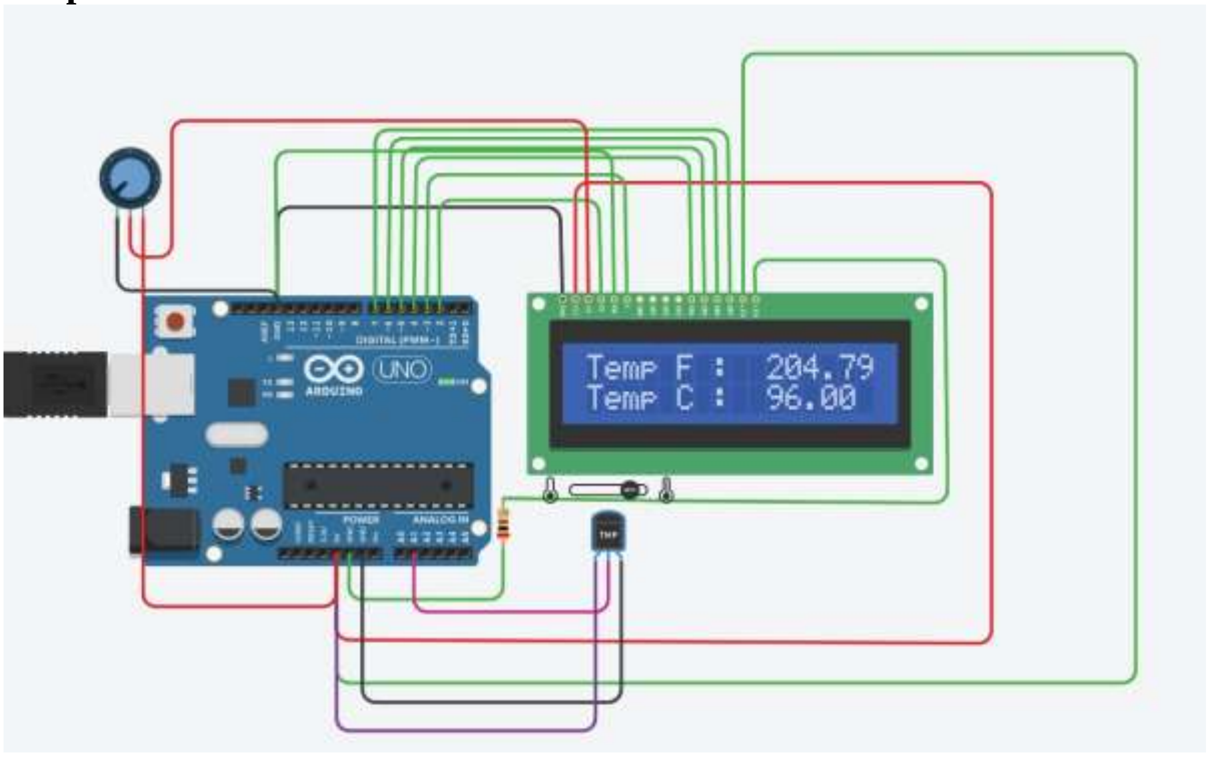
Circuit Diagram:



Program:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(2,3,4,5,6,7);
void setup()
{
  pinMode(A1,INPUT);
  lcd.begin(16,2);
  lcd.print("Temp F :");
}

void loop()
{
  lcd.setCursor(10,0);
  int sensorValue = analogRead(A1);
  float volts = (5.0/1024) * sensorValue;
  float tempC = (volts - 0.5) * 100;
  float tempF = (tempC * 9/5 ) + 32;
  lcd.print(tempF);
  lcd.setCursor(0,1);
  lcd.print("Temp C :");
  lcd.setCursor(10,1);
  lcd.print(tempC);

}
```

Output:



Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 17

Aim: To interface PIR sensor with Arduino and write a program to turn on and off LED depending on motion detection.

Component: Arduino Uno, PIR Sensor, Red LED, 100 Ω Resistor

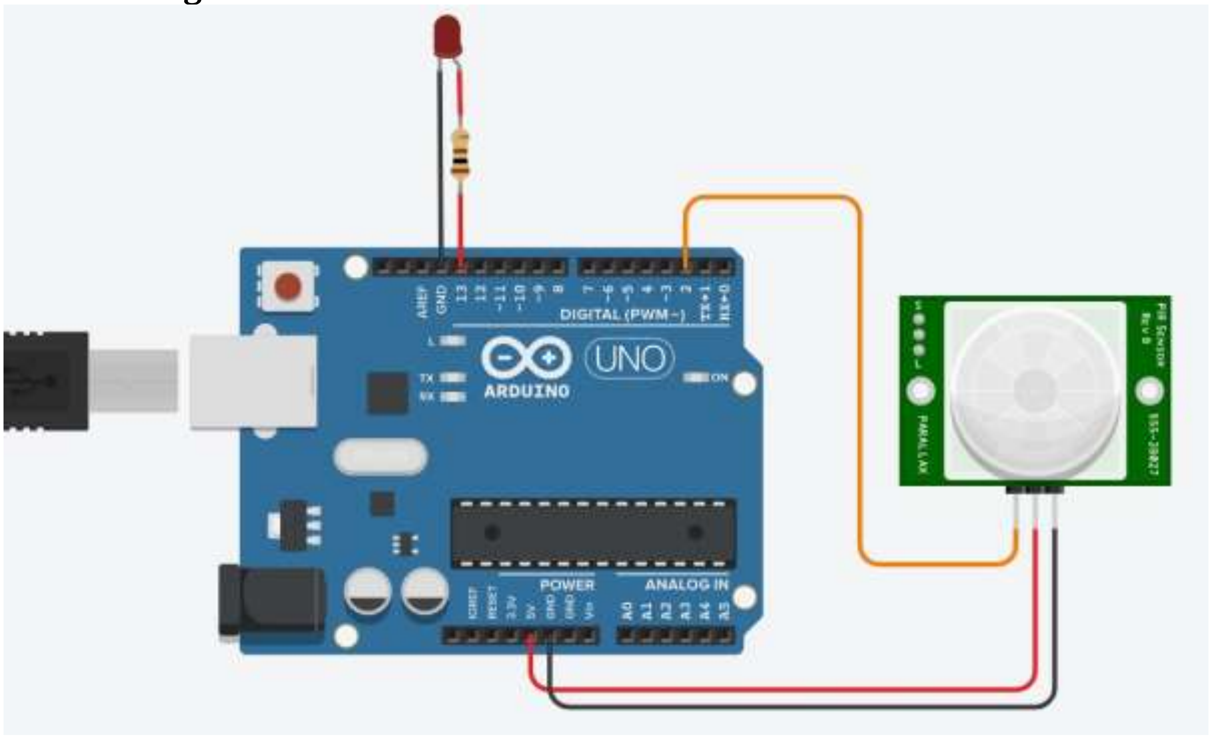
Theory:

PIR sensors are more complicated than many of the other sensors explained in these tutorials (like photocells, FSRs and tilt switches) because there are multiple variables that affect the sensors input and output. To begin explaining how a basic sensor works, we'll use this rather nice diagram

The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

Name: Harshal Chavan Division: A
Roll No: 202124

Circuit Diagram:

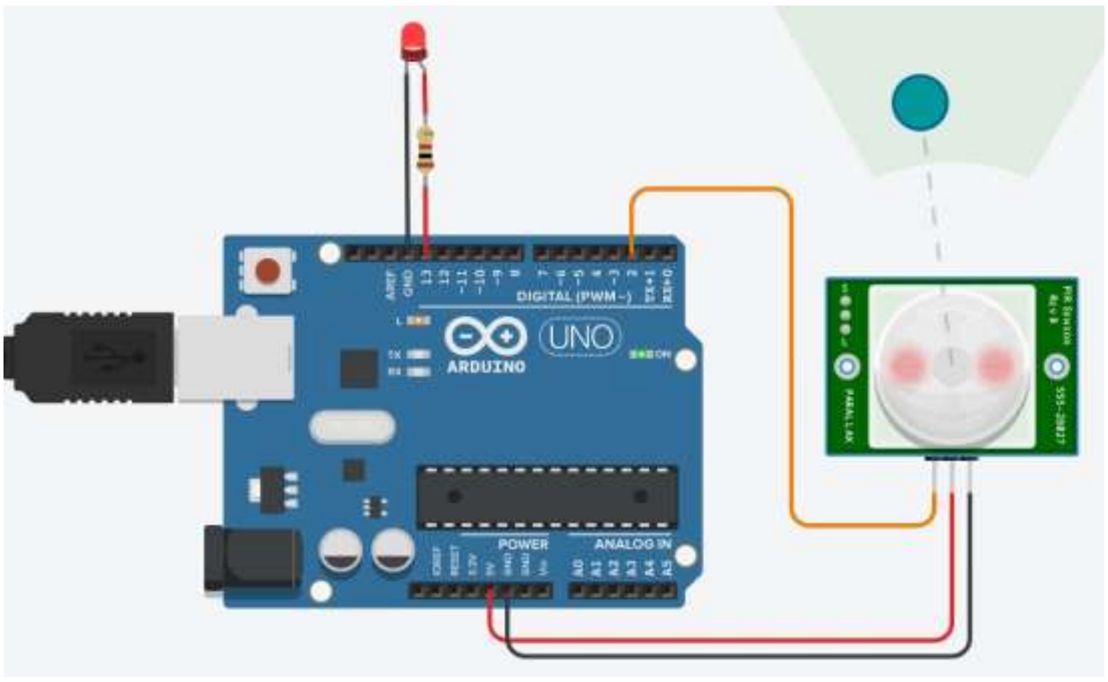


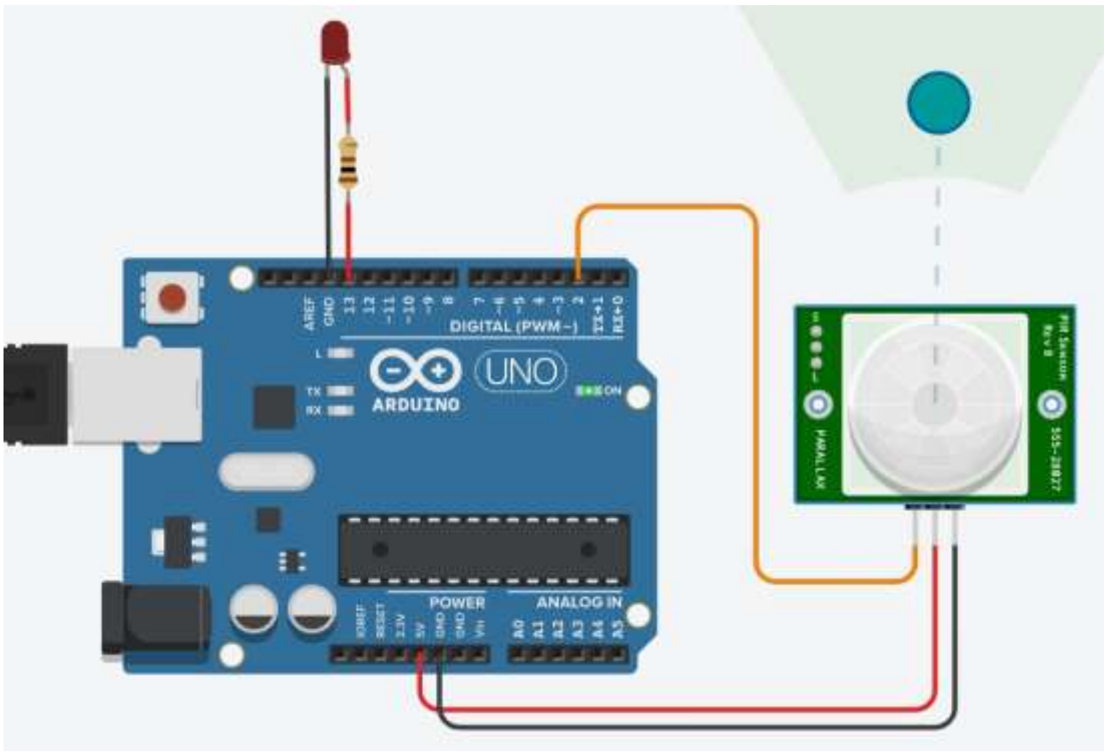
Program:

```
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(2, INPUT);
}
void loop()
{
  int sensorMotion = digitalRead(2);
  if (sensorMotion == HIGH)
  digitalWrite(13, HIGH);
  else
    digitalWrite(13, LOW);
}
```

Output:

Name: Harshal Chavan Division: A
Roll No: 202124





Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 18

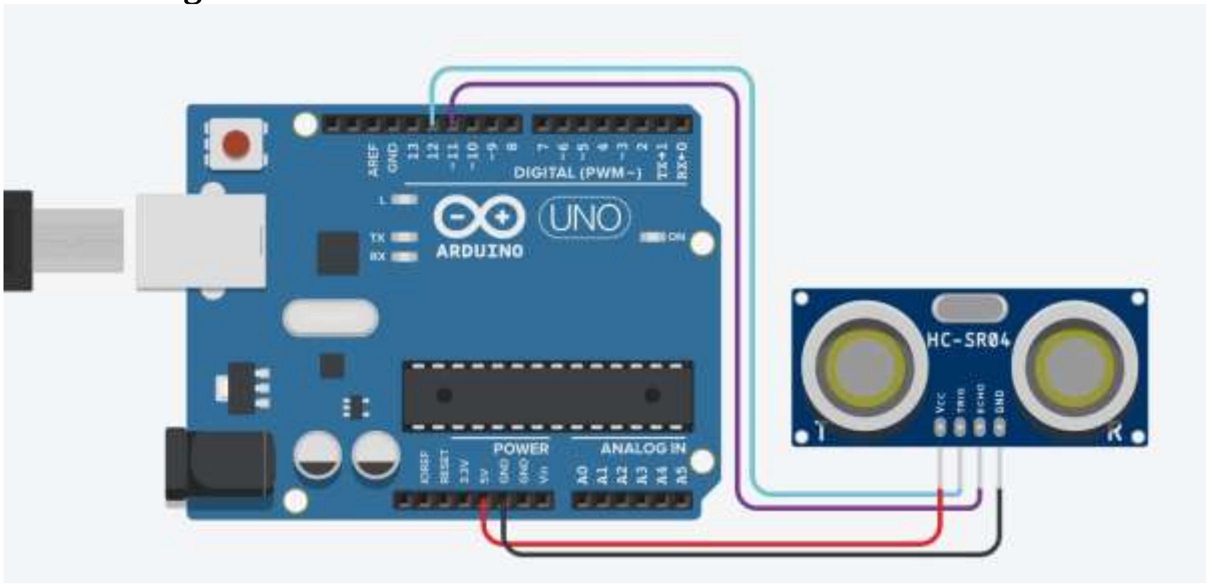
Aim: To interface Ultrasonic sensor with Arduino and write a program to display object distance(in inch/cm) in serial monitor depending on sound detection.

Component: Arduino Uno, Ultrasonic Distance Sensor

Theory:

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves.
An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity.
High-frequency sound waves reflect from boundaries to produce distinct echo patterns
Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.

Circuit Diagram:

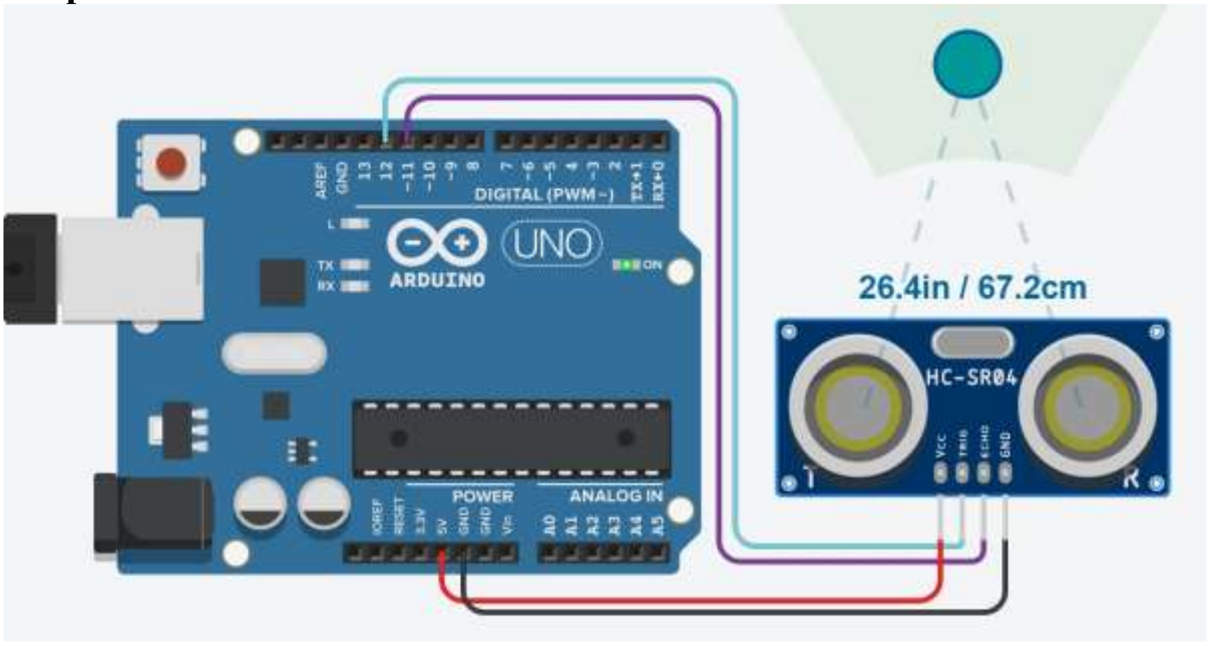


Program:

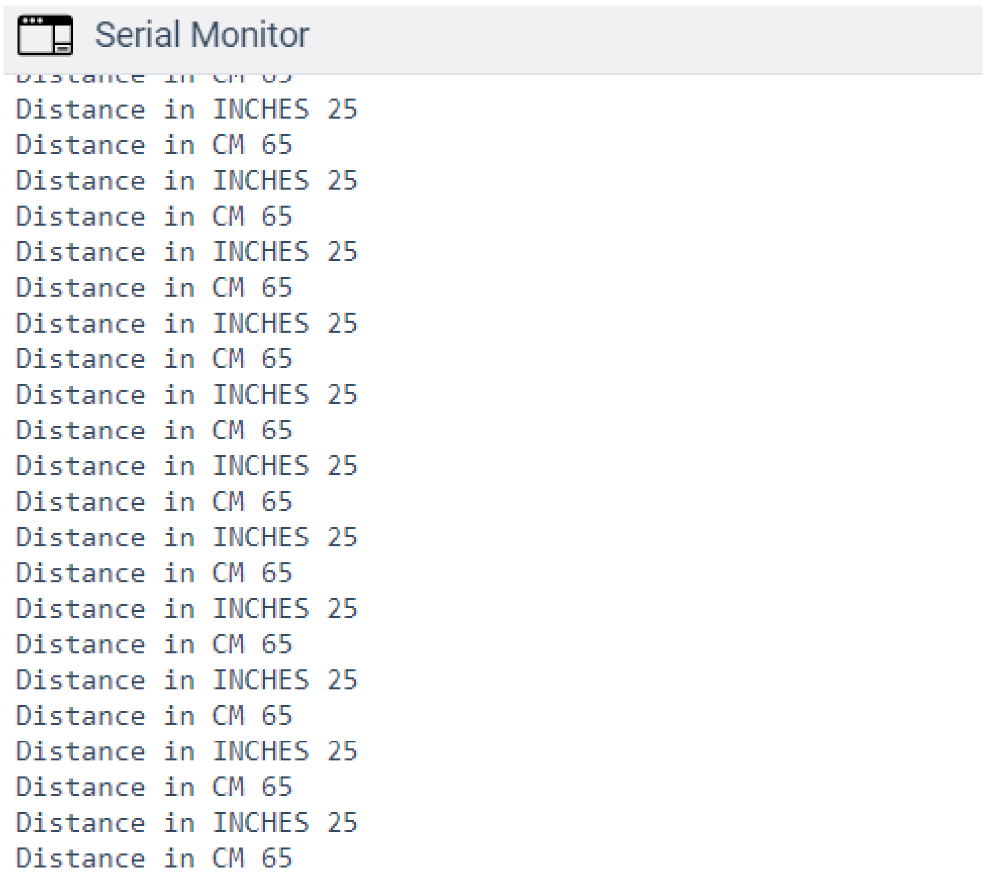
```
int trig = 12;
int echo = 11;
int travelTime;
int distance;
int distanceInch;

void setup()
```

```
{
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(9600);
}
void loop()
{
  digitalWrite(trig , LOW);
  delayMicroseconds(10);
  digitalWrite(trig , HIGH);
  delay(10);
  digitalWrite(trig , LOW);
  travelTime = pulseIn(echo,HIGH);
  delay(20);
  distance = (travelTime / 2) * 0.0343;
  distanceInch = ( travelTime / 2)* 0.013464 ;
  Serial.print("Distance in CM ");
  Serial.println(distance);
  Serial.print("Distance in INCHES ");
  Serial.println(distanceInch);
}
```

Output:



Name: Harshal Chavan Division: A
Roll No: 202124



```
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
Distance in INCHES 25
Distance in CM 65
```

Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 19

Aim: To interface servo motor with Arduino and write a program to sweep a servo back and forth through its full range of motion.

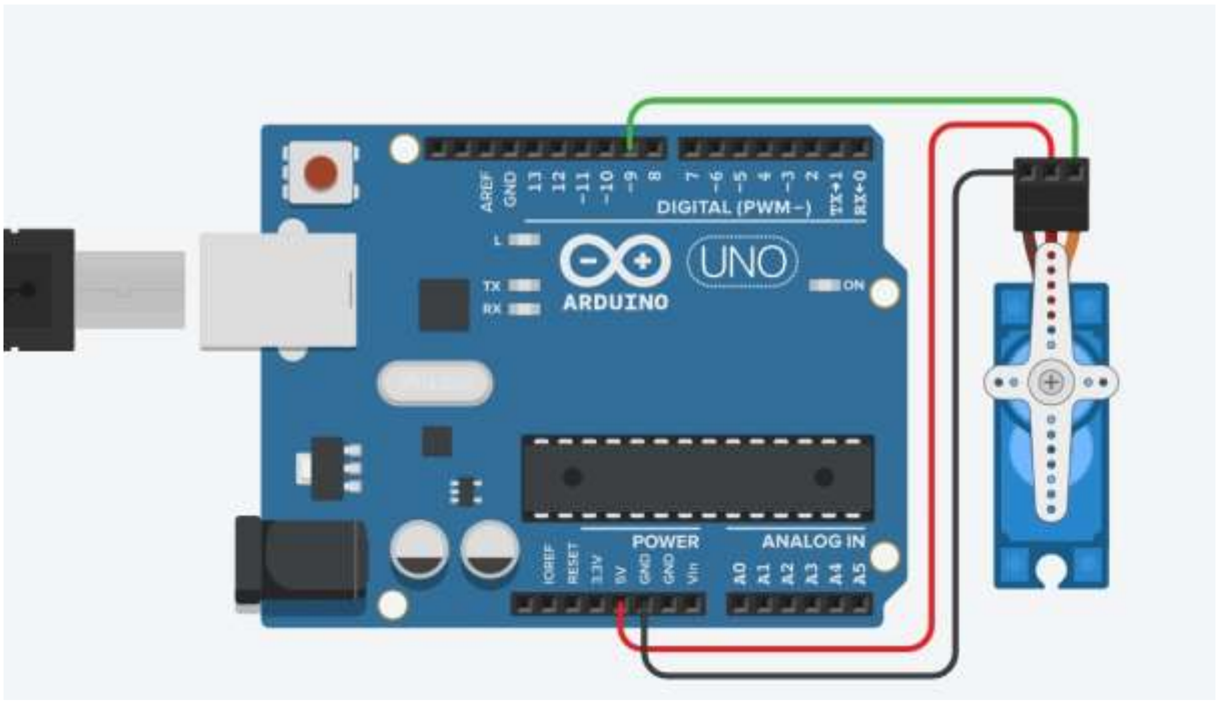
Component: Arduino Uno, Positional Micro Servo

Theory:

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration.[1] It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.
Servomotors are not a specific class of motor, although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system.
Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.
A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.
The motor is paired with some type of position encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

Circuit Diagram:

Name: Harshal Chavan Division: A
Roll No: 202124

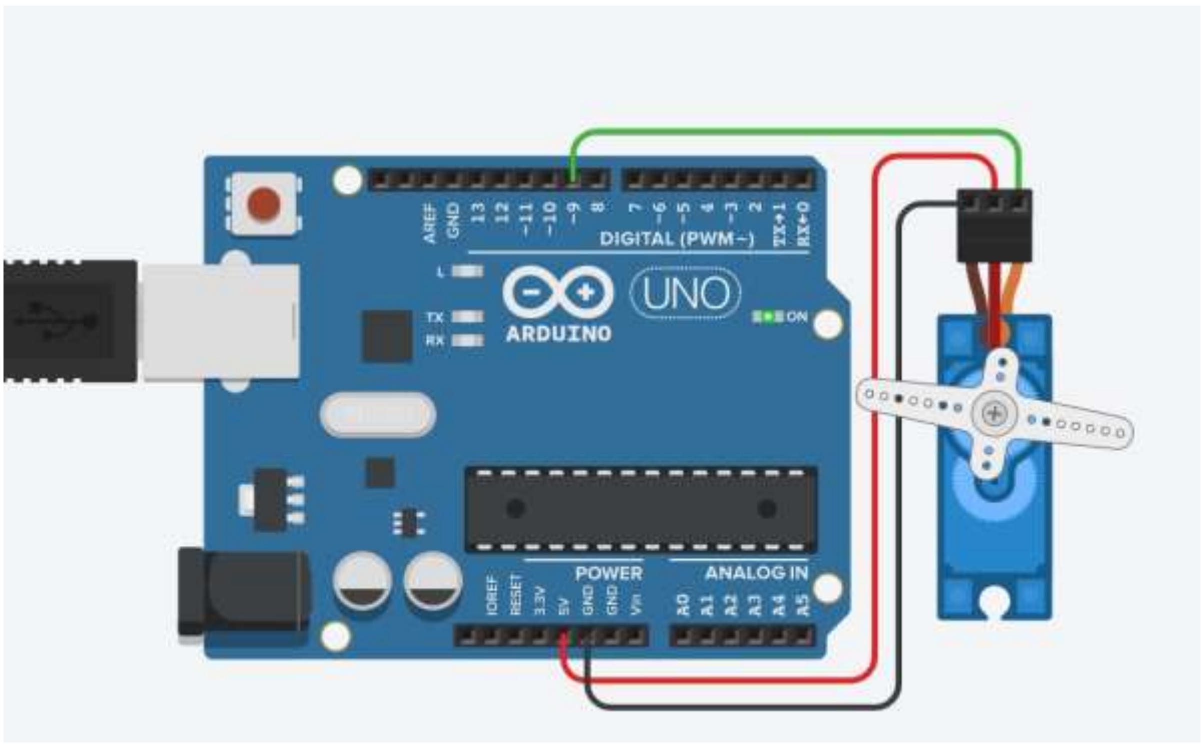


Program:

```
#include <Servo.h>

int servoPin = 9;
int servoPos = 45;
int servoInitalPos = 180;
Servo myServo;

void setup()
{
  myServo.attach(servoPin);
}
void loop()
{
 int pos = 0;
 int dtwait=15;

 for(pos = 0; pos < 180; pos += 1) {
 myServo.write(pos);
 delay(dtwait);
 }
 for(pos = 180; pos>=1; pos -= 1) {
 myServo.write(pos);
 delay(dtwait);
 }
}
```

Output:

Name: Harshal Chavan Division: A
Roll No: 202124

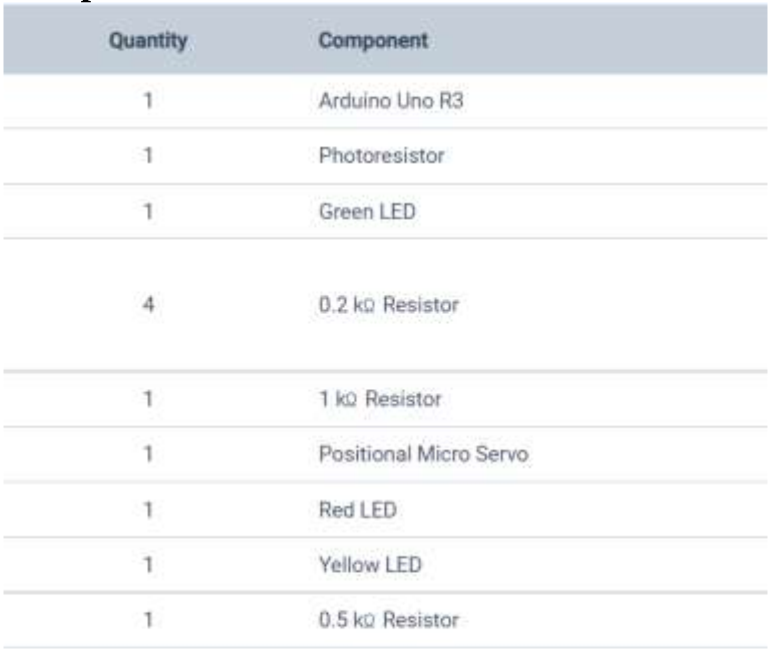


Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 20

Aim: To interface servo motor, Photo resistor (LDR) with Arduino and write a program to sweep a servo back and forth through its full range of motion depending on light.

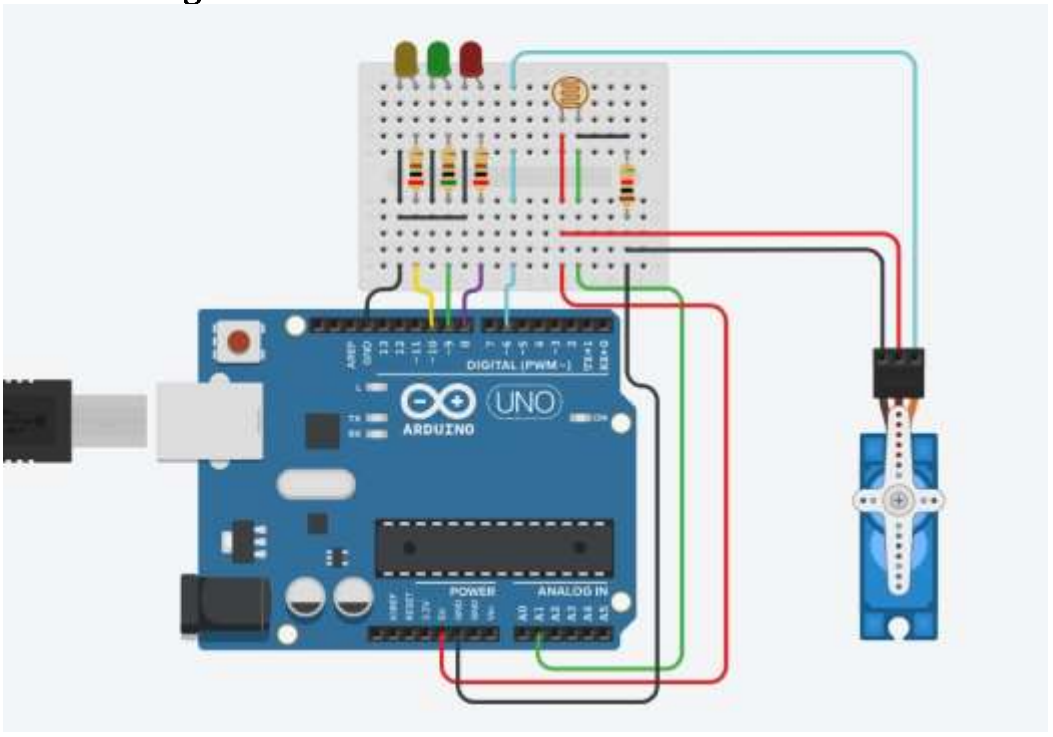
Component:



Quantity	Component
1	Arduino Uno R3
1	Photoresistor
1	Green LED
4	0.2 kΩ Resistor
1	1 kΩ Resistor
1	Positional Micro Servo
1	Red LED
1	Yellow LED
1	0.5 kΩ Resistor

Theory:

Circuit Diagram:



Program:

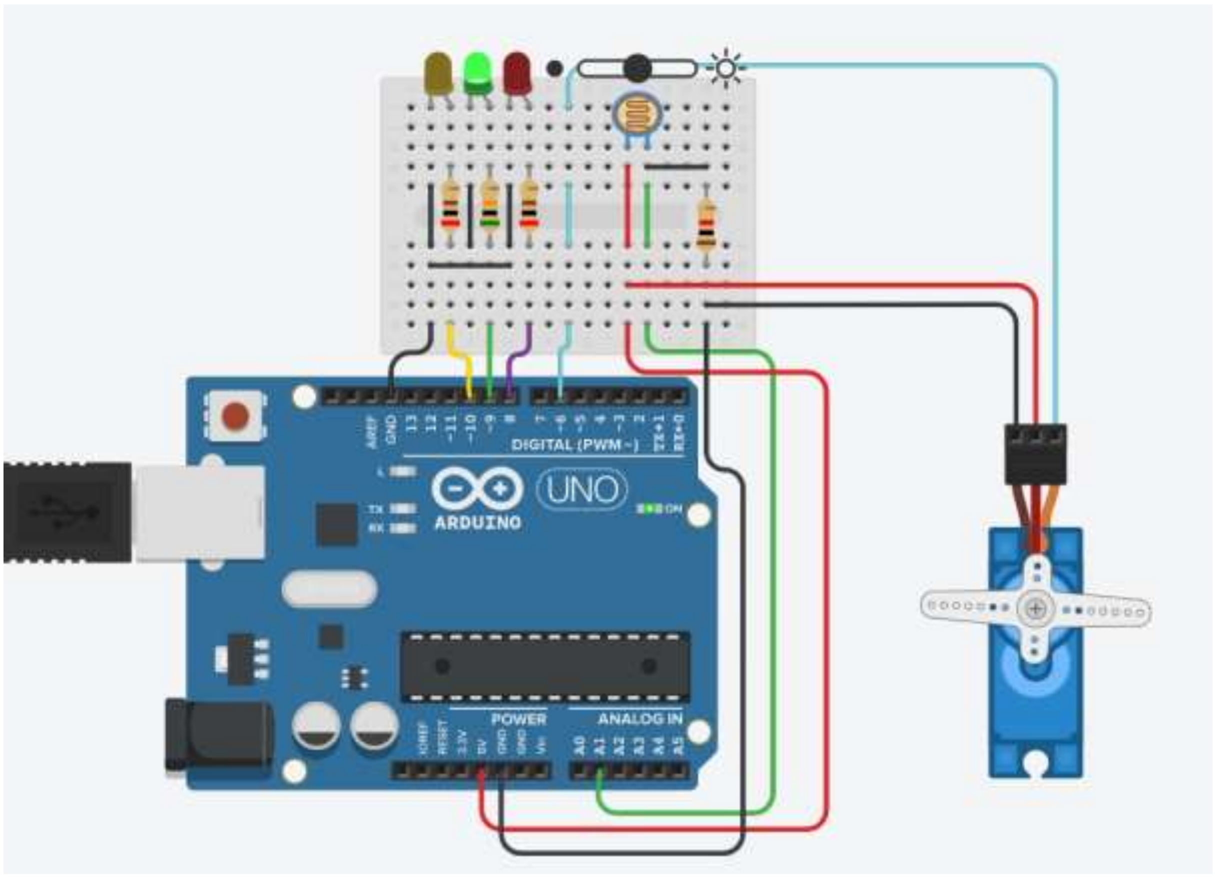
```
#include <Servo.h>

int servoPin = 6;
int servoPos = 0;
```

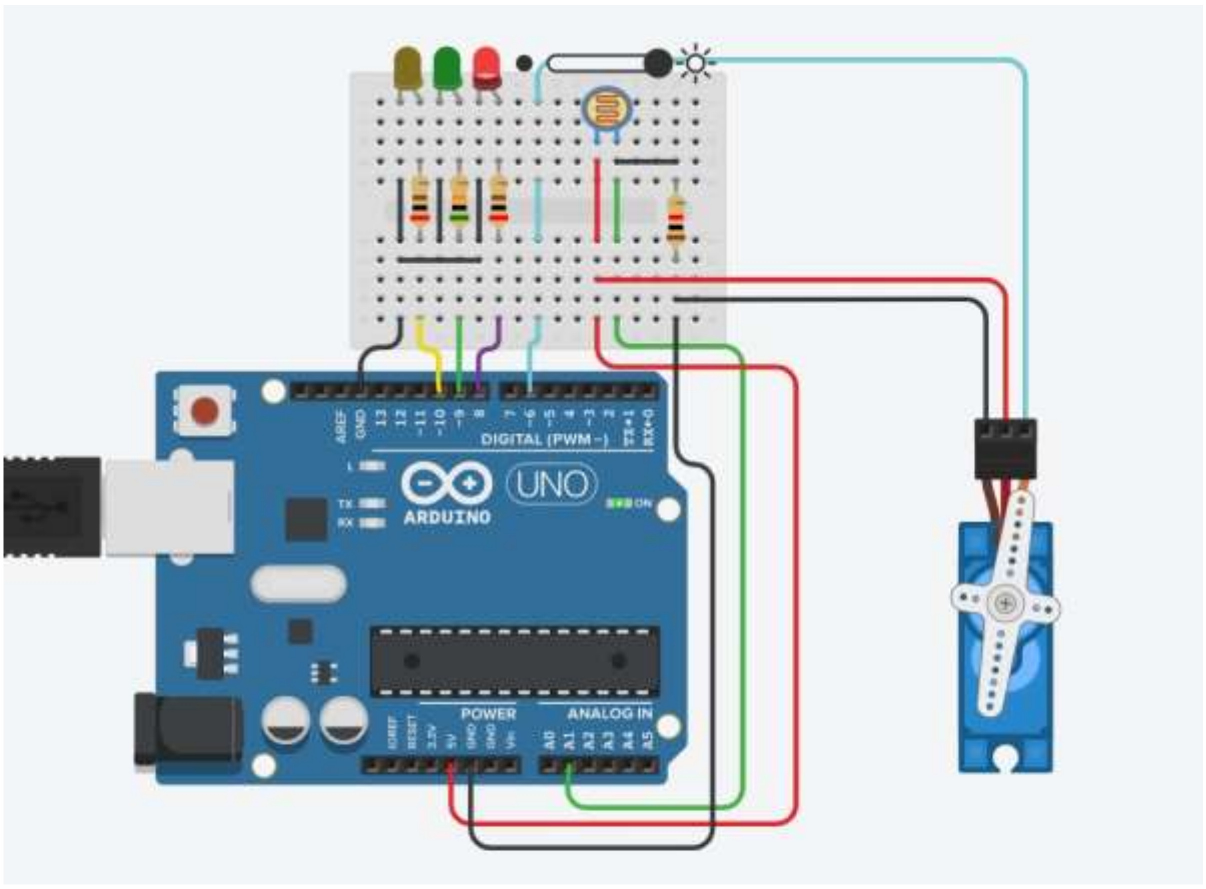
Name: Harshal Chavan Division: A
Roll No: 202124

```
int red = 8;
int green = 9;
int yellow = 10;
Servo myServo;

void setup()
{
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(yellow, OUTPUT);
  myServo.attach(servoPin);
  Serial.begin(9600);
}

void loop()
{
  Serial.println(analogRead(A1));
  Serial.println(analogRead(A1));
  if (analogRead(A1) <= 700 && analogRead(A1) >595){
    digitalWrite(red, HIGH);
    digitalWrite(green, LOW);
    digitalWrite(yellow, LOW);
    myServo.write(180);
  }
  else if (analogRead(A1) <= 595 && analogRead(A1) > 200){
    digitalWrite(red, LOW);
    digitalWrite(green,  HIGH);
    digitalWrite(yellow, LOW);
    myServo.write(90);
  }
  else{
    digitalWrite(red, LOW);
    digitalWrite(green, LOW);
    digitalWrite(yellow, HIGH);
    myServo.write(0);
  }

}
```

Output:

Name: Harshal Chavan Division: A
Roll No: 202124



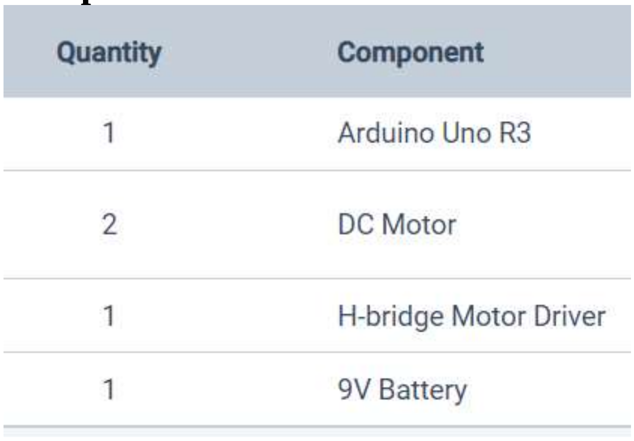


Name: Harshal Chavan   Division: A
Roll No: 202124

Practical No. 21

Aim: Write a program to control two DC motors with Arduino.

Component:



Quantity	Component
1	Arduino Uno R3
2	DC Motor
1	H-bridge Motor Driver
1	9V Battery

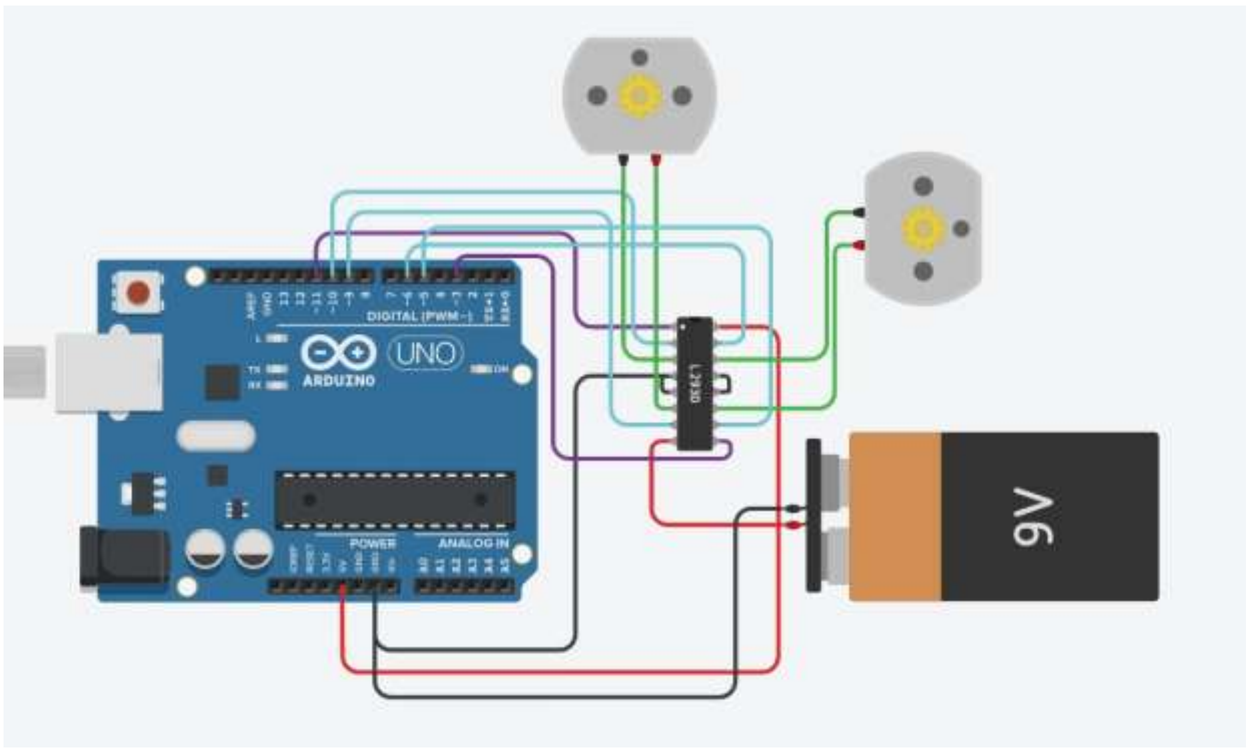
Theory:

DC Motor:

A direct current (DC) motor is a type of electric machine that converts electrical energy into mechanical energy. DC motors take electrical power through direct current, and convert this energy into mechanical rotation.
DC motors use magnetic fields that occur from the electrical currents generated, which powers the movement of a rotor fixed within the output shaft. The output torque and speed depend upon both the electrical input and the design of the motor.

Circuit Diagram:

Name: Harshal Chavan Division: A
Roll No: 202124

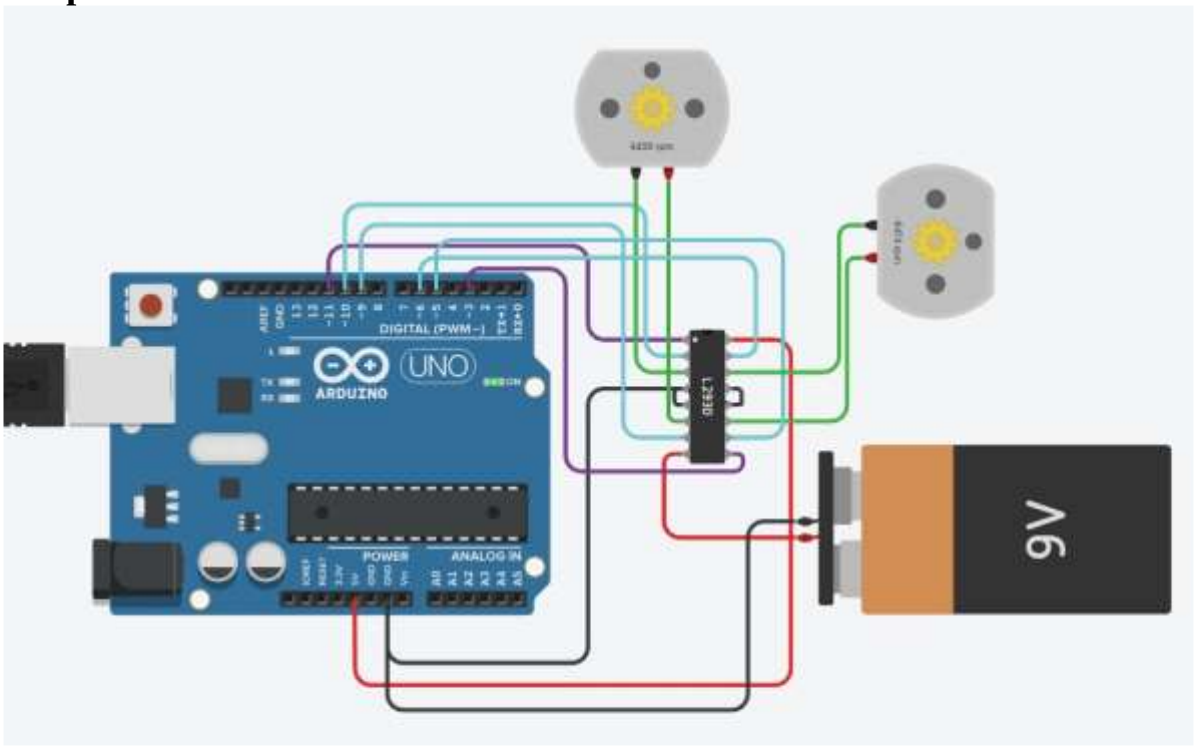


Program:

```
int speedPin = 11;
int speedPin2 = 3;
int dir1= 9;
int dir2= 10;
int dir3= 5;
int dir4= 6;

void setup()
{
  pinMode(speedPin, OUTPUT);
  pinMode(speedPin2, OUTPUT);
  pinMode(dir1, OUTPUT);
  pinMode(dir2, OUTPUT);
  pinMode(dir3, OUTPUT);
  pinMode(dir4, OUTPUT);

}

void loop()
{
  digitalWrite(dir1, HIGH);
  digitalWrite(dir2, LOW);
  digitalWrite(dir3, LOW);
  digitalWrite(dir4, HIGH);
  analogWrite(speedPin,100);
  analogWrite(speedPin2,100);
}
```

Name: Harshal Chavan Division: A
Roll No: 202124

Output:



Name: Harshal Chavan Division: A
Roll No: 202124

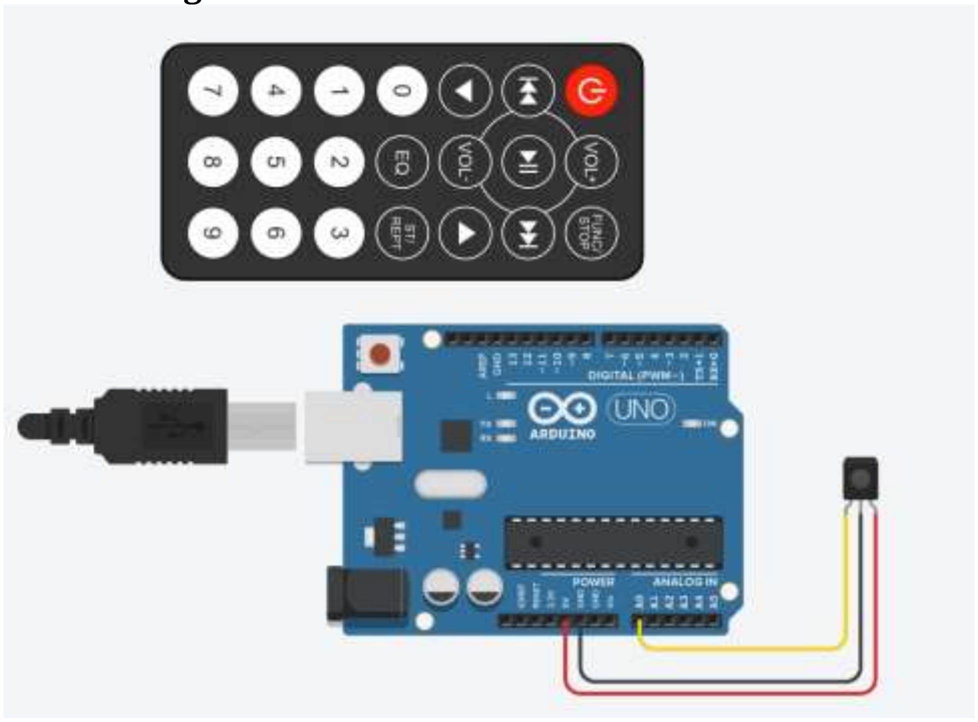
Practical No. 22

Aim: To interface IR remote with Arduino Write a program to create useful commands from the IR remote control.

Component: Arduino Uno, IR Sensor

Theory:

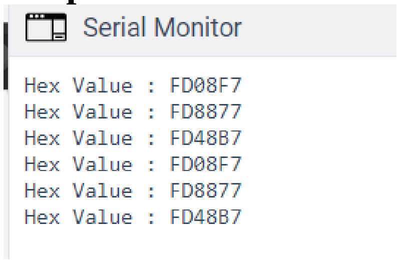
Circuit Diagram:



Program:

```
#include <IRremote.h>

int IRpin = A0;
IRrecv IR(IRpin);

decode_results res;


void setup()
{
  Serial.begin(9600);
  IR.enableIRIn();
  IR.blink13(true);
}

void loop()
{
  while(IR.decode(&res) == 0){
  }
  IR.resume();
  if (res.value == 0xFD08F7){
    Serial.print("Hex Value : ");
    Serial.println(res.value,HEX);
  }

  else if (res.value == 0xFD8877){
    Serial.print("Hex Value : ");
    Serial.println(res.value,HEX);
  }

  else if (res.value == 0xFD48B7){
    Serial.print("Hex Value : ");
    Serial.println(res.value,HEX);
  }
}
```

Output:



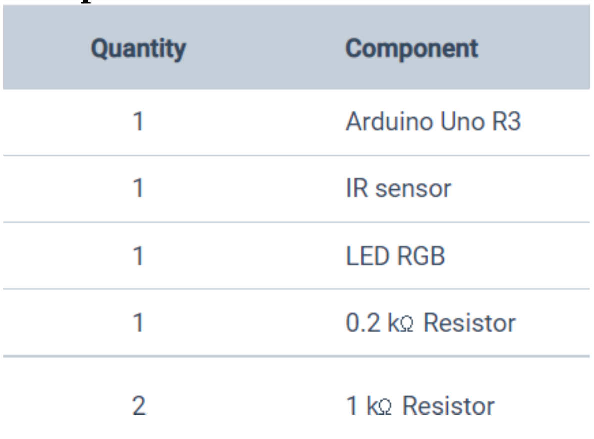
```
Hex Value : FD08F7
Hex Value : FD8877
Hex Value : FD48B7
Hex Value : FD08F7
Hex Value : FD8877
Hex Value : FD48B7
```

Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 23

Aim: To interface RGB led and IR remote, write a program to control RGB led with IR remote.

Component:



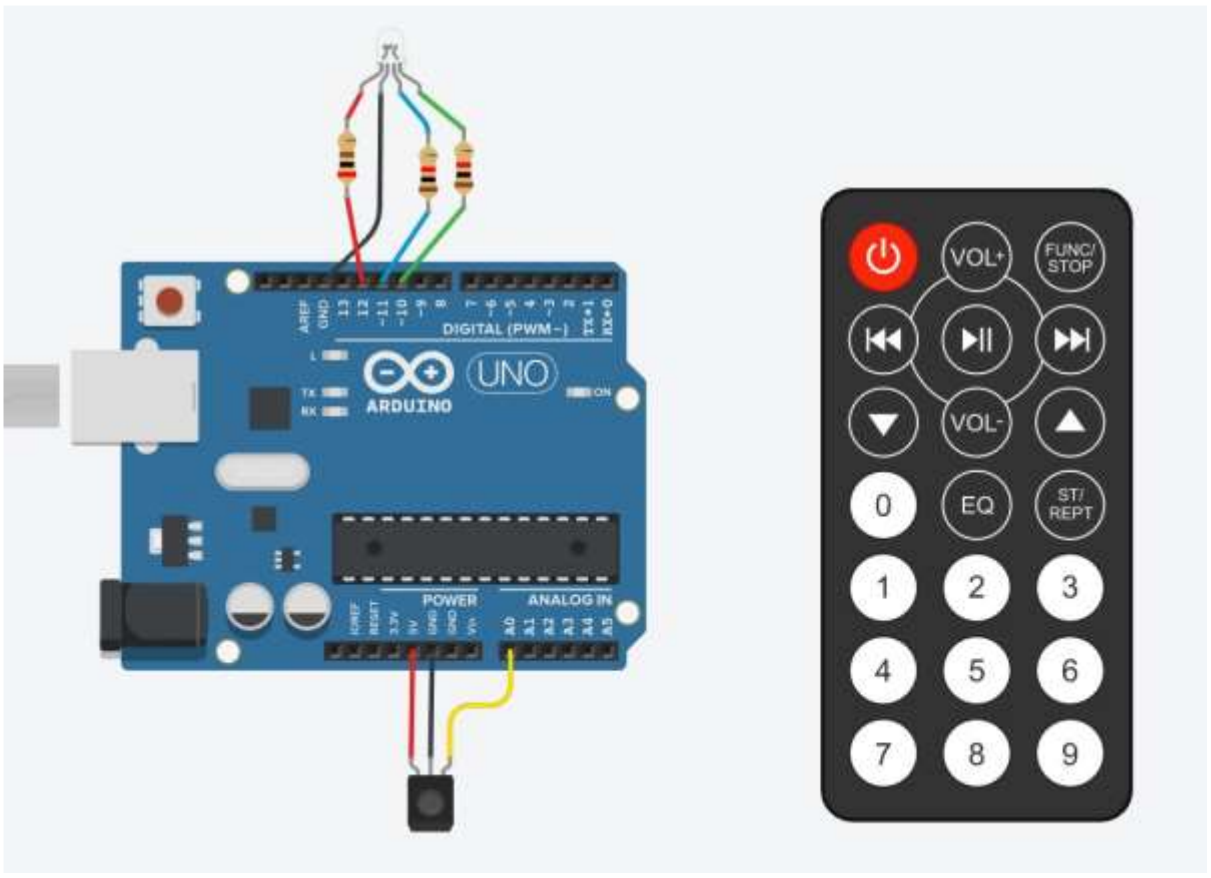
Quantity	Component
1	Arduino Uno R3
1	IR sensor
1	LED RGB
1	0.2 kΩ Resistor
2	1 kΩ Resistor

Theory:

(InfraRed remote control) A handheld, wireless device used to operate audio, video and other electronic equipment within a room using light signals in the infrared (IR) range.
Infrared light requires line of sight to its destination. Low-end remotes use only one transmitter at the end of the unit and have to be aimed directly at the equipment.
High-quality remotes have three or four powerful IR transmitters set at different angles to shower the room with signals.

Circuit Diagram:

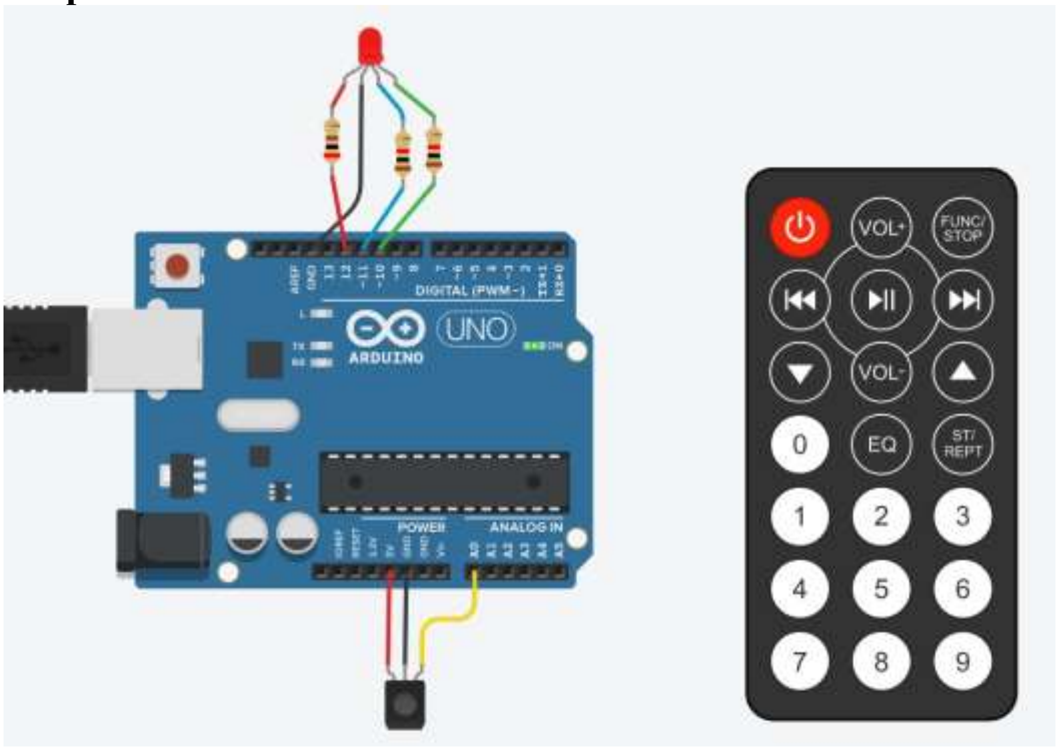
Name: Harshal Chavan Division: A
Roll No: 202124



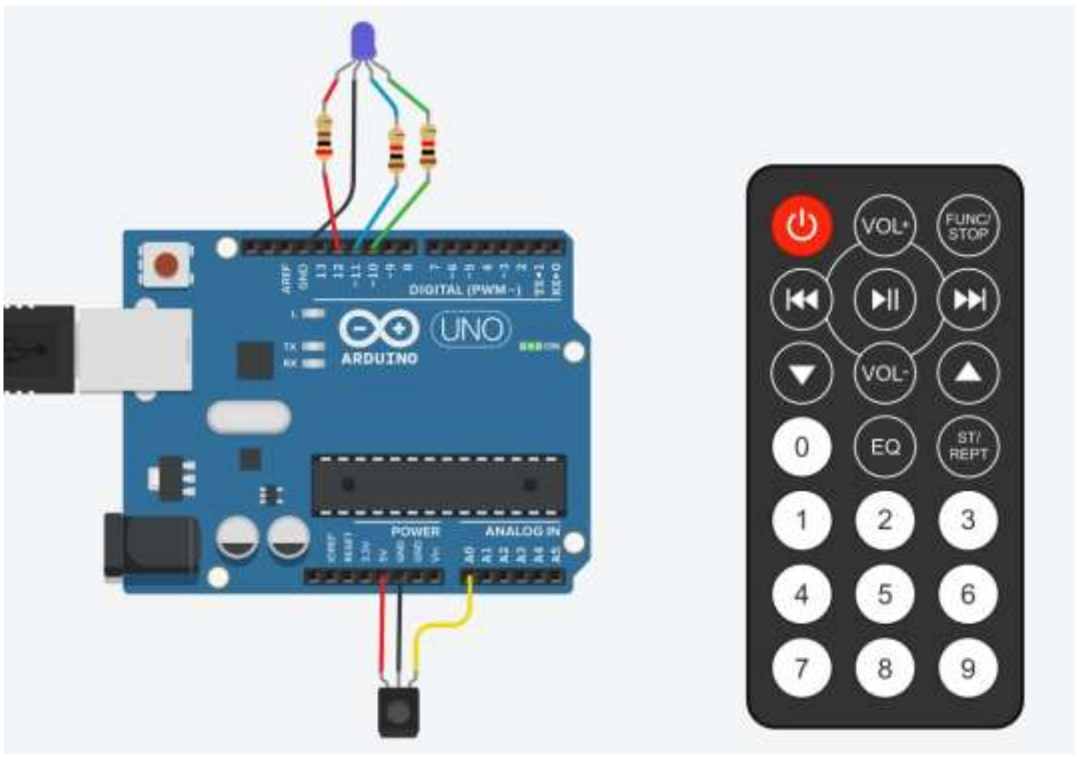
Program:

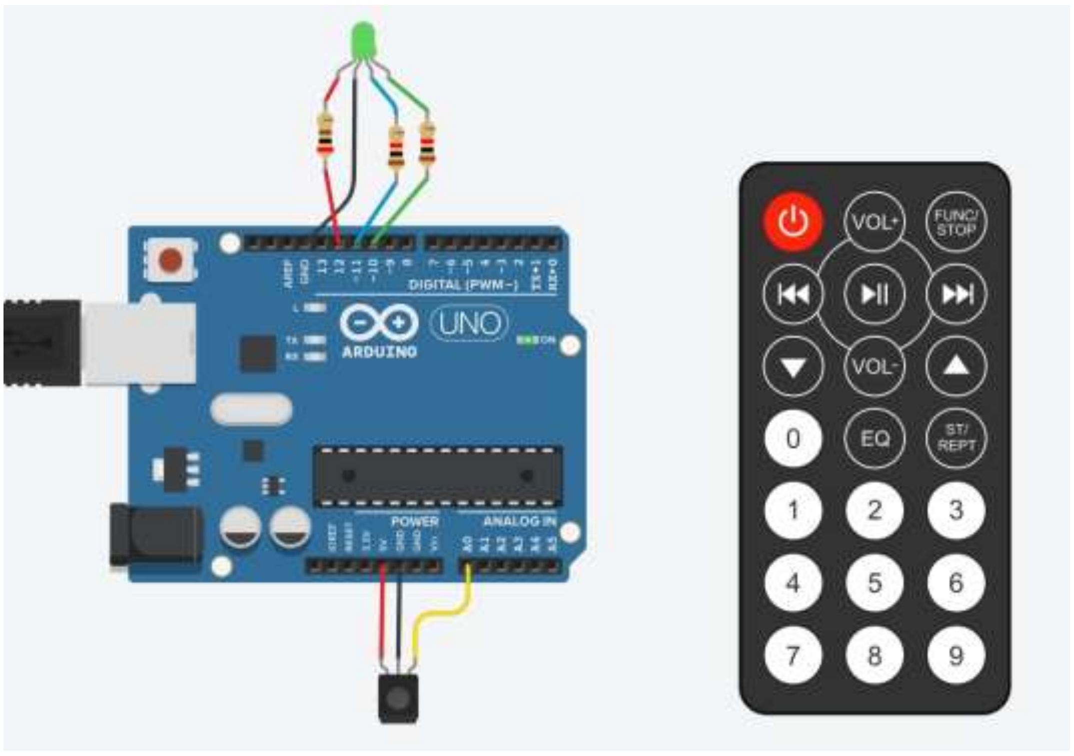
```
#include <IRremote.h>
int IRpin = A0;
IRrecv IR(IRpin);
decode_results res;
int red = 12;
int blue = 11;
int green = 10;
void setup()
{
  Serial.begin(9600);
  IR.enableIRIn();
  IR.blink13(true);
  pinMode(red, OUTPUT);
  pinMode(blue, OUTPUT);
  pinMode(green, OUTPUT);
}

void loop()
{
  while(IR.decode(&res) == 0){
  }
  //Serial.println(res.value);
  //Serial.print("Hex Value : ");
  Serial.println(res.value,HEX);
  IR.resume();
  if (res.value == 0xFD08F7){
    digitalWrite(red, HIGH);
    digitalWrite(blue, LOW);
    digitalWrite(green, LOW);
  }

  else if (res.value == 0xFD8877){
    digitalWrite(red, LOW);
    digitalWrite(blue, HIGH);
    digitalWrite(green, LOW);
  }

  else if (res.value == 0xFD48B7){
    digitalWrite(red, LOW);
    digitalWrite(blue, LOW);
    digitalWrite(green, HIGH);
  }}
```

Output:



Name: Harshal Chavan Division: A
Roll No: 202124





Name: Harshal Chavan   Division: A
Roll No: 202124

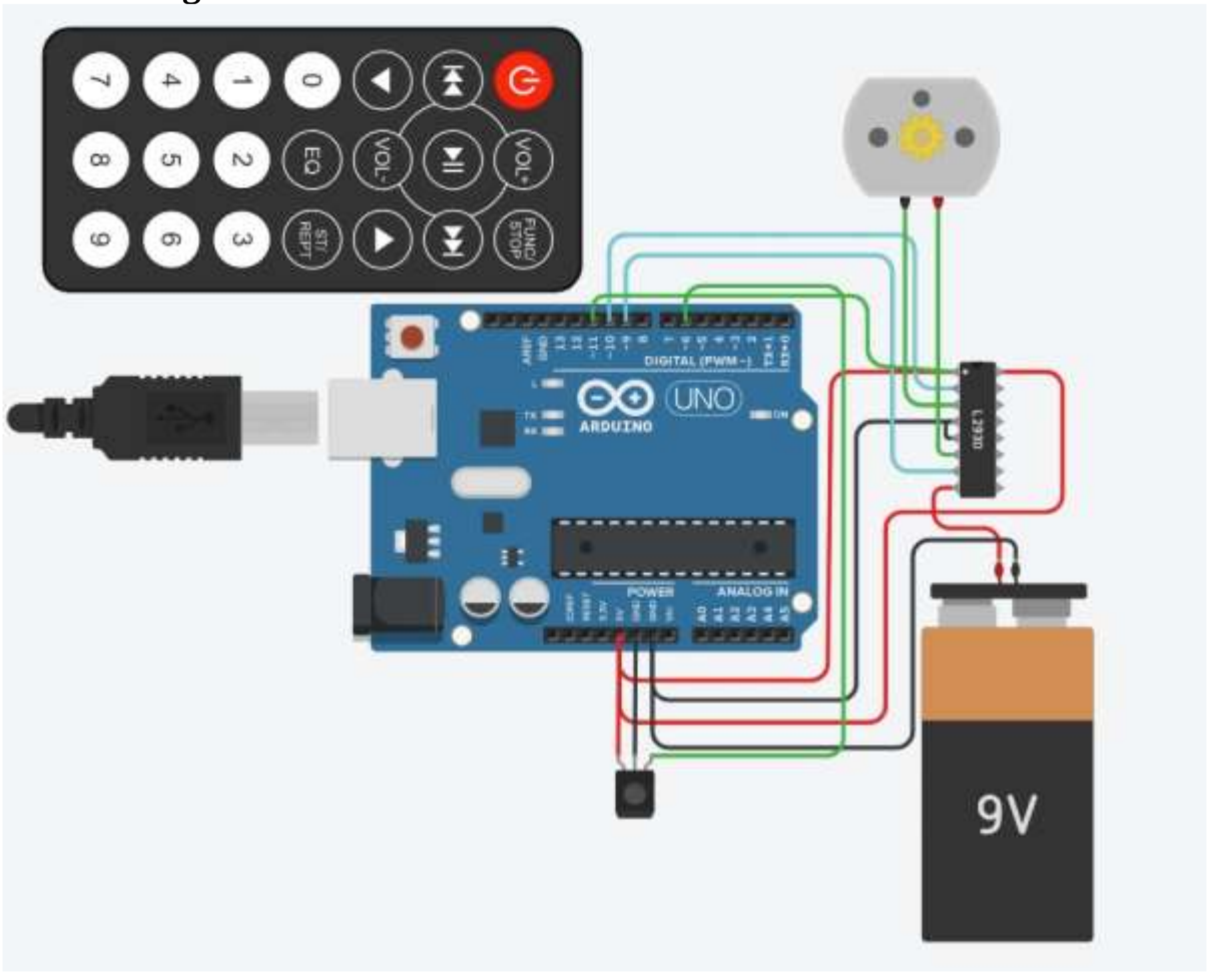
Practical No. 24

Aim: Write a program to control DC motor speed with an IR remote

Component:

Quantity	Component
1	Arduino Uno R3
1	DC Motor
1	H-bridge Motor Driver
1	9V Battery
1	IR sensor

Circuit Diagram:



Program:

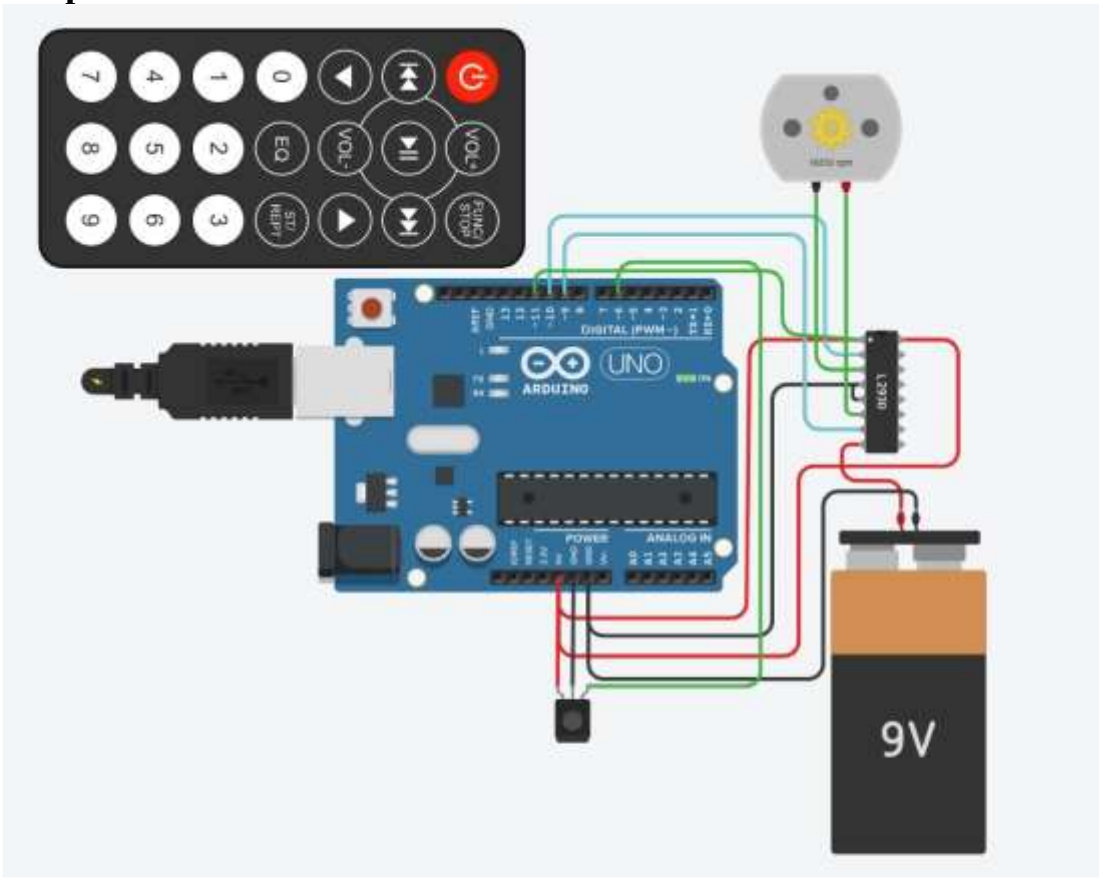
```
#include <IRremote.h>
#define speed1 2295
#define speed2 29885
#define speed4 12495
```

Name: Harshal Chavan   Division: A
 Roll No: 202124

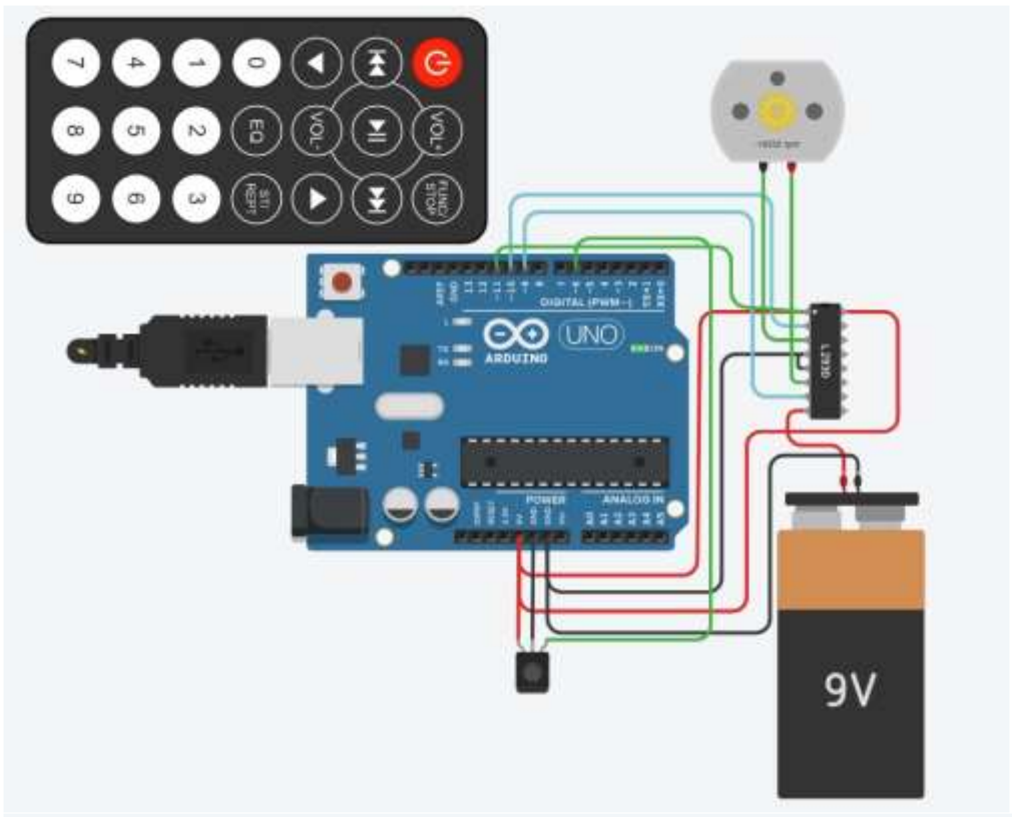
```
#define stop 255


// C++ code
// 1 and 2

int speedPin = 11;
int dir1= 9;
int dir2= 10;

int IRSensor = 6;

IRrecv IR(IRSensor);
decode_results results;

unsigned int value_decode;


void setup()
{
  pinMode(speedPin, OUTPUT);
  pinMode(dir1, OUTPUT);
  pinMode(dir2, OUTPUT);

  Serial.begin(9600);
  IR.enableIRIn();
}

void loop() {
  if (IR.decode(&results))
  {

    delay(1000);
    value_decode = results.value;
    Serial.println(value_decode);
    advance(value_decode);
    IR.resume(); // Receive the next value
  }
}

void advance(unsigned int option)
{
            switch(option)
    {
       case speed1:
                        digitalWrite(dir1,1);
                        digitalWrite(dir2,0);
                        analogWrite(speedPin,200);
                        break;
       case speed2:
```

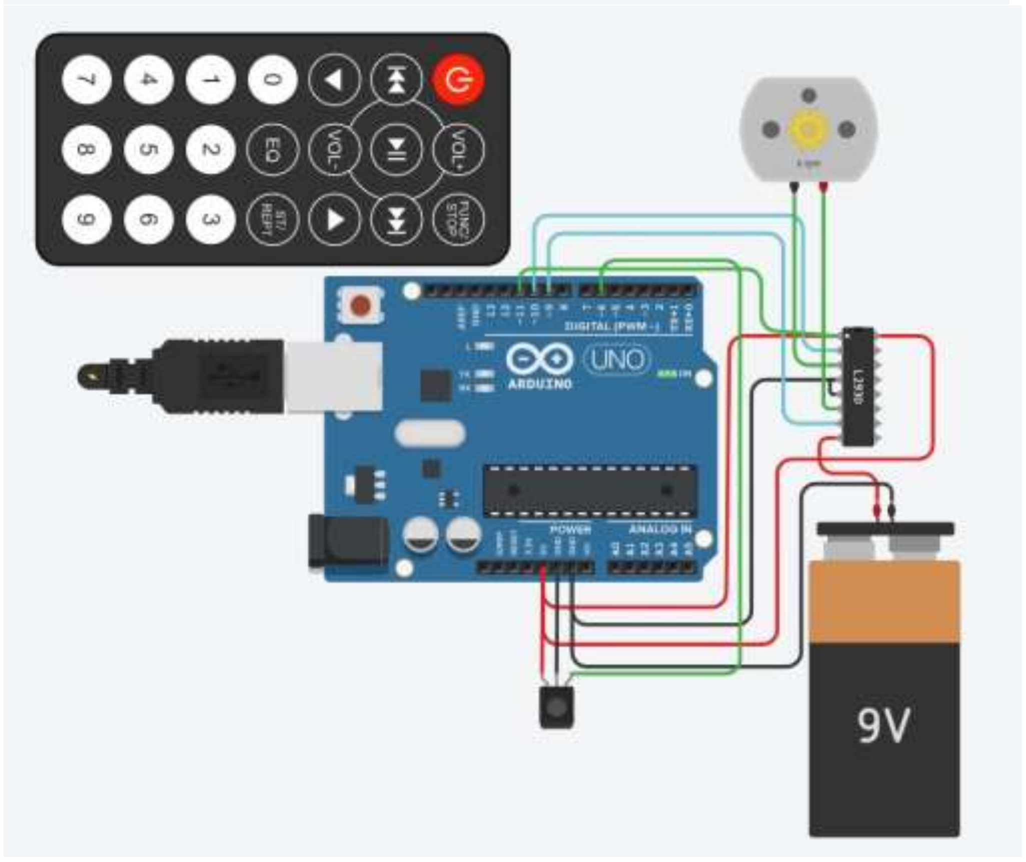
```
                digitalWrite(dir1,0);
                digitalWrite(dir2,1);
                analogWrite(speedPin,100);
                break;
   case stop:
                digitalWrite(dir1,0);
                digitalWrite(dir2,0);
        break;
   default:
                digitalWrite(dir1,0);
                digitalWrite(dir2,0);
                break;
   }
}
```

Output:



Name: Harshal Chavan Division: A
Roll No: 202124



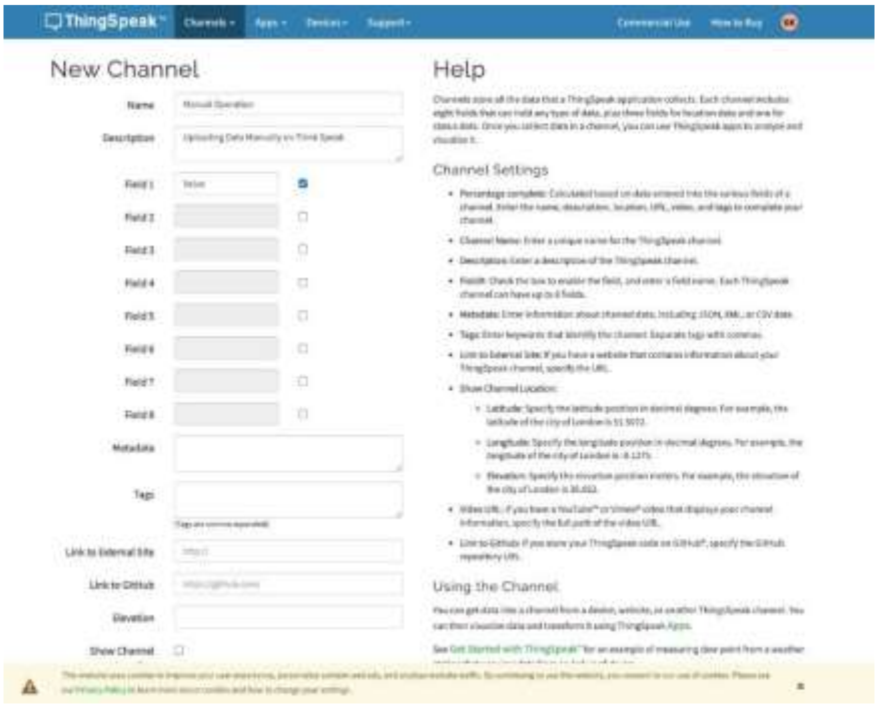


Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 25

Aim: Upload data on Thingspeak cloud manually.
Component:
Theory:
Steps:

1. **Create Channel on Thingspeak**



New Channel

Name: Manual Operation
Description: Uploading Data Manually on Think Speak
Field 1: Value ✓
Field 2
Field 3
Field 4
Field 5
Field 6
Field 7
Field 8
Metadata
Tags
Link to External Site: http://
Link to Github: https://github.com
Elevation
Show Channel

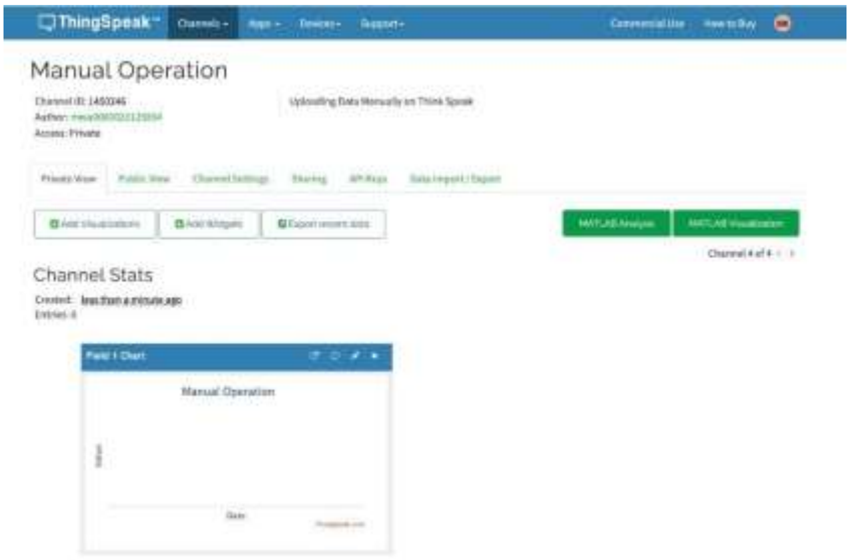
Help

Channel Settings

Using the Channel

[illegible]

2. **Manual Operation**



Manual Operation

Channel ID: 1450046
Author: [illegible]
Access: Private

Uploading Data Manually on Think Speak

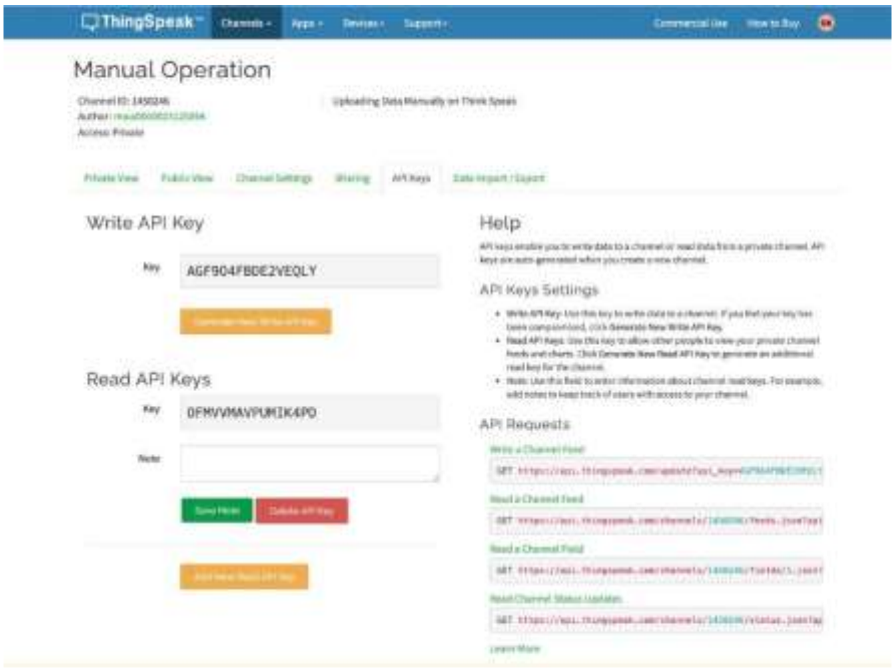
Channel 4 of 4

Channel Stats

Created: less than a minute ago
Entries: 0

3. **Click on API Keys**

Name: Harshal Chavan Division: A
Roll No: 202124



Manual Operation

Channel ID: 1450246
Author: [illegible]
Access: Private

Uploading Data Manually on Think Speak

Write API Key

Key AGF904FBDE2VEQLY

Read API Keys

Key 0FMVVMAVPUMIK4PO

Note

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- Write API Key: Use this key to write data to a channel. If you feel your key has been compromised, click Generate New Write API Key.
- Read API Keys: Use this key to allow other people to view your private channel feeds and charts. Click Generate New Read API Key to generate an additional read key for the channel.
- Note: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=AGF904FBDE2VEQLY
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/1450246/feeds.json?api
```

Read a Channel Field

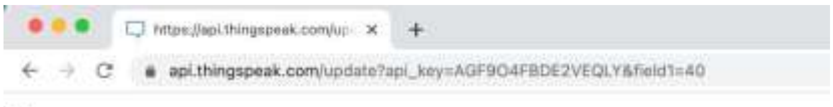
```
GET https://api.thingspeak.com/channels/1450246/fields/1.json?
```

Read Channel Status Updates

```
GET https://api.thingspeak.com/channels/1450246/status.json?ap
```

Learn More

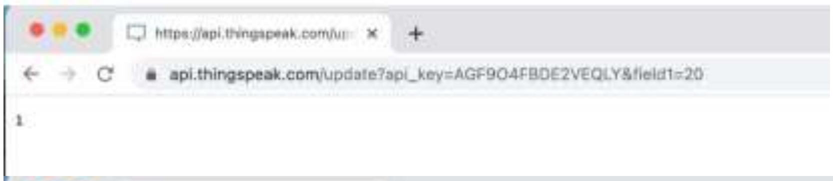
4. Calling the API by passing values in URL



https://api.thingspeak.com/up ×

api.thingspeak.com/update?api_key=AGF9O4FBDE2VEQLY&field1=40

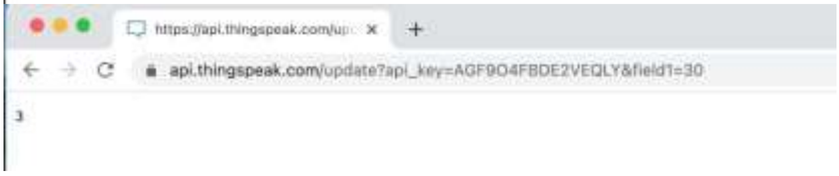
```
2
```



https://api.thingspeak.com/up ×

api.thingspeak.com/update?api_key=AGF9O4FBDE2VEQLY&field1=20

```
1
```



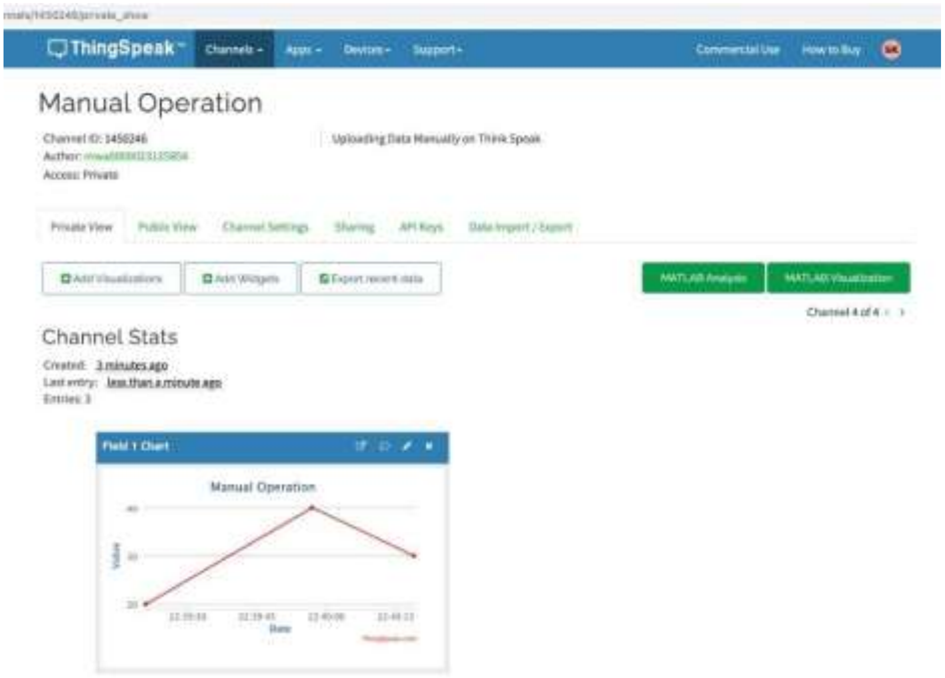
https://api.thingspeak.com/up ×

api.thingspeak.com/update?api_key=AGF9O4FBDE2VEQLY&field1=30

```
3
```

Name: Harshal Chavan Division: A
Roll No: 202124

5. Click on Private View to view the graph for passed values



Manual Operation

Channel ID: 1456246
Author: [illegible]
Access: Private

Uploading Data Manually on Think Speak

Channel Stats

Created: 3 minutes ago
Last entry: less than a minute ago
Entries: 3

Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 26

Aim: To update readings to Thingspeak from Arduino using Tinkercad.

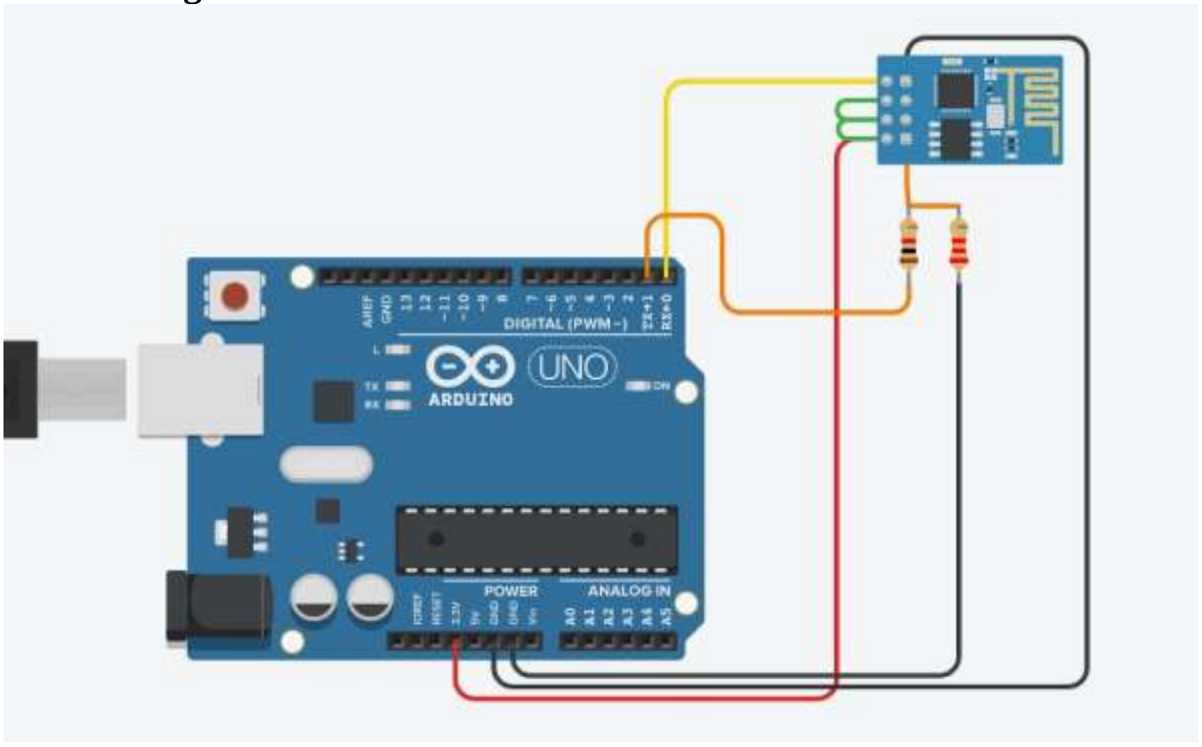
Component:

Quantity	Component
1	Arduino Uno R3
1	Wifi Module (ESP8266)
1	1 kΩ Resistor
1	2.2 kΩ Resistor

Theory:

ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP and MQTT protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates.

Circuit Diagram:

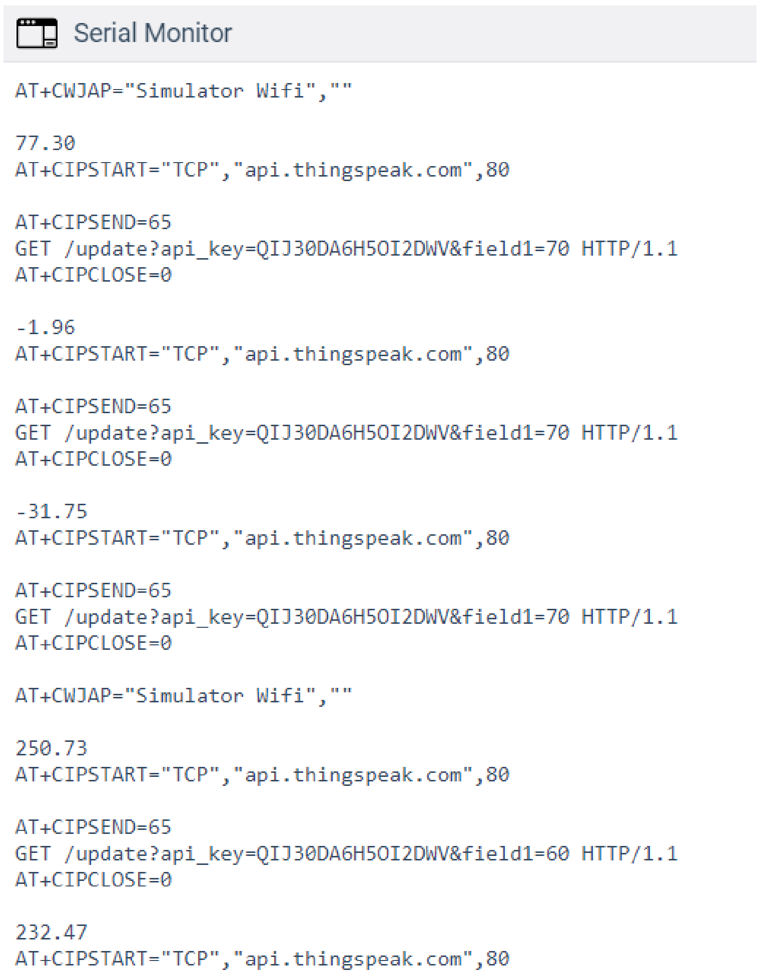


Program:

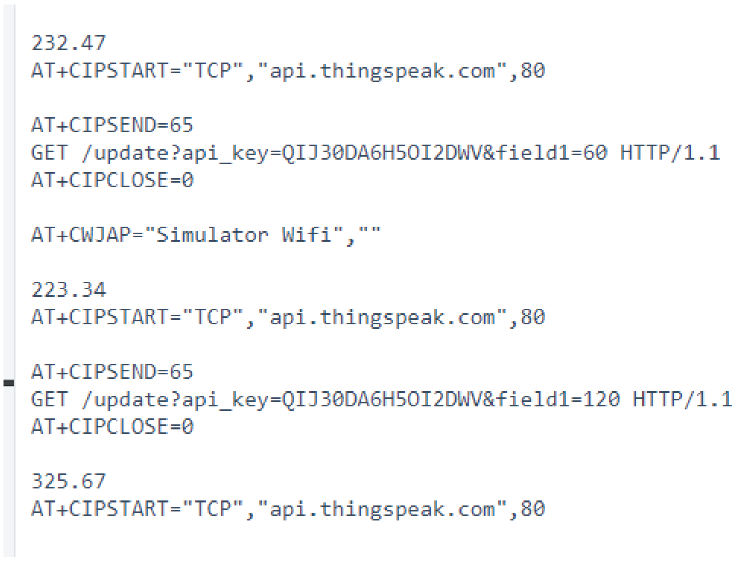
```
void setup() {

  Serial.begin(115200);
  pinMode(A0, INPUT);
  delay(1000);
  Serial.println("AT+CWJAP=\"Simulator Wifi\",\"\"\r\n");
  delay(3000);

}

void loop() {
  int senseValue = analogRead(A0);
  float volt = (senseValue/1020.0) * 4.9; //Volts float
  float tempC = (volt -0.5) * 100; //Celcius
  Serial.println(tempC);
  Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");
  delay(5000);
int len = 65;
Serial.print("AT+CIPSEND=");
Serial.println(len);
delay(10);
Serial.print("GET /update?api_key=QIJ30DA6H5OI2DWV&field1=120 HTTP/1.1\r\n");
delay(100);
Serial.println("AT+CIPCLOSE=0\r\n");
delay(6000);
}
```

Output:

Name: Harshal Chavan Division: A
Roll No: 202124

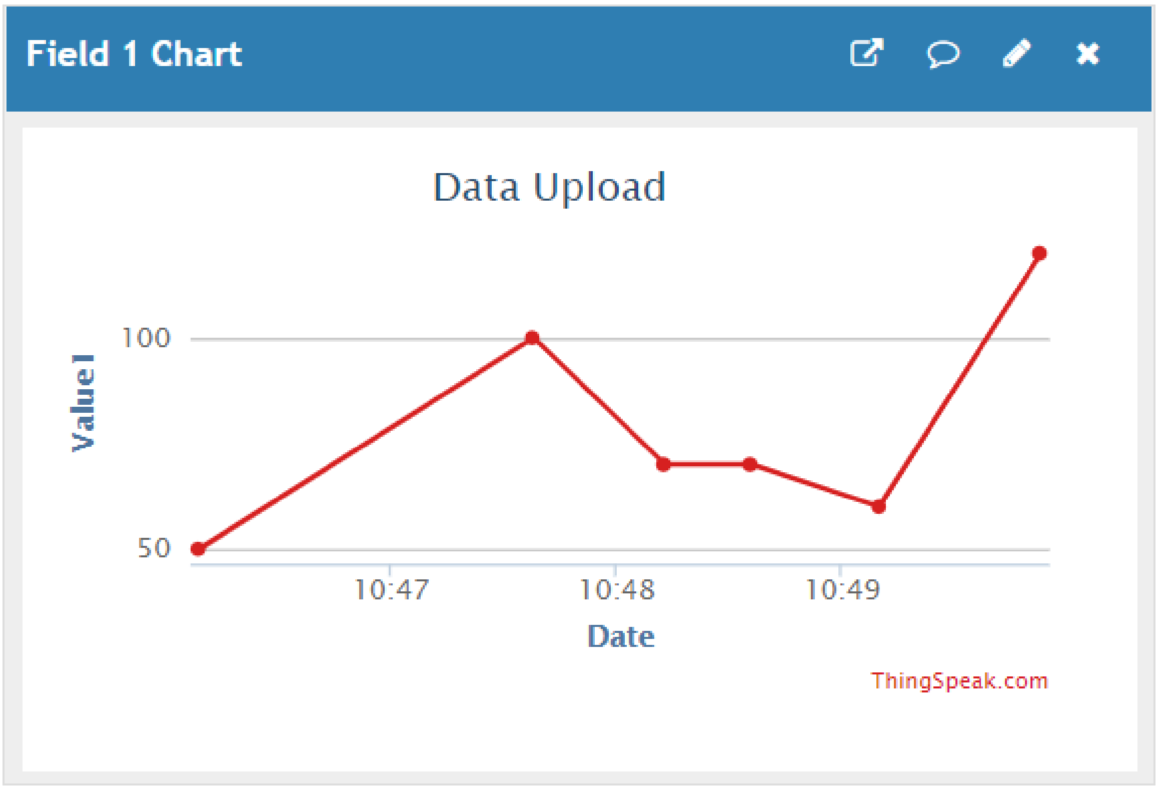


```
AT+CWJAP="Simulator Wifi",""

77.30
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=QIJ30DA6H5OI2DWV&field1=70 HTTP/1.1
AT+CIPCLOSE=0

-1.96
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=QIJ30DA6H5OI2DWV&field1=70 HTTP/1.1
AT+CIPCLOSE=0

-31.75
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=QIJ30DA6H5OI2DWV&field1=70 HTTP/1.1
AT+CIPCLOSE=0

AT+CWJAP="Simulator Wifi",""

250.73
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=QIJ30DA6H5OI2DWV&field1=60 HTTP/1.1
AT+CIPCLOSE=0

232.47
AT+CIPSTART="TCP","api.thingspeak.com",80
```



```
232.47
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=QIJ30DA6H5OI2DWV&field1=60 HTTP/1.1
AT+CIPCLOSE=0

AT+CWJAP="Simulator Wifi",""

223.34
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=QIJ30DA6H5OI2DWV&field1=120 HTTP/1.1
AT+CIPCLOSE=0

325.67
AT+CIPSTART="TCP","api.thingspeak.com",80
```

Name: Harshal Chavan Division: A
Roll No: 202124



Name: Harshal Chavan Division: A
Roll No: 202124

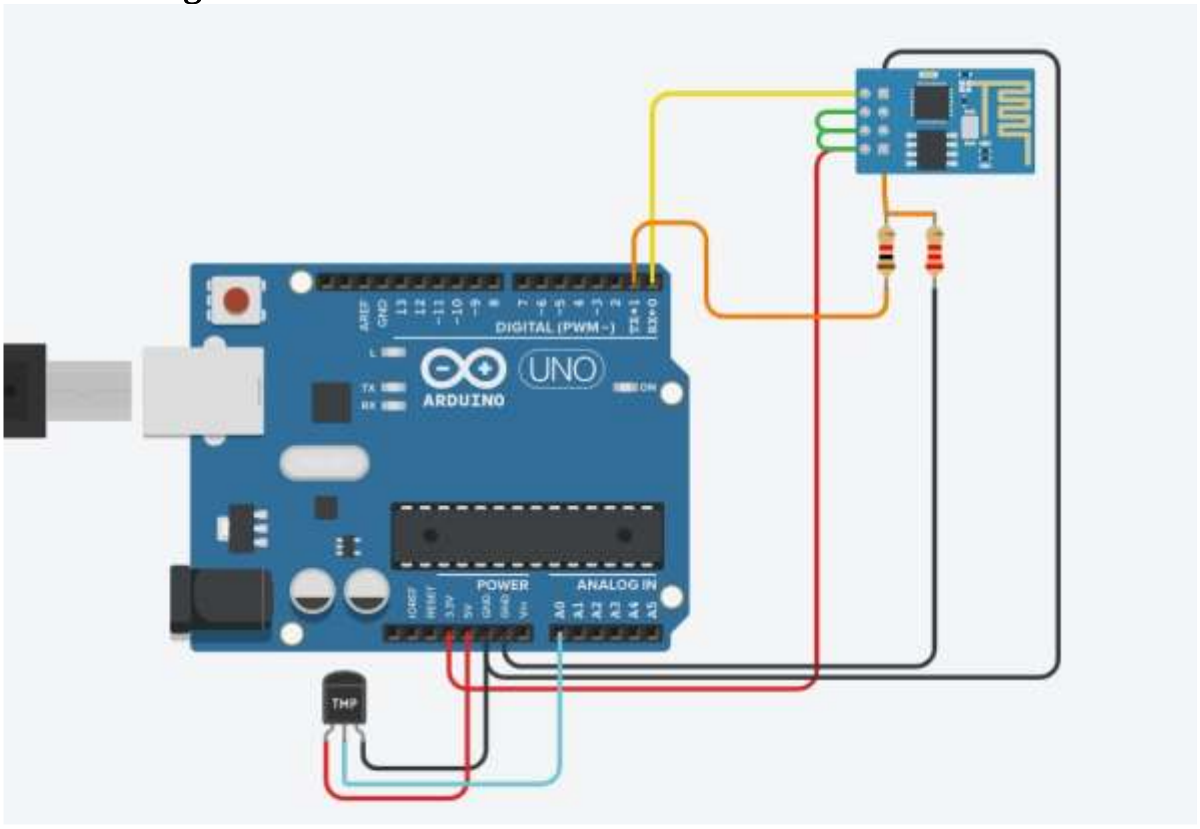
Practical No. 27

Aim: To interface Temperature sensor and ESP8266 with Arduino and update temperature reading to Thingspeak.

Component:

Quantity	Component
1	Arduino Uno R3
1	Wifi Module (ESP8266)
1	1 kΩ Resistor
1	2.2 kΩ Resistor
1	Temperature Sensor [TMP36]

Circuit Diagram:



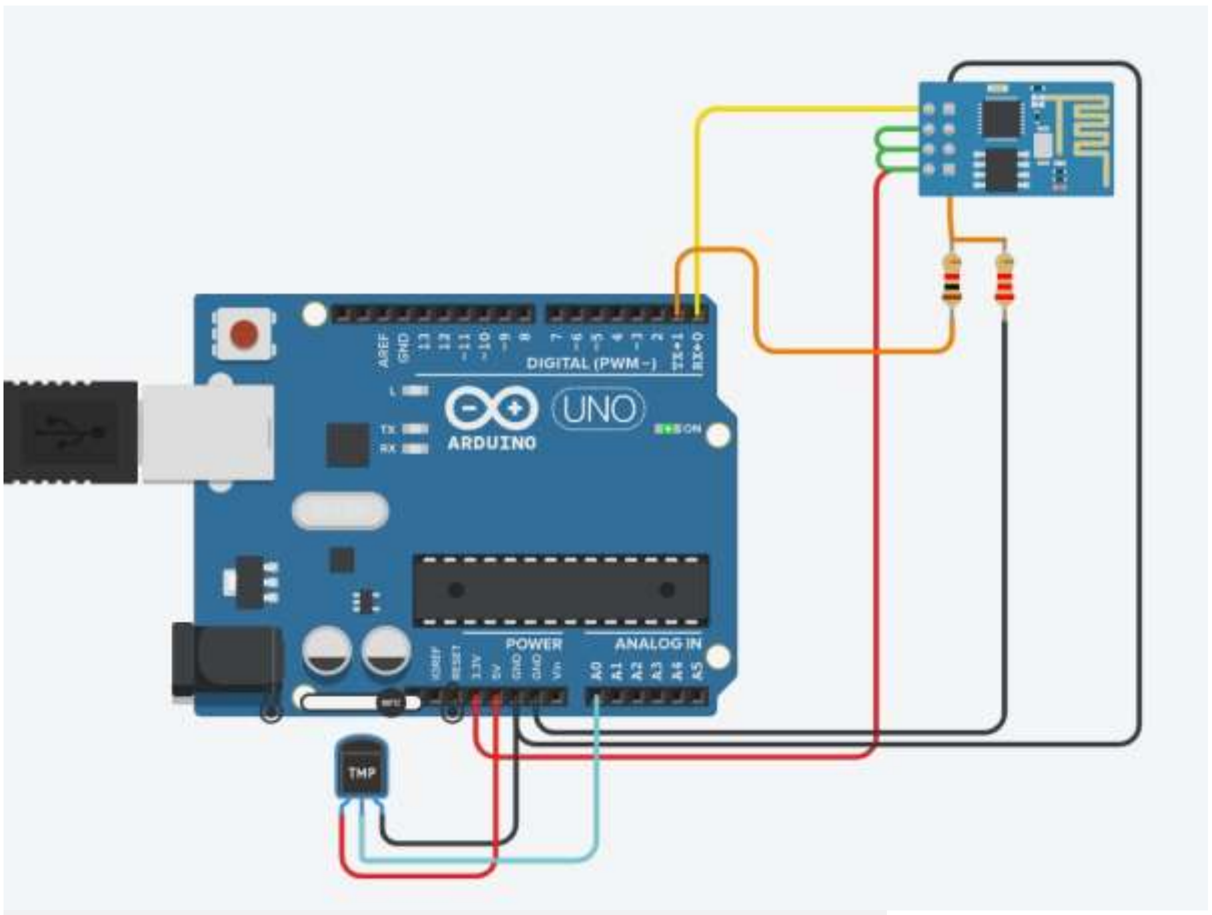
Name: Harshal Chavan Division: A
Roll No: 202124

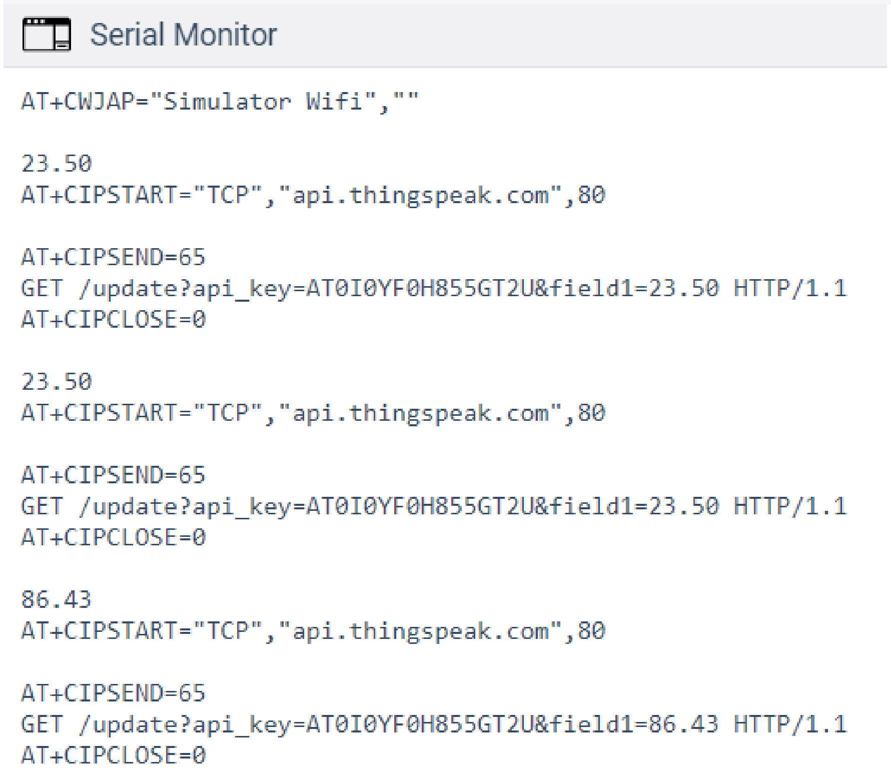
Program:

```
void setup() {

  Serial.begin(115200);
  pinMode(A0, INPUT);
  delay(1000);
  Serial.println("AT+CWJAP=\"Simulator Wifi\",\"\"\r\n");
  delay(3000);

}

void loop() {
  int senseValue = analogRead(A0);
  float volt = (senseValue/1020.0) * 4.9; //Volts float
  float tempC = (volt -0.5) * 100; //Celcius
  Serial.println(tempC);
  Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");
  delay(5000);
int len = 65;
Serial.print("AT+CIPSEND=");
Serial.println(len);
delay(10);
Serial.print("GET /update?api_key=AT0I0YF0H855GT2U&field1=" + String(tempC)+"
HTTP/1.1\r\n");
delay(100);
Serial.println("AT+CIPCLOSE=0\r\n");
delay(6000);
}
```

Output:

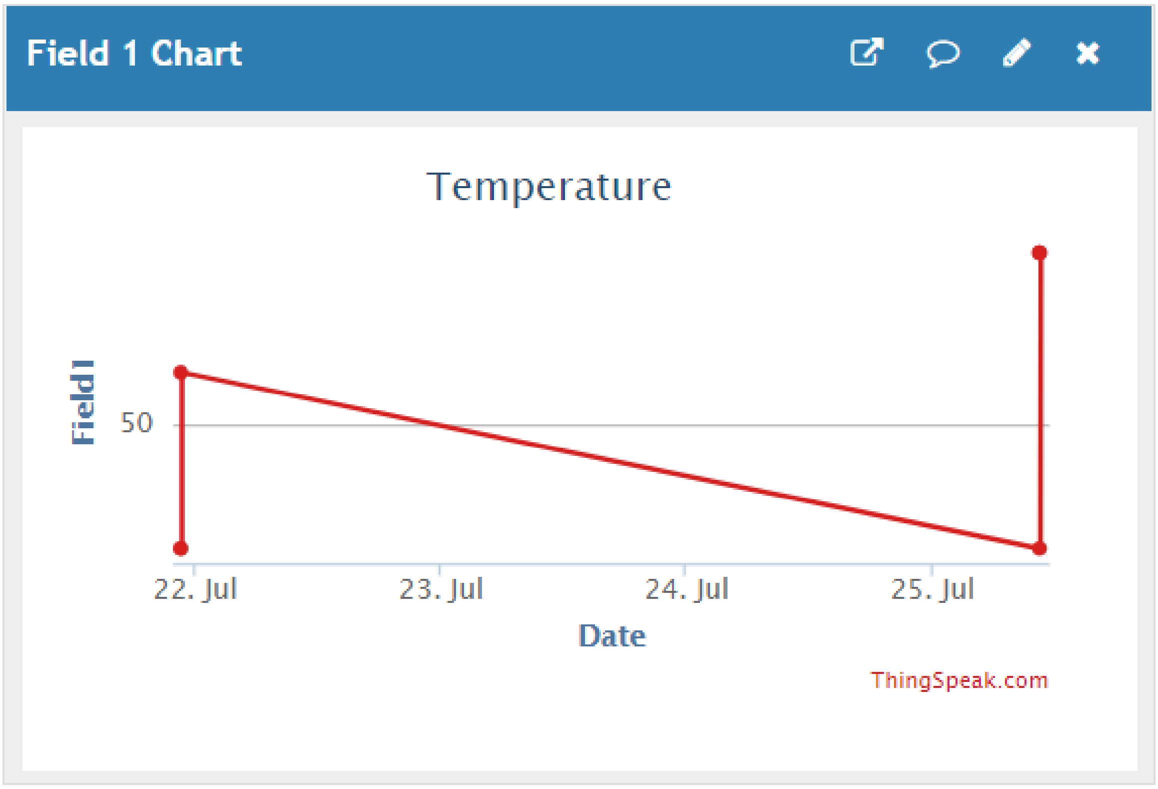
Name: Harshal Chavan Division: A
Roll No: 202124




Serial Monitor

```
AT+CWJAP="Simulator Wifi",""

23.50
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=AT0I0YF0H855GT2U&field1=23.50 HTTP/1.1
AT+CIPCLOSE=0

23.50
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=AT0I0YF0H855GT2U&field1=23.50 HTTP/1.1
AT+CIPCLOSE=0

86.43
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=65
GET /update?api_key=AT0I0YF0H855GT2U&field1=86.43 HTTP/1.1
AT+CIPCLOSE=0
```

Name: Harshal Chavan Division: A
Roll No: 202124

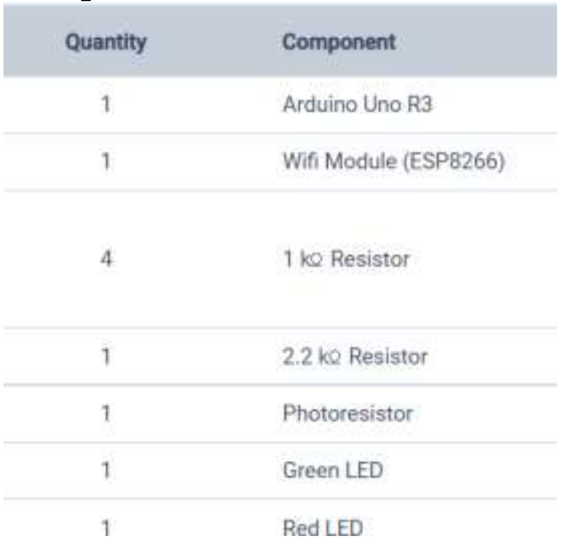


Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 28

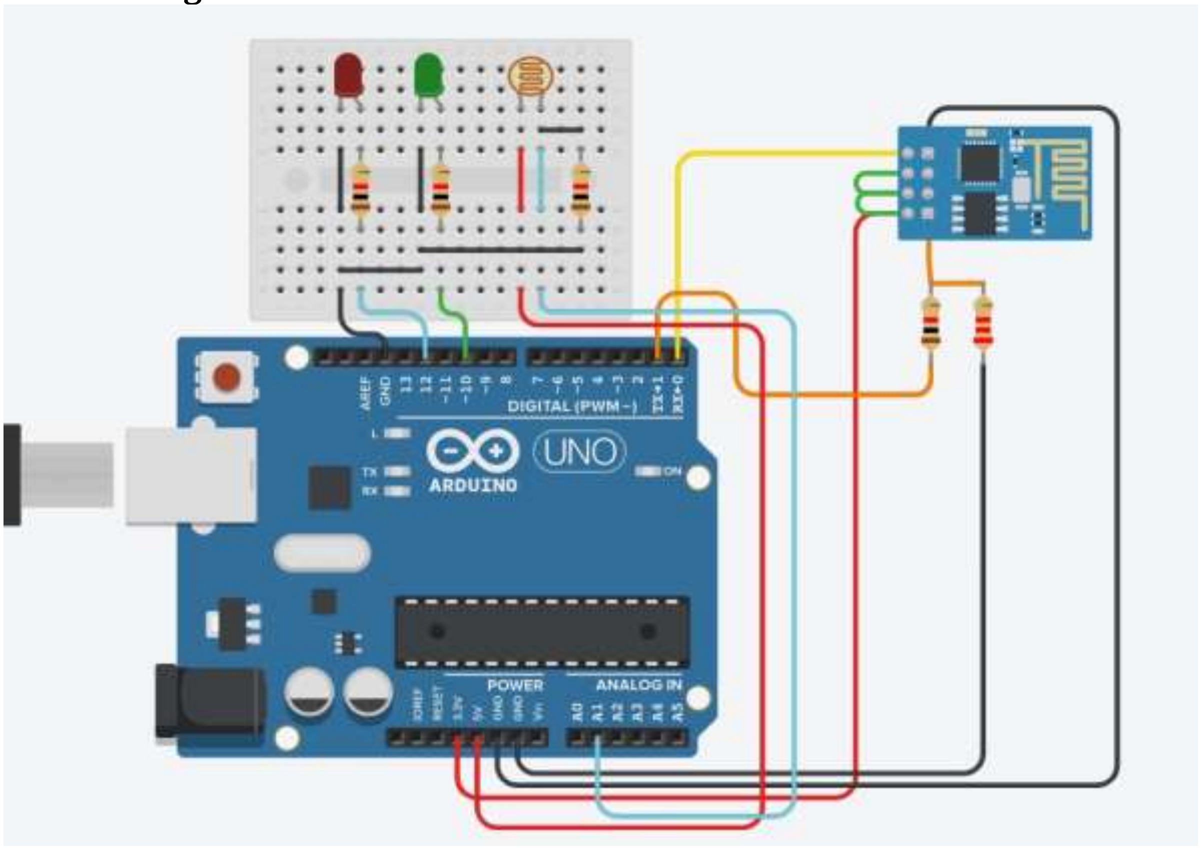
Aim: To interface LDR sensor, LED and ESP8266 with Arduino and update light intensity values to Thingspeak and tweet “LIGHT ON” message on tweeter when light intensity value is less than 300.

Component:



Quantity	Component
1	Arduino Uno R3
1	Wifi Module (ESP8266)
4	1 kΩ Resistor
1	2.2 kΩ Resistor
1	Photoresistor
1	Green LED
1	Red LED

Circuit Diagram:



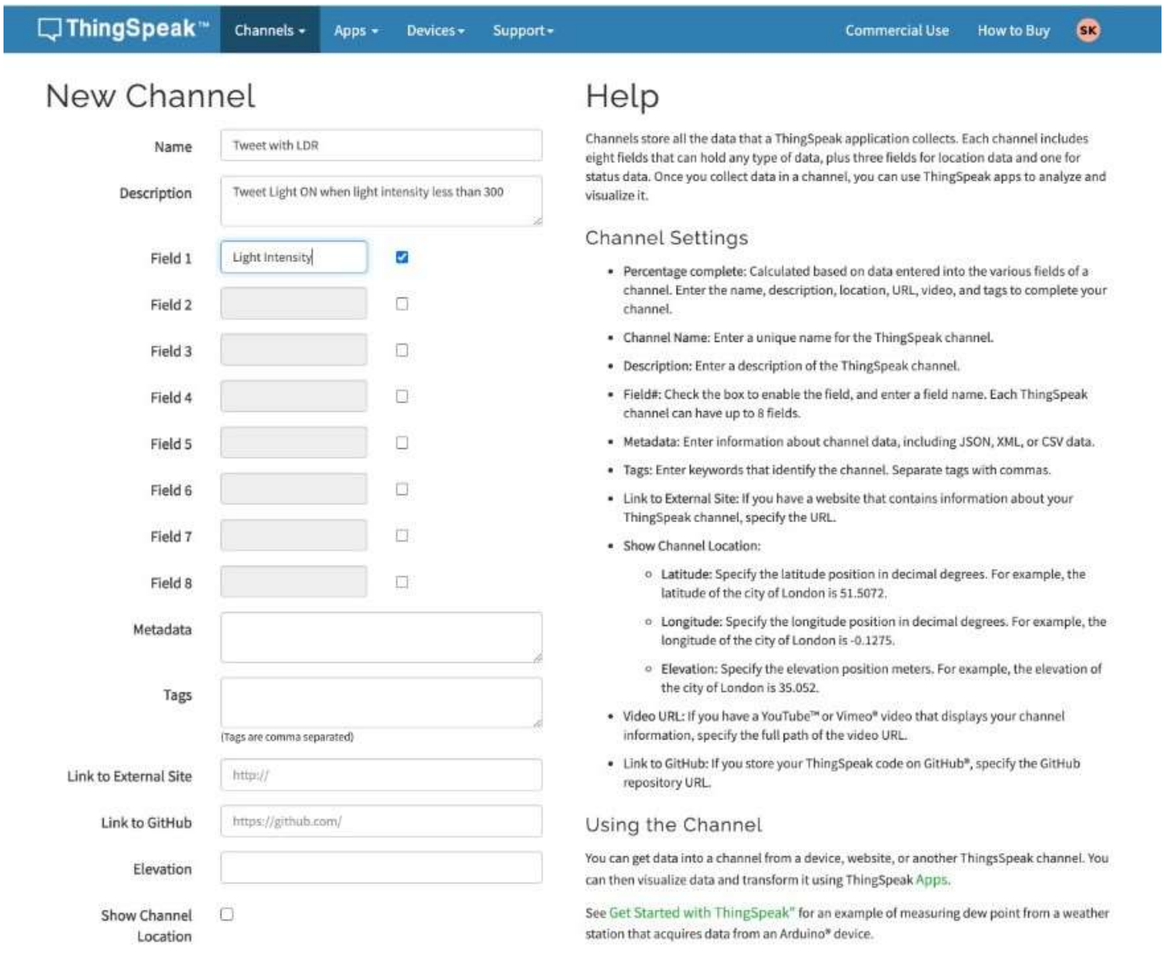
Program:

```
int sense_value;
void setup() {
  Serial.begin(115200);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(A1, INPUT);
  delay(1000);
  Serial.println("AT+CWJAP=\"Simulator Wifi\",\"\"\r\n");
  delay(3000);


}

void loop() {
  int sense_value = analogRead(A1);
  Serial.println();
  if (sense_value <= 300){
    digitalWrite(12, HIGH);
    digitalWrite(10, LOW);
  }
  else {
    digitalWrite(10, HIGH);
    digitalWrite(12, LOW);
  }

  Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");
  delay(500);
int len = 65;
Serial.print("AT+CIPSEND=");
Serial.println(len);
delay(10);
Serial.print("GET /update?api_key=T1HHE4KTNLIKELVC&field1=" +
String(sense_value)+" HTTP/1.1\r\n");
delay(100);
Serial.println("AT+CIPCLOSE=0\r\n");
delay(600);
}
```

Name: Harshal Chavan Division: A
Roll No: 202124

Steps:

1. Create channel for Tweet with LDR



New Channel

Name	Tweet with LDR
Description	Tweet Light ON when light intensity less than 300
Field 1	Light Intensity ☑
Field 2	☐
Field 3	☐
Field 4	☐
Field 5	☐
Field 6	☐
Field 7	☐
Field 8	☐
Metadata	
Tags	(Tags are comma separated)
Link to External Site	http://
Link to GitHub	https://github.com/
Elevation	
Show Channel Location	☐

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

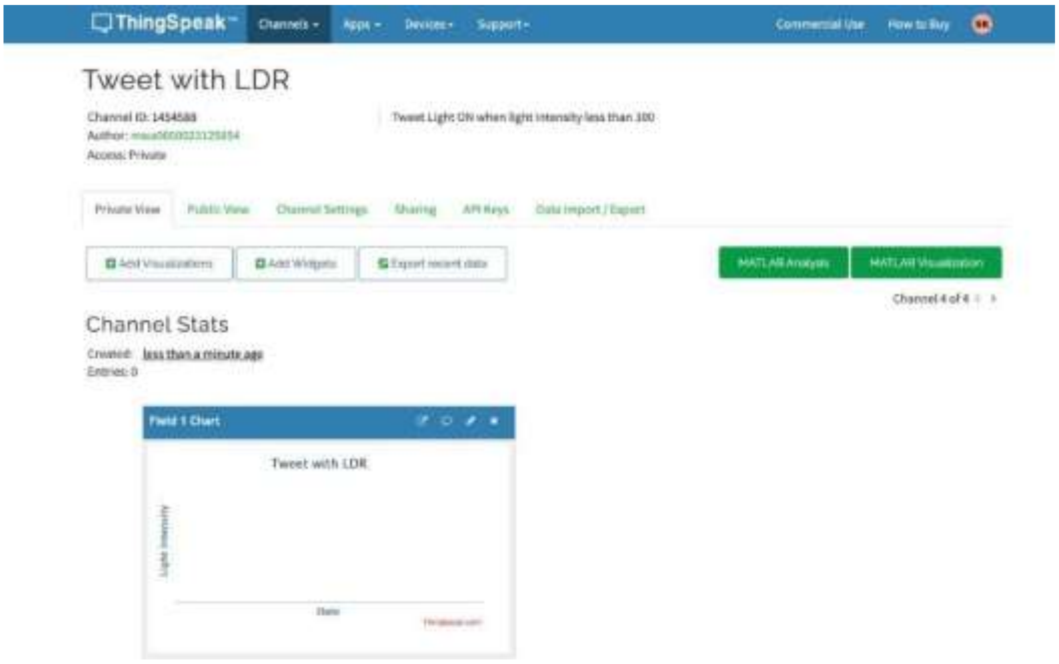
- Percentage complete: Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- Channel Name: Enter a unique name for the ThingSpeak channel.
- Description: Enter a description of the ThingSpeak channel.
- Field#: Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata: Enter information about channel data, including JSON, XML, or CSV data.
- Tags: Enter keywords that identify the channel. Separate tags with commas.
- Link to External Site: If you have a website that contains information about your ThingSpeak channel, specify the URL.
- Show Channel Location:
  - Latitude: Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - Longitude: Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - Elevation: Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- Video URL: If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- Link to GitHub: If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

Using the Channel

You can get data into a channel from a device, website, or another ThingsSpeak channel. You can then visualize data and transform it using ThingSpeak Apps.

See Get Started with ThingSpeak" for an example of measuring dew point from a weather station that acquires data from an Arduino® device.

2. Channel view



Tweet with LDR

Channel ID: 1454588
Author: [illegible]
Access: Private

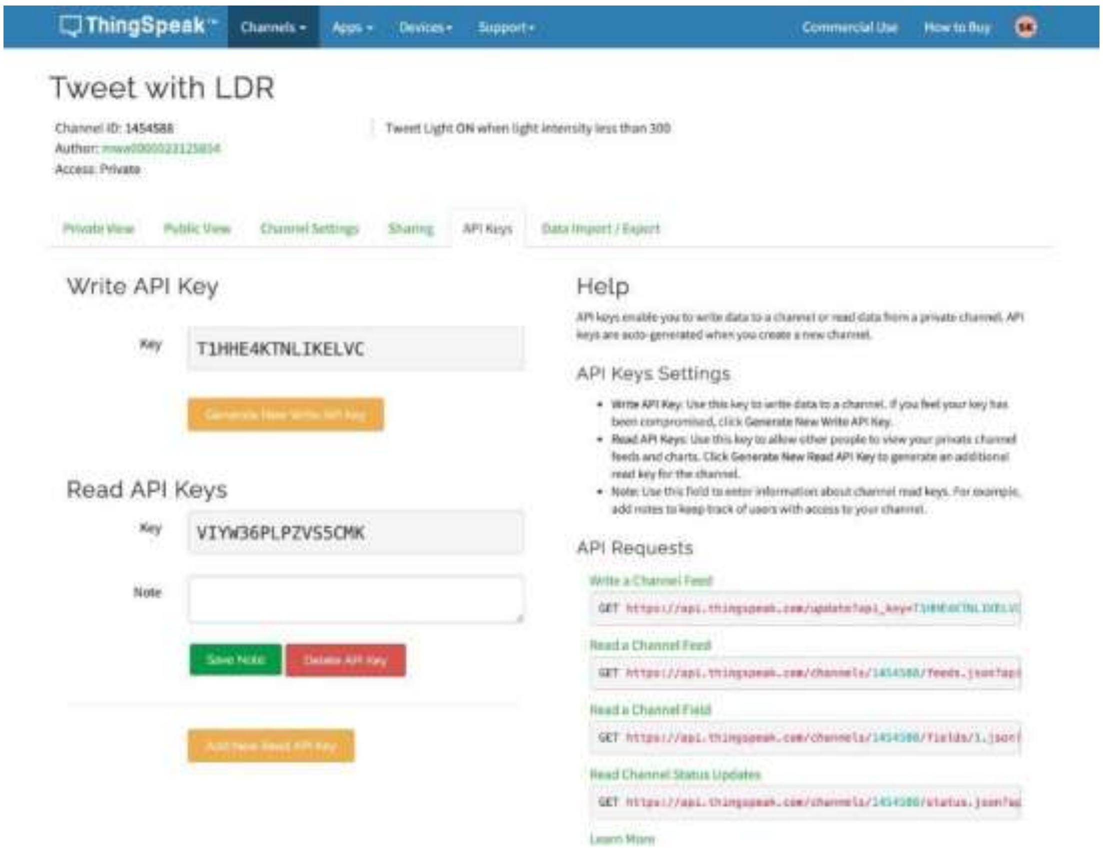
Tweet Light ON when light intensity less than 300

Channel Stats

Created: less than a minute ago
Entries: 0

Name: Harshal Chavan Division: A
Roll No: 202124

3. API view



ThingSpeak™ Channels ▾ Apps ▾ Devices ▾ Support ▾ Commercial Use How to Buy

Tweet with LDR

Channel ID: 1454588
Author: mwa0000023125834
Access: Private

Tweet Light ON when light intensity less than 300

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Write API Key

Key: T1HHE4KTNLIKELVC

Generate New Write API Key

Read API Keys

Key: VIYW36PLPZVS5CMK

Note:

Save Note | Delete API Key

Add New Read API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

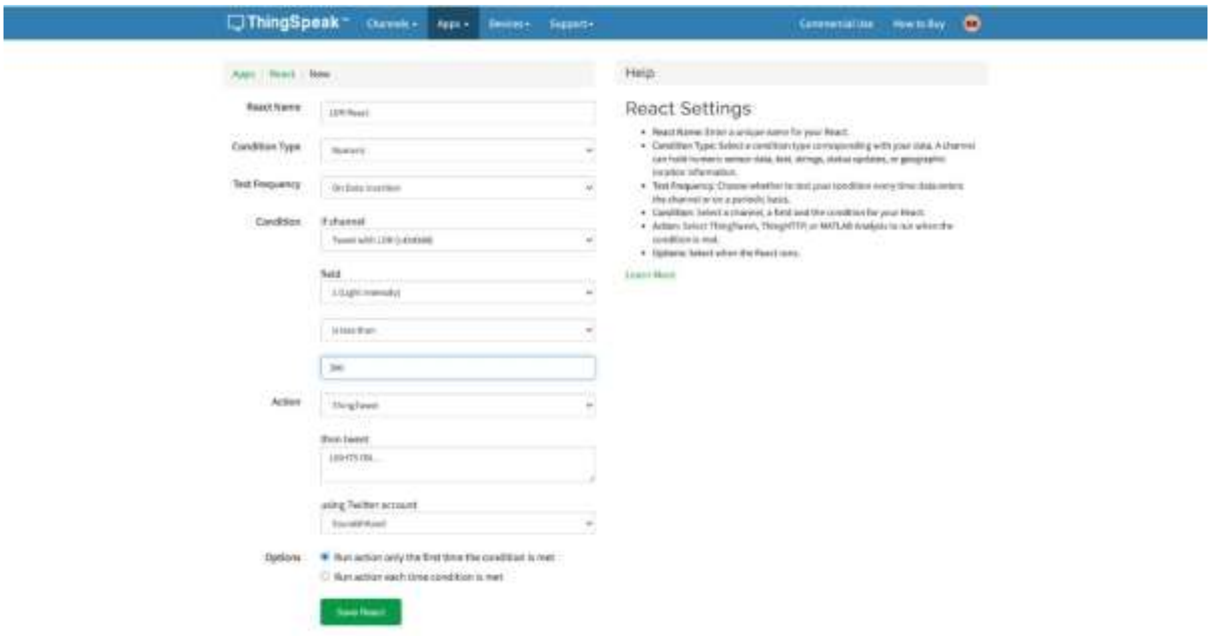
API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed
```
GET https://api.thingspeak.com/update?api_key=T1HHE4KTNLIKELVC
```
Read a Channel Feed
```
GET https://api.thingspeak.com/channels/1454588/feeds.json?api
```
Read a Channel Field
```
GET https://api.thingspeak.com/channels/1454588/fields/1.json
```
Read Channel Status Updates
```
GET https://api.thingspeak.com/channels/1454588/status.json?ap
```
Learn More

4. React App



ThingSpeak™ Channels ▾ Apps ▾ Devices ▾ Support ▾ Commercial Use How to Buy

Apps / React / New

React Name: LDR React
Condition Type: Numeric
Test Frequency: On Data Insertion
Condition: If channel: Tweet with LDR (1454588)
field: 1 (Light intensity)
is less than
300
Action: ThingTweet
then tweet: LIGHTS ON...
using Twitter account: [illegible]
Options: [✓ Run action only the first time the condition is met]
Run action each time condition is met

Save React

Help

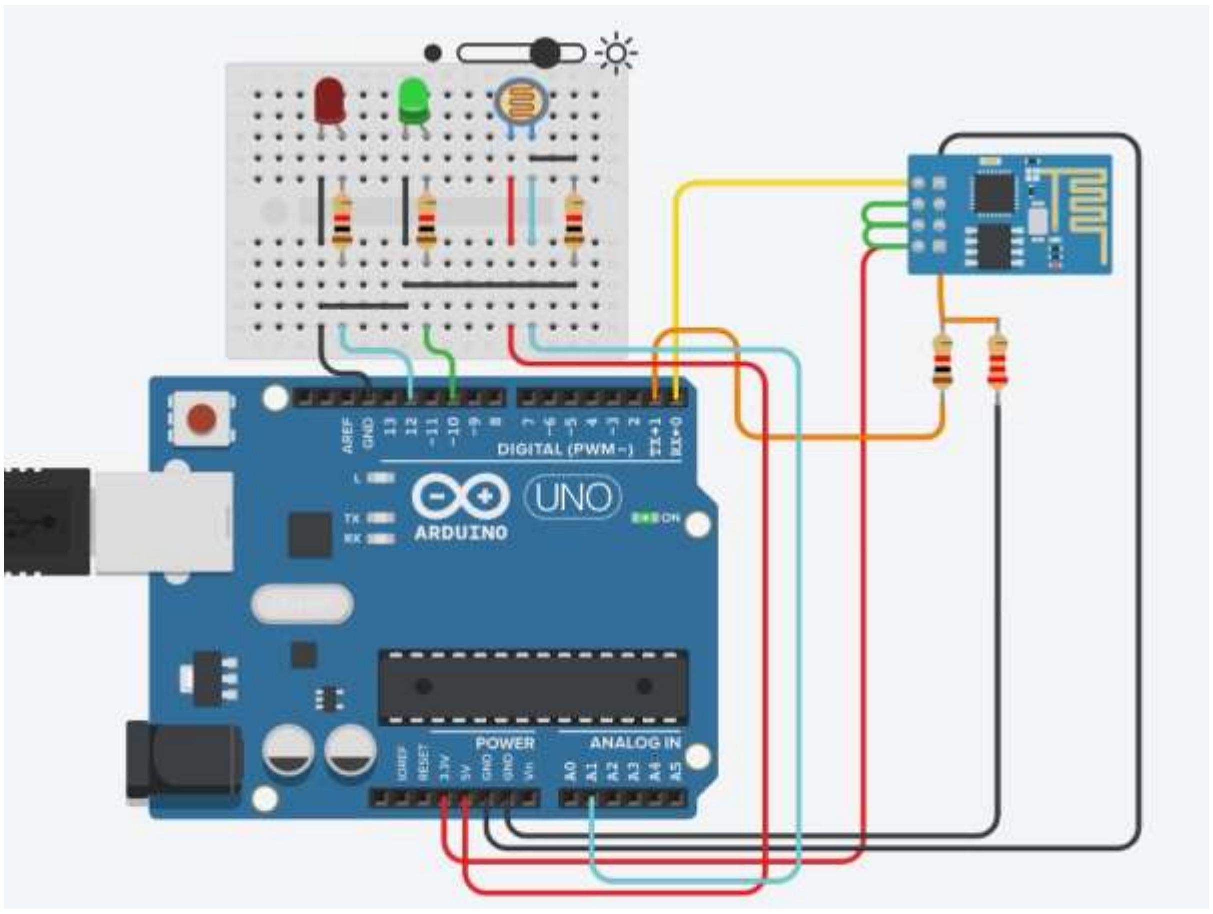
React Settings

- React Name: Enter a unique name for your React.
- Condition Type: Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- Test Frequency: Choose whether to test your condition every time data enters the channel or on a periodic basis.
- Condition: Select a channel, a field and the condition for your React.
- Action: Select ThingTweet, ThingHTTP or MATLAB Analysis to run when the condition is met.
- Options: Select when the React runs.

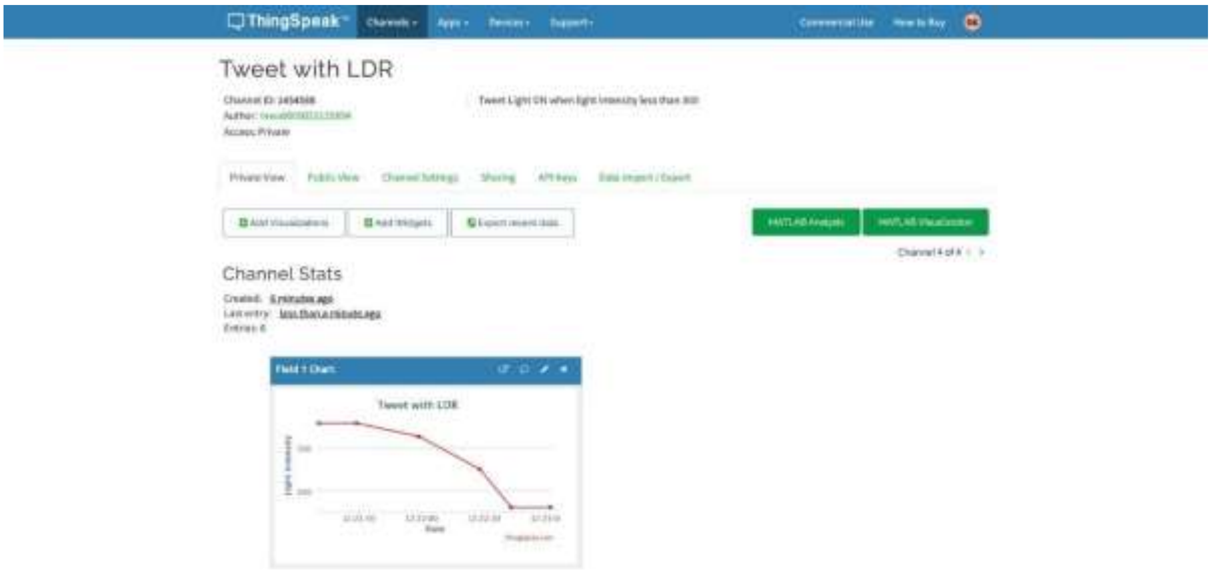
Learn More

Name: Harshal Chavan Division: A
Roll No: 202124

5. Start simulator and decrease LDR value below 300



6. Channel stats



Tweet with LDR

Channel ID: 1454568
Author: [illegible]
Access: Private

Tweet Light ON when light intensity less than 300

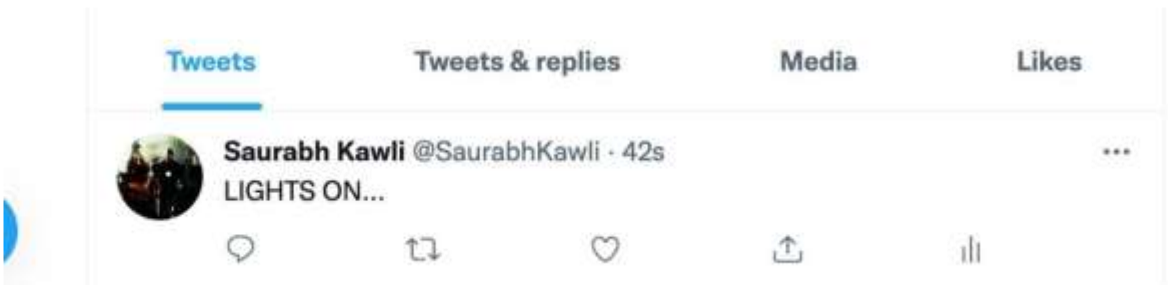
Channel 4 of 4

Channel Stats

Created: 6 minutes ago
Last entry: less than a minute ago
Entries: 6

Name: Harshal Chavan Division: A
Roll No: 202124

7. Tweet



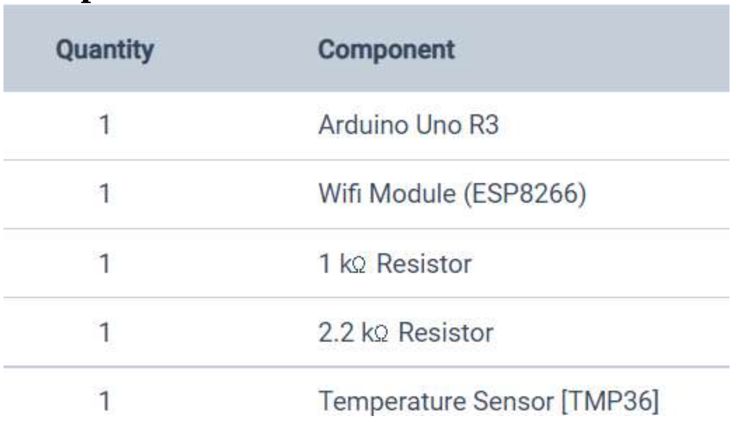
Saurabh Kawli @SaurabhKawli · 42s
LIGHTS ON...

Name: Harshal Chavan Division: A
Roll No: 202124

Practical No. 29

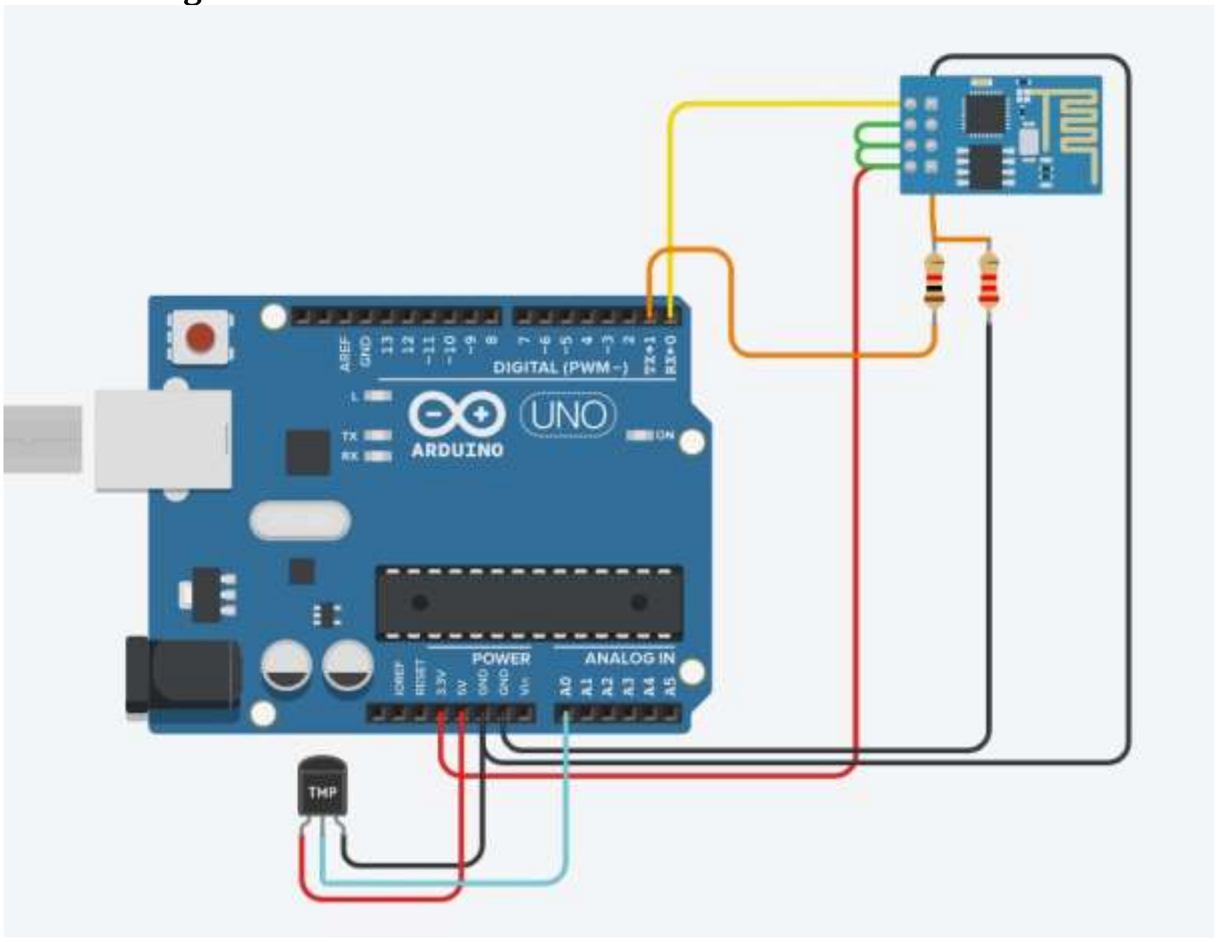
Aim: To interface Temperature sensor and ESP8266 with Arduino and update temperature values to Thingspeak and tweet “High Temp” message on tweeter when temperature value is greater than 40C.

Component:



Quantity	Component
1	Arduino Uno R3
1	Wifi Module (ESP8266)
1	1 kΩ Resistor
1	2.2 kΩ Resistor
1	Temperature Sensor [TMP36]

Circuit Diagram:

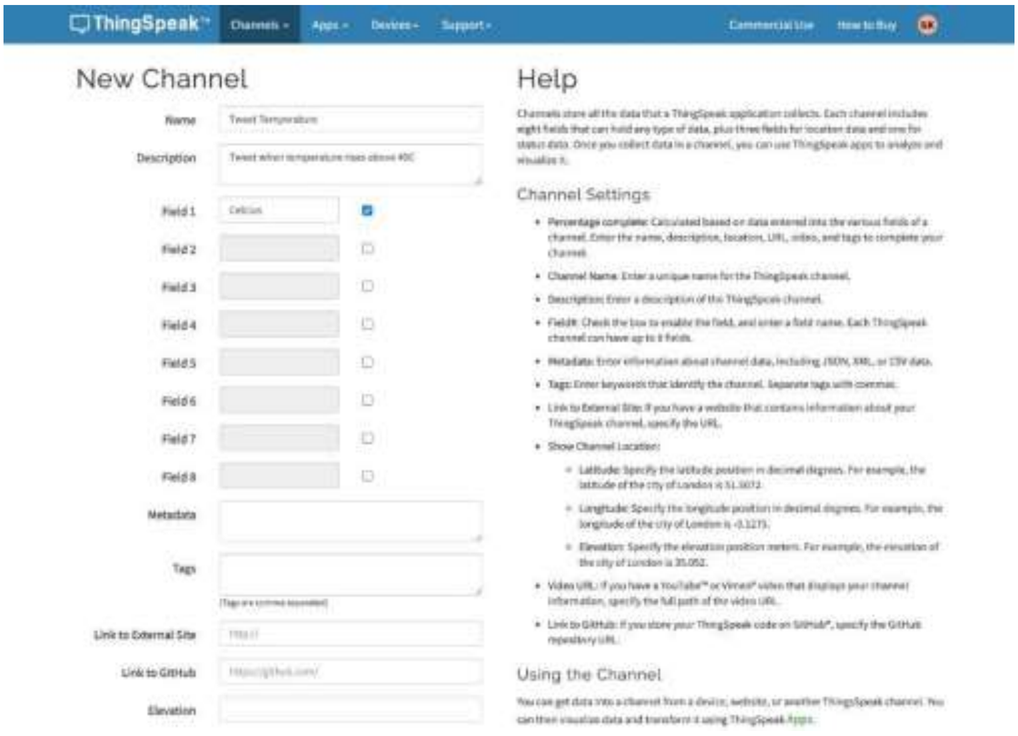


Program:

```
void setup() {

  Serial.begin(115200);
  pinMode(A0, INPUT);
  delay(1000);
  Serial.println("AT+CWJAP=\"Simulator Wifi\",\"\"\r\n");
  delay(3000);

}

void loop() {
  int senseValue = analogRead(A0);
  float volt = (senseValue/1020.0) * 4.9; //Volts float
  float tempC = (volt -0.5) * 100; //Celcius
  Serial.println(tempC);
  Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");
  delay(5000);
int len = 65;
Serial.print("AT+CIPSEND=");
Serial.println(len);
delay(10);
Serial.print("GET /update?api_key=AT0I0YF0H855GT2U&field1=" + String(tempC)+"
HTTP/1.1\r\n");
delay(100);
Serial.println("AT+CIPCLOSE=0\r\n");
delay(6000);
}
```

Steps:

1. **Create new channel on ThingSpeak**



New Channel

Name: Tweet Temperature

Description: Tweet when temperature rises above 40C

Field 1: Celcius [✓]

Field 2 ☐

Field 3 ☐

Field 4 ☐

Field 5 ☐

Field 6 ☐

Field 7 ☐

Field 8 ☐

Metadata

Tags

(Tags are comma separated)

Link to External Site: http://

Link to GitHub: https://github.com/

Elevation

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

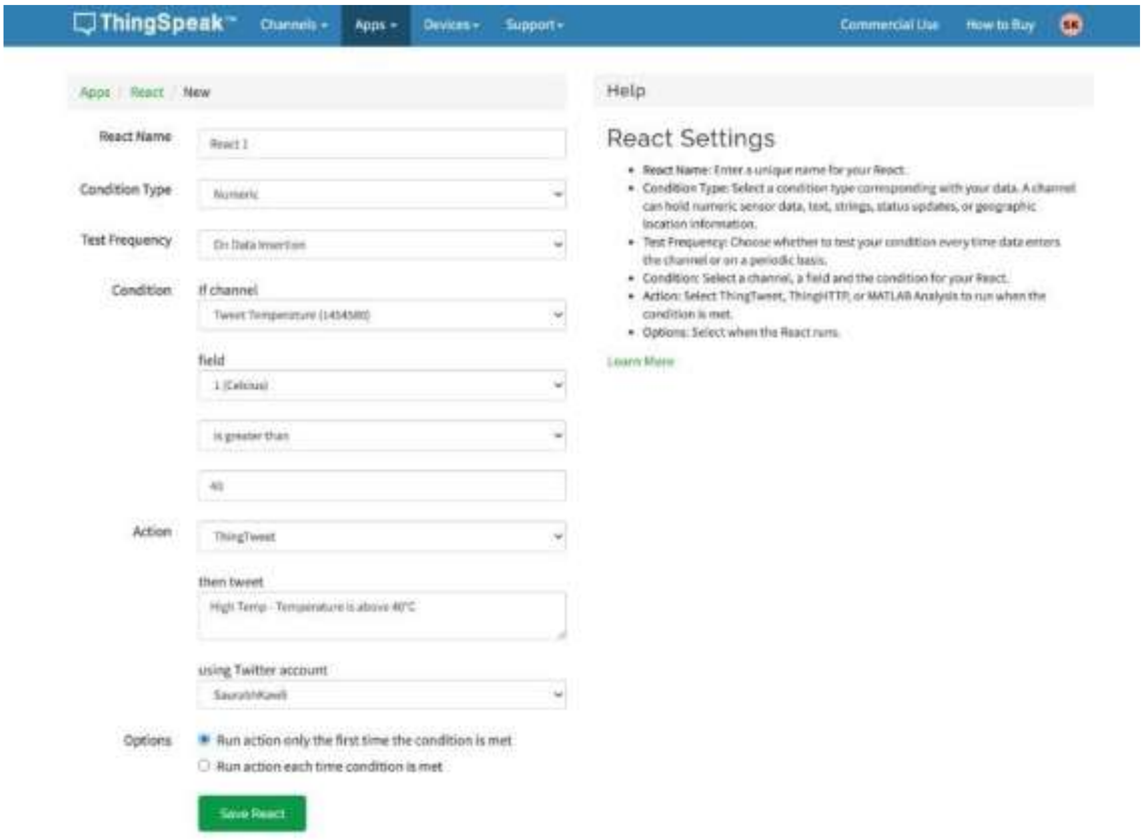
- Percentage complete: Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- Channel Name: Enter a unique name for the ThingSpeak channel.
- Description: Enter a description of the ThingSpeak channel.
- Field#: Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata: Enter information about channel data, including JSON, XML, or CSV data.
- Tags: Enter keywords that identify the channel. Separate tags with commas.
- Link to External Site: If you have a website that contains information about your ThingSpeak channel, specify the URL.
- Show Channel Location:
  - Latitude: Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - Longitude: Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - Elevation: Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- Video URL: If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- Link to GitHub: If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

Using the Channel

You can get data into a channel from a device, website, or another ThingSpeak channel. You can then visualize data and transform it using ThingSpeak Apps.

Name: Harshal Chavan Division: A
Roll No: 202124

2. Go to apps => react app => create new

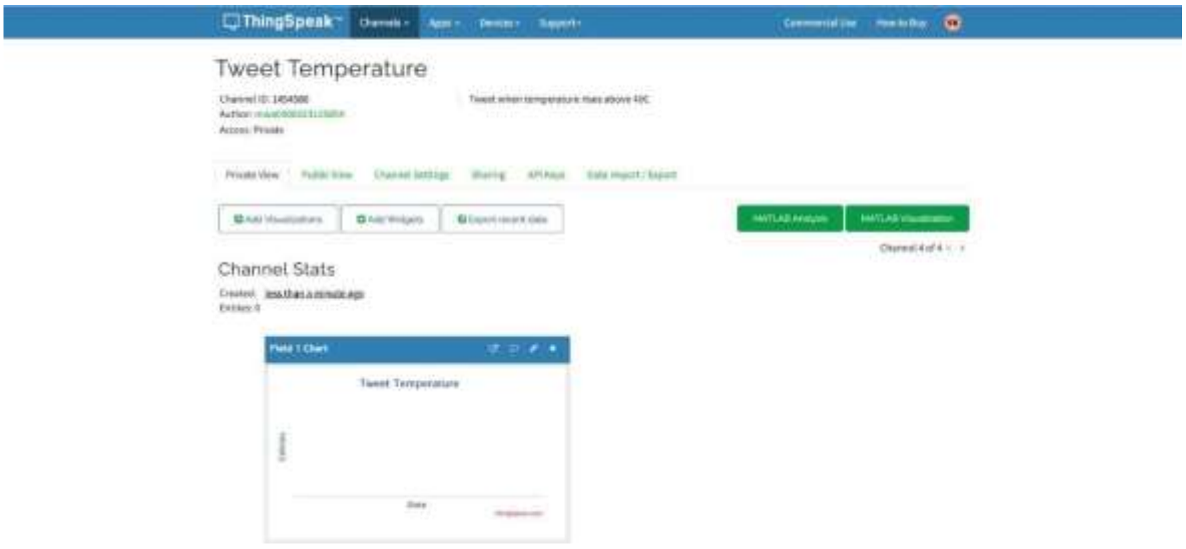


React Name: React 1
Condition Type: Numeric
Test Frequency: On Data Insertion
Condition: If channel Tweet Temperature (1454580)
field 1 (Celsius)
is greater than
40
Action: ThingTweet
then tweet High Temp - Temperature is above 40°C
using Twitter account SaurabhKawli
Options: [✓ Run action only the first time the condition is met] Run action each time condition is met

React Settings

- React Name: Enter a unique name for your React.
- Condition Type: Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- Test Frequency: Choose whether to test your condition every time data enters the channel or on a periodic basis.
- Condition: Select a channel, a field and the condition for your React.
- Action: Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- Options: Select when the React runs.

3. Tweet Temperature channel



Tweet Temperature

Channel ID: 1454580
Author: [illegible]
Access: Private

Tweet when temperature rises above 40C

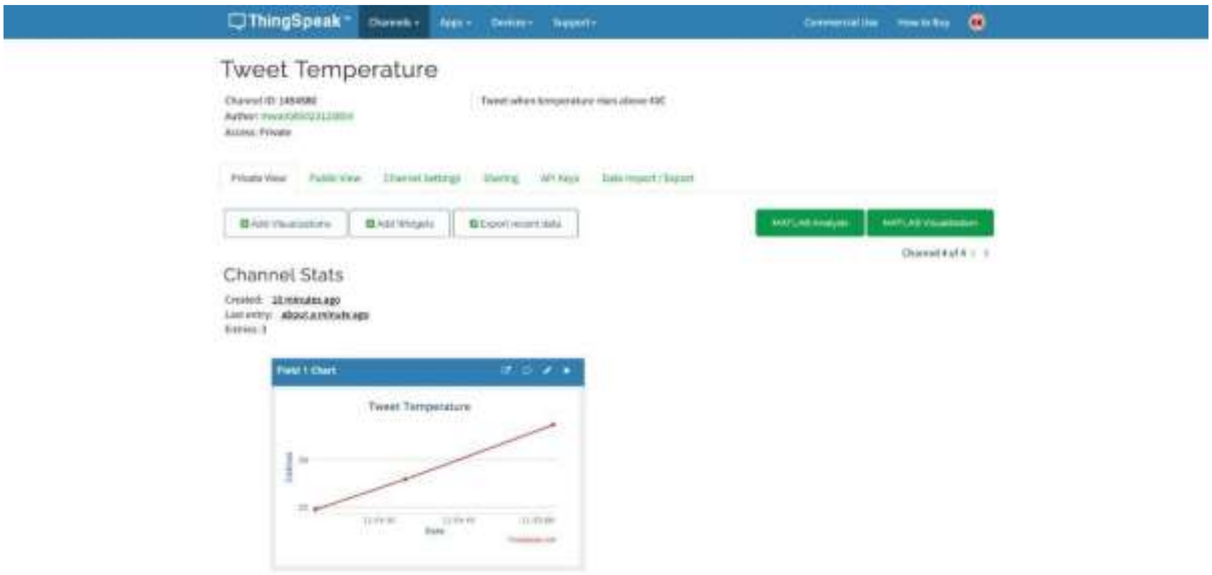
Channel 4 of 4

Channel Stats

Created: less than a minute ago
Entries: 0

Name: Harshal Chavan Division: A
Roll No: 202124

4. **Start the simulator and increase temperature above 40**
5. **Channel Stats graph**



Tweet Temperature

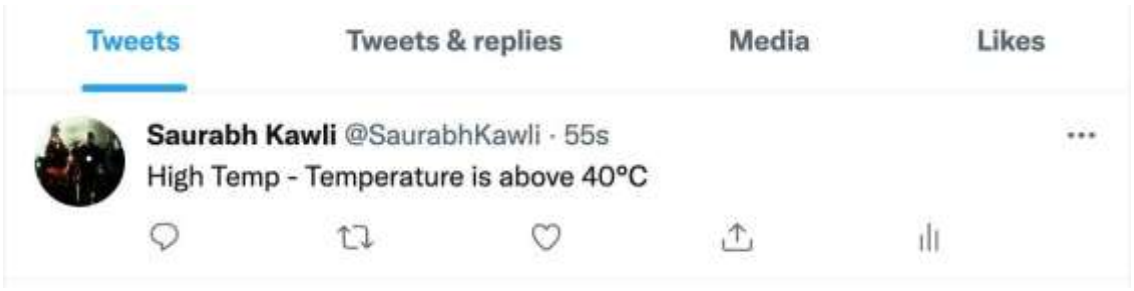
Channel ID: 1454580
Author: [illegible]
Access: Private

Tweet when temperature rises above 40C

Channel Stats

Created: 10 minutes ago
Last entry: about a minute ago
Entries: 3

6. **See the Tweet**



Saurabh Kawli @SaurabhKawli · 55s
High Temp - Temperature is above 40°C

Name: Harshal Chavan Division: A
Roll No: 202124