## Table of Contents



Practical 1:

1(a) AIM: In

Overvi

- Th

- T

# Practical 1: Introduction to Arduino Circuits

## 1(a) AIM: Introduction to Arduino Circuits and Breadboarding

### Overview of Arduino and its Capabilities

- The Arduino is a development board for the ATMEGA328 microcontroller.
- There are other useful features like input/output pins, a USB port for communication between the Arduino and a computer, and a 9V DC power connector. The different parts can be referred in the figure below.

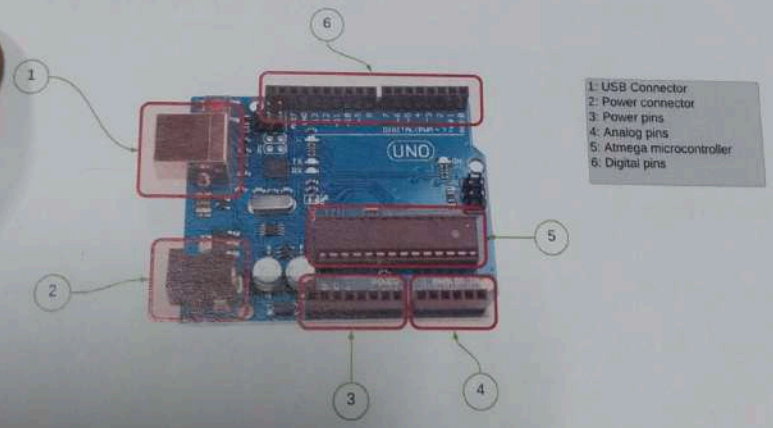



Figure 1.1: *Arduino UNO Board*

- One of the reasons the Arduino is so popular is that it is super easy to write the Arduino code. Most microcontrollers are programmed with the C language or assembly code, which is difficult to master, whereas Arduino programming language uses C/C++, which

is much easier to learn and use.

- The Arduino is a versatile electronics development platform, so you can connect a huge variety of input and output devices to it. Input devices could be a temperature sensor, magnetometer, PIR sensor, ultrasonic range finder, and many more.

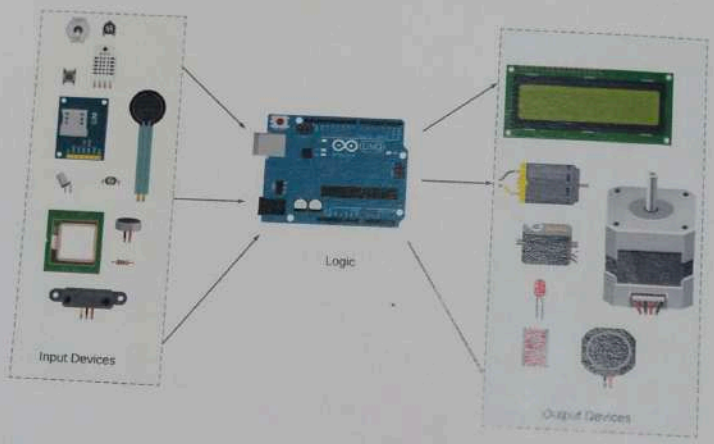



Figure 1.2: *Arduino Input Output Sensors*

## Breadboarding basics for connecting components

- The rows of a solderless breadboard are connected inside, allowing you to connect components by plugging them into the same row as each other.
- The special long rails along the edges are for easy access to power and ground.

**Note:** It's a best practice to always connect 5V and ground to these long rails as a starting point for any Arduino circuit.

1(b) **AIM: Write a Program to enable switching effect in LED using Arduino**

- **OBJECTIVE:**

  To combine basic hardware and software interactions, offering a tangible demonstration of how code can control physical components.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | Red LED |
| R1 | 1 | 220Ω |

- **CIRCUIT DIAGRAM:**

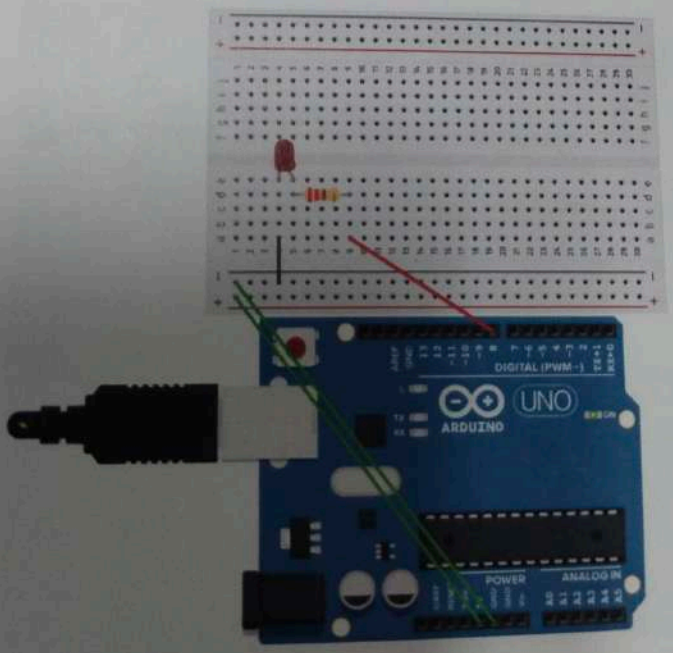



Figure 1.3: *Connection Diagram*

- **PROGRAM:**

```
int animationSpeed = 0;
void setup()
{
pinMode(8, OUTPUT);
}
void loop()
{
animationSpeed = 400;
digitalWrite(8, HIGH);
// Wait for animationSpeed millisecond(s)
delay(animationSpeed);
digitalWrite(8, LOW);
// Wait for animationSpeed millisecond(s)
delay(animationSpeed);
}
```

- **OUTPUT:**

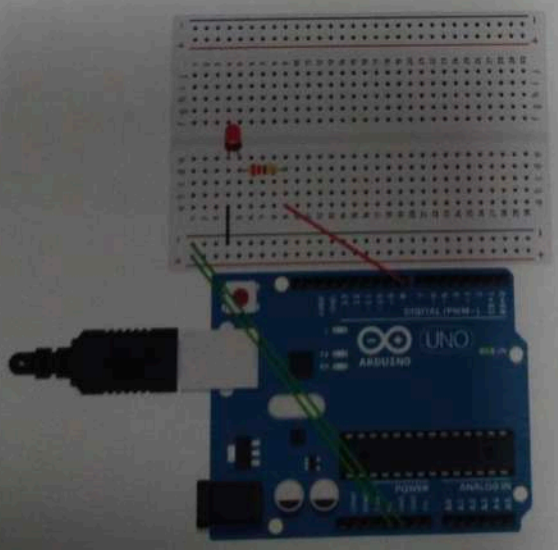



Figure 1.4: *LED in ON state*

## Practical 2: Interface Arduino with Light Sensitive Resistor

**AIM: Write a Program to interface Light Sensitive Resistor with Arduino**

- **OBJECTIVE:**

  To glow a bulb by sensing the light intensity falling on the Light Sensitive Resistor, which is also known as Light Dependent Resistor (LDR) using Arduino.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| L1 | 1 | Light Bulb |
| R1 | 1 | 1 kΩ Resistor |
| R2 | 1 | 10 kΩ Resistor |
| R3 | 1 | Photoresistor |

- **CIRCUIT DIAGRAM:**

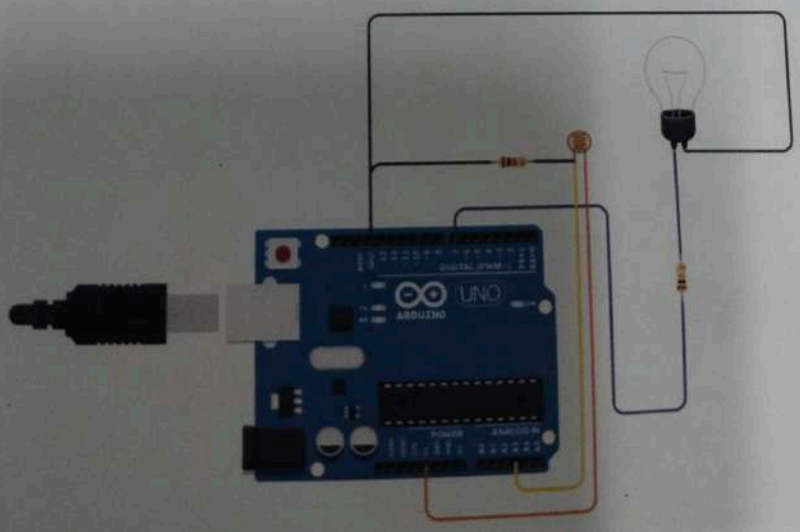



Figure 2.1: *Connection Diagram*

- **PROGRAM:**

```
int ldr = A3; //assigning ldr pin no.
int bulb = 7; //assigning bulb pin no.
void setup()
{
//setting pinmode as output
pinMode(bulb, OUTPUT);
//setting pinmode as input
pinMode(ldr, INPUT);
} void loop()
{ //reading light intensity
if ( analogRead(ldr) > 500)
digitalWrite(bulb, 0); //turn OFF condition
else
digitalWrite(bulb, 1); //turn ON condition
}
```

- **OUTPUT:**

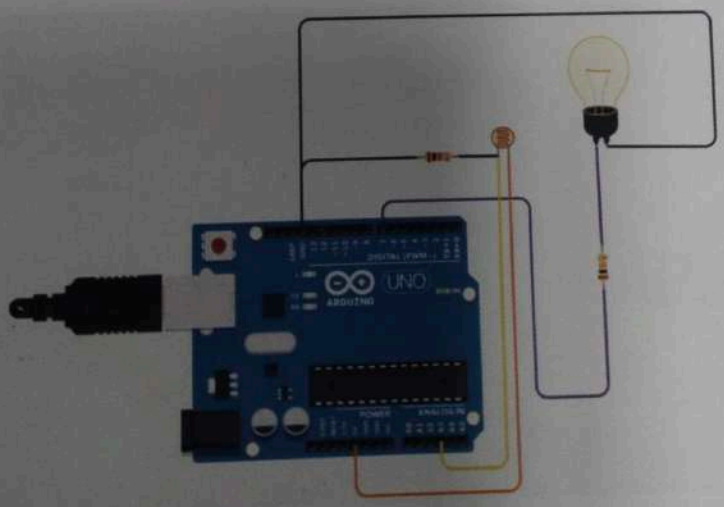


Figure 2.2: *Setup Output*

## Practical 3: Interface Arduino with a Temperature Sensor

**AIM: Write a Program to interface Temperature Sensor with Arduino**

- **OBJECTIVE:**

  To know the types of sensors used to measure the temperature of the surroundings using Arduino and display the result on Serial Monitor or any other available display units.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | Red LED |
| D2 | 1 | Yellow LED |
| D3 | 1 | Green LED |
| U2 | 1 | Temperature Sensor (TMP36) |
| R1, R2, R3 | 3 | 220Ω |

- **CIRCUIT DIAGRAM:**

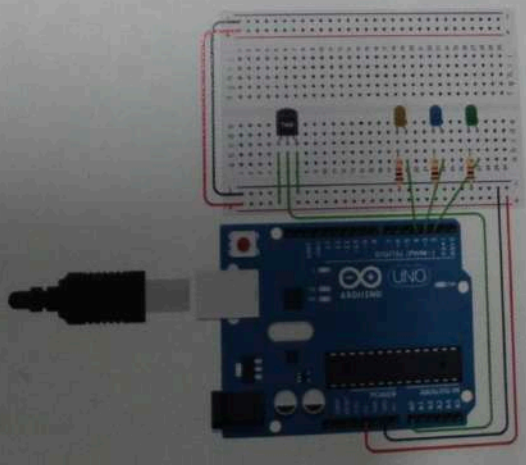


Figure 3.1: *Connection Diagram*

| | | |
|---|---|---|
| D1 | 1 | Red LED |
| D2 | 1 | Yellow LED |
| D3 | 1 | Green LED |
| U2 | 1 | Temperature Sensor (TMP36) |
| R1, R2, R3 | 3 | 220Ω |

## CUIT DIAGRAM:

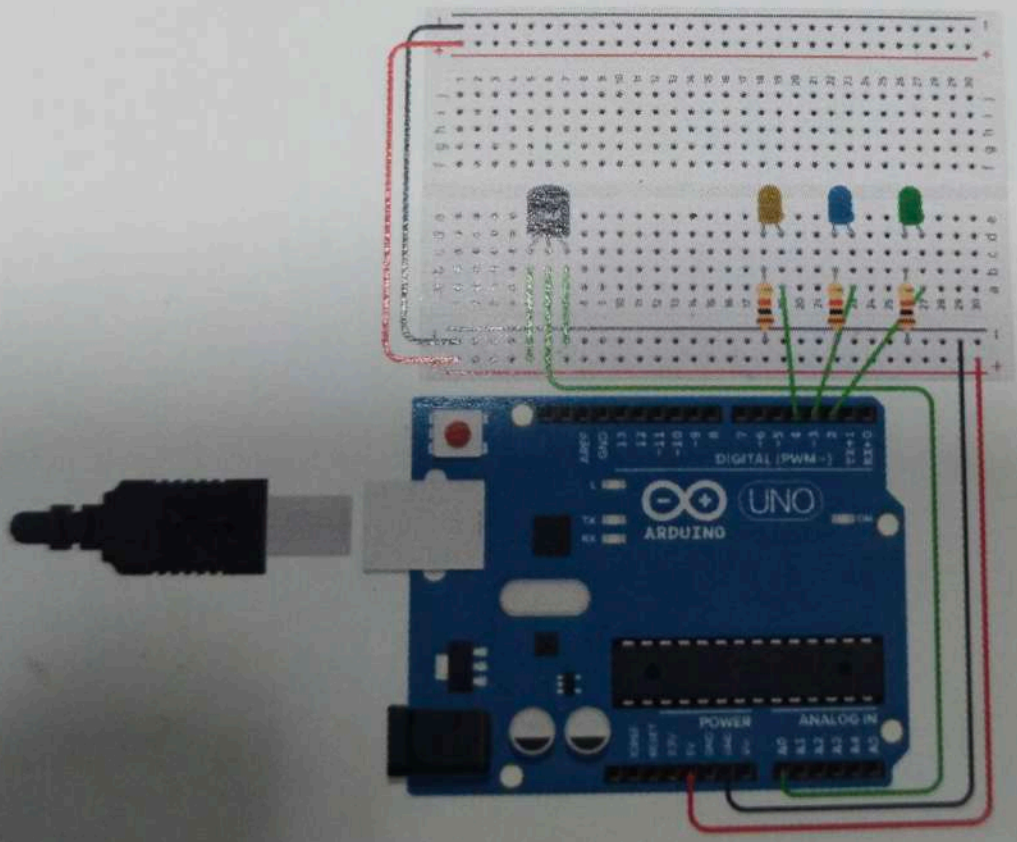



Figure 3.1: *Connection Diagram*

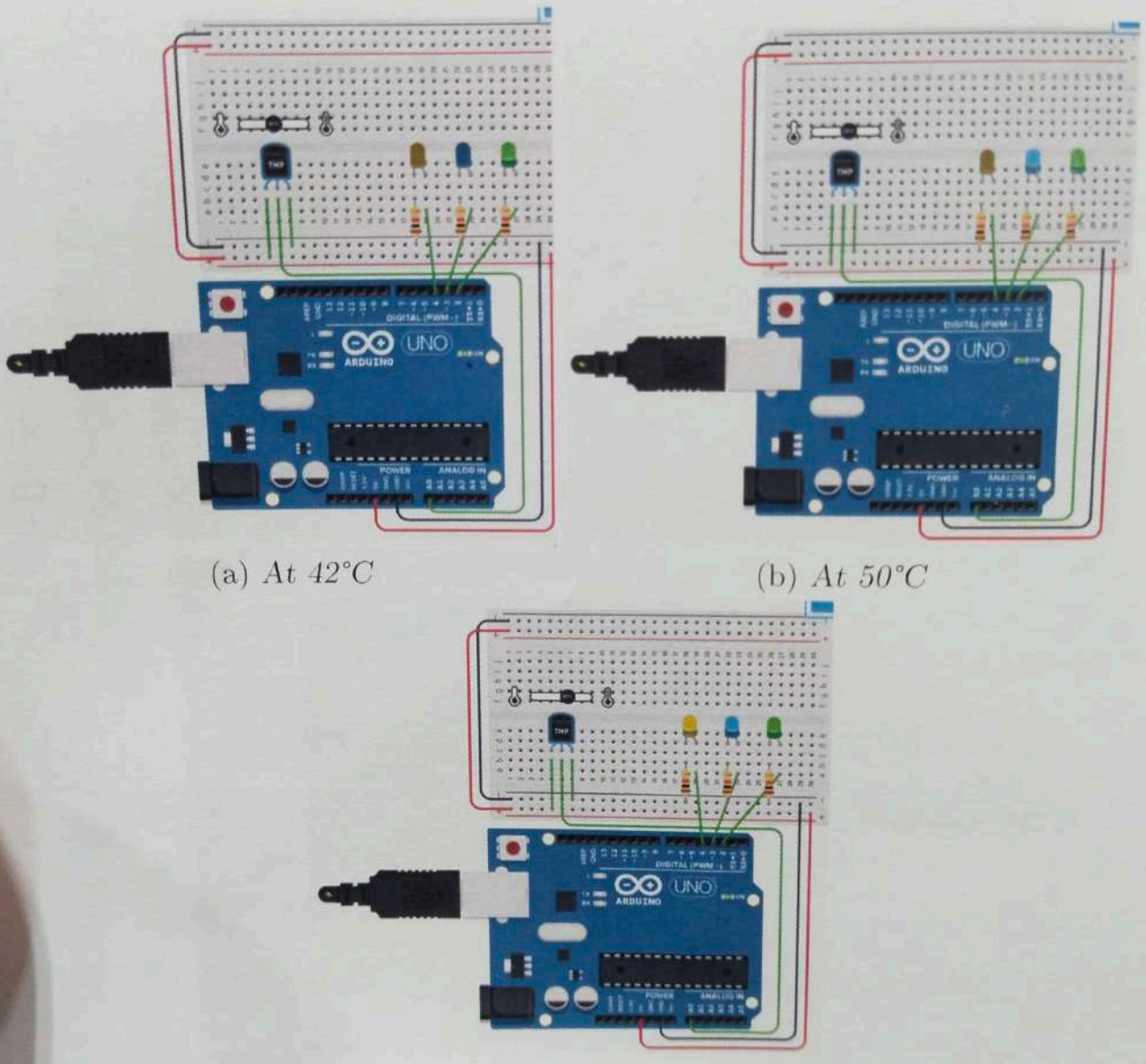
- PROGRAM:

```
int baselineTemp = 0;//Declare threshold temperature
int celsius = 0;
int fahrenheit = 0;
void setup()
{
pinMode(A0, INPUT);//Connect Sensor to Analog input
Serial.begin(9600);
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
}
void loop()
{
baselineTemp = 40;
celsius = map(((analogRead(A0) - 20) * 3.04), 0, 1023, -40,
125);
fahrenheit = ((celsius * 9) / 5 + 32);
Serial.print(celsius);
Serial.print(" C, ");
Serial.print(fahrenheit);
Serial.println(" F");
//Control LED switching based on sensed Temperature value
if (celsius < baselineTemp) {
digitalWrite(2, LOW);
```

```
digitalWrite(3, LOW);
digitalWrite(4, LOW);
}
if (celsius >= baselineTemp && celsius < baselineTemp +
10) {
digitalWrite(2, HIGH);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
}
if (celsius >= baselineTemp + 10 && celsius < baselineTemp
+ 20) {
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
}
if (celsius >= baselineTemp + 20 && celsius < baselineTemp
+ 30) {
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
}
if (celsius >= baselineTemp + 30) {
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
```

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

- OUTPUT:




(a) *At 42°C* (b) *At 50°C*

(c) *At 62°C*

Figure 3.2: *Output of the setup at various temperatures*

## Practical 4: Interface Arduino with Humidity Sensor

**AIM: Write a program to interface Arduino with Humidity Sensor**

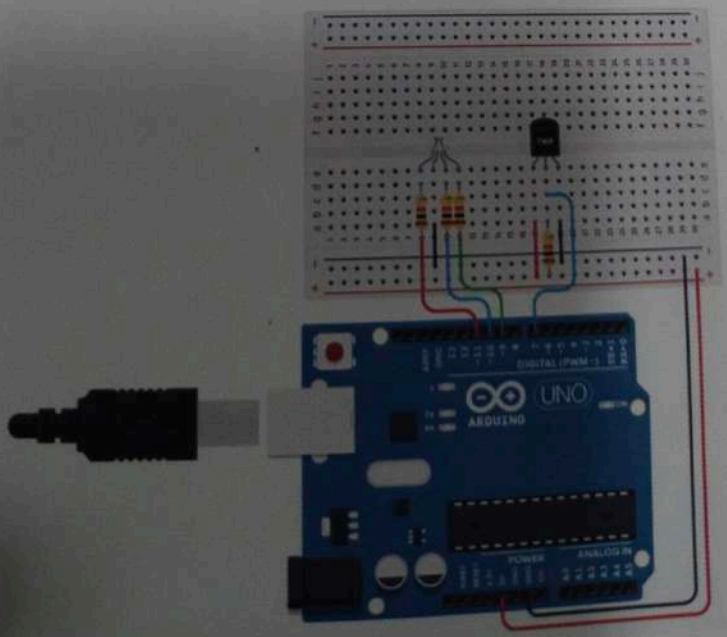
- **OBJECTIVE:**

  To know the procedure to interface Humidity sensor with arduino and provide the output on Serial monitor or any available disply. To know an alternate approach for indicating humidity control rate by using an LED.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | RGB LED |
| UDHT11 PROXY | 1 | Temperature Sensor [TMP36] |
| R1, R2, R3, R4 | 4 | 1 kΩ |

- **CIRCUIT DIAGRAM:**




Diagram

To know an alternate [illegible]

by using an LED.

- COMPONENTS REQUIRED:

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | RGB LED |
| UDHT11 PROXY | 1 | Temperature Sensor [TMP36] |
| R1, R2, R3, R4 | 4 | 1 kΩ |

- CIRCUIT DIAGRAM:

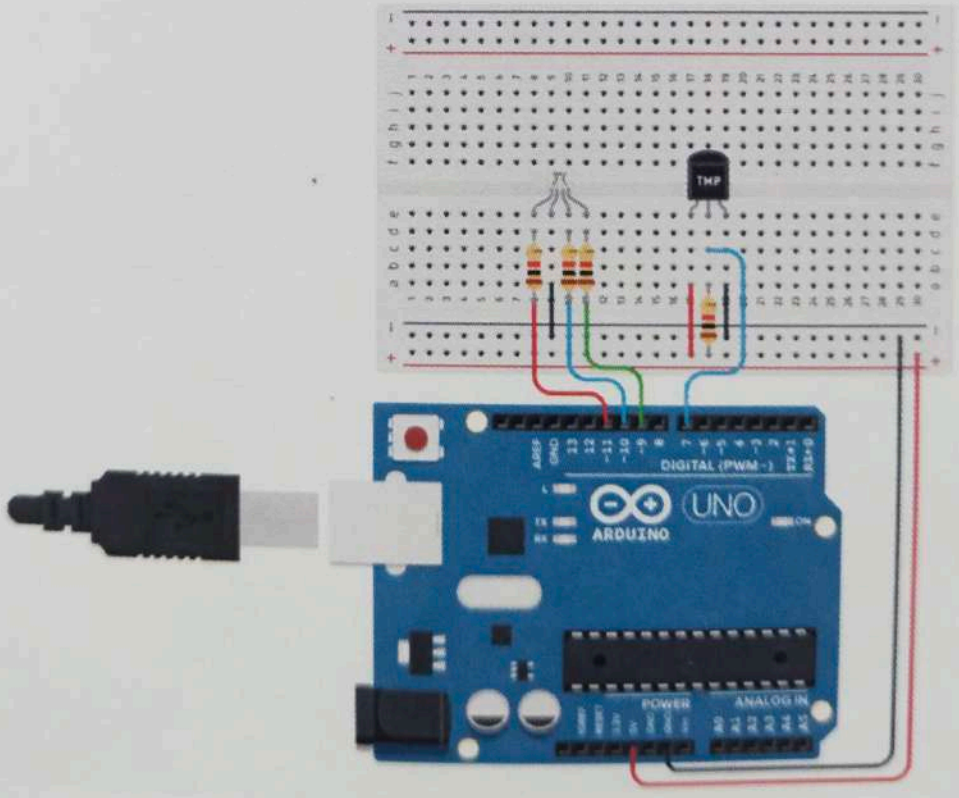



Figure 4.1: *Connection Diagram*

To know an alternate approach for indicating humidity cont

by using an LED.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | RGB LED |
| UDHT11 PROXY | 1 | Temperature Sensor [TMP36] |
| R1, R2, R3, R4 | 4 | 1 kΩ |

- **CIRCUIT DIAGRAM:**

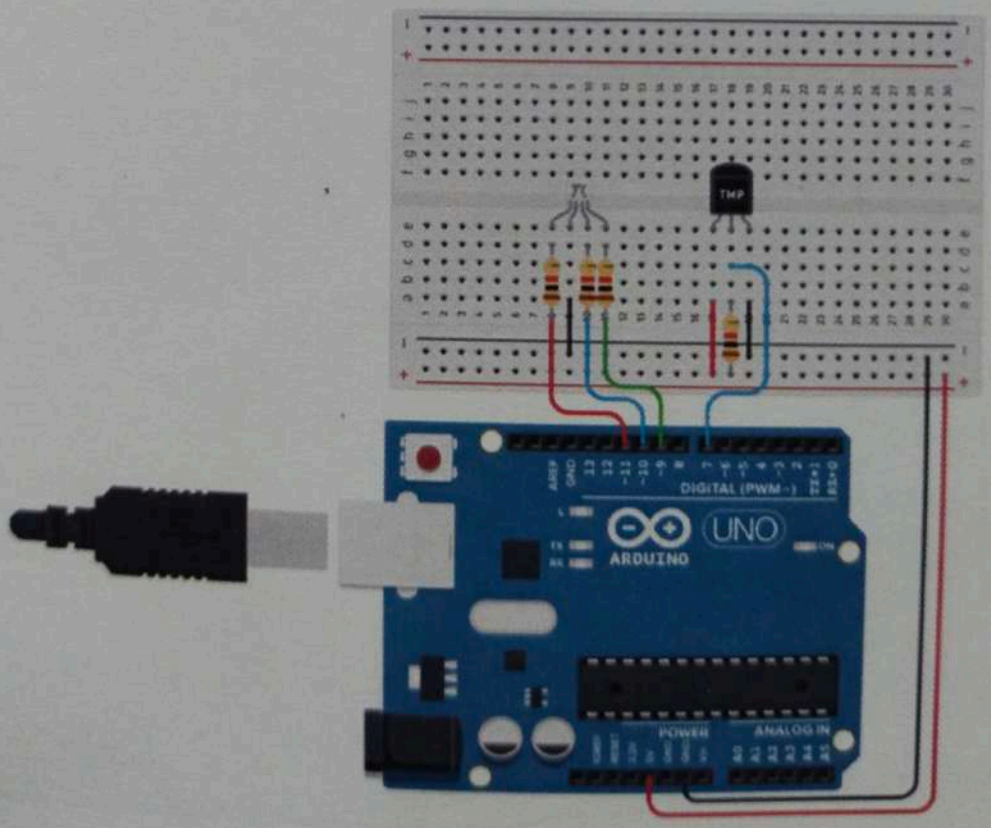



Figure 4.1: *Connection Diagram*

To know an alternate approach for indicating humidity contr

by using an LED.

- COMPONENTS REQUIRED:

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | RGB LED |
| UDHT11 PROXY | 1 | Temperature Sensor [TMP36] |
| R1, R2, R3, R4 | 4 | 1 kΩ |

- CIRCUIT DIAGRAM:

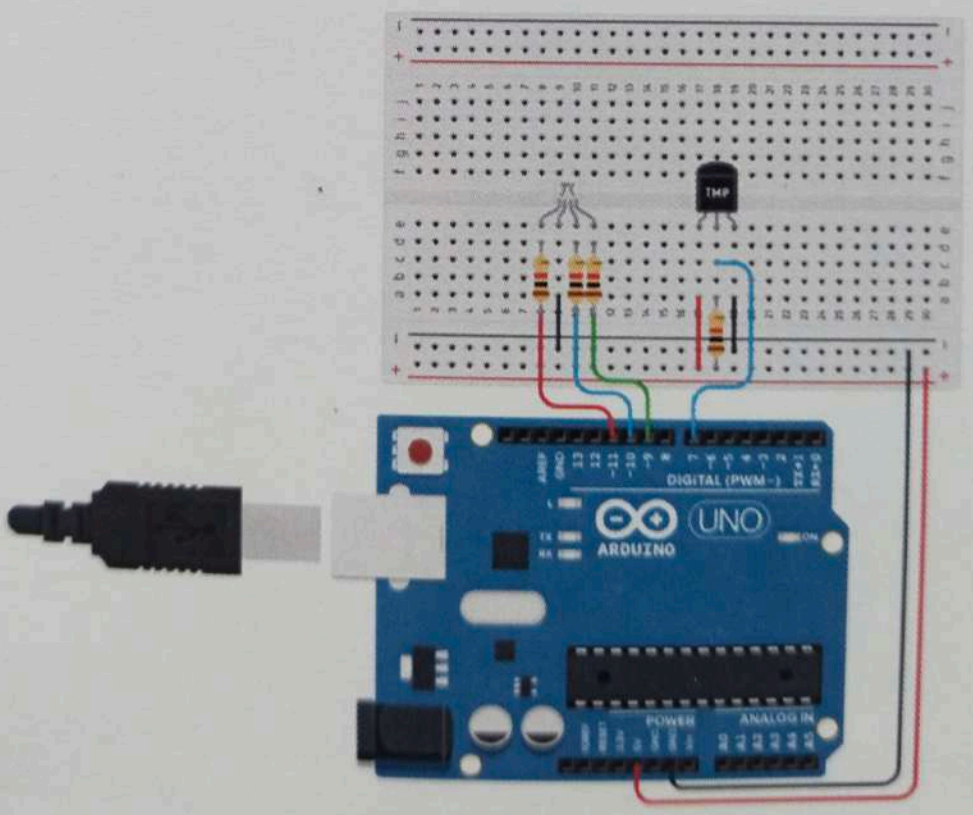



Figure 4.1: *Connection Diagram*

- PROGRAM:

```
#include <DHT.h>
#define DHTPIN 7
#define DHTTYPE DHT11
DHT dht(DHTPIN,DHTTYPE);
//Assign PIN nos.  to Tricolor LED
int ledR=11;
int ledG=9;
int ledB=10;
int map1;
int map2;
int map3;
void setup() {
Serial.begin(9600);
dht.begin();
pinMode(ledR,OUTPUT);
pinMode(ledG,OUTPUT);
pinMode(ledB,OUTPUT);
}
void loop() {
int h = dht.readHumidity();
int t = dht.readTemperature();
Serial.print("Humidity:  ");
Serial.print(h);
Serial.print(" %");
```

```
Serial.print("Temperature:  ");
Serial.print(t);
Serial.println(" °C");
if(h>0 && h<=20){
analogWrite (ledR,100);
analogWrite (ledG,0);
analogWrite (ledB,0);
}
if(h>20 && h<=30){
analogWrite (ledR,0);
analogWrite (ledG,100);
analogWrite (ledB,0);
}
if(h>30 && h<=40){
analogWrite (ledR,0);
analogWrite (ledG,0);
analogWrite (ledB,100);
}
```

- **OUTPUT:**

  The LED will glow in:

Case 1: RED color if Humidity detected is less than 20.

Case 2: GREEN color if Humidity detected is between 20 and 30.

Case 3: Blue color if Humidity detected is between 30 and 40.

# Practical 5: Programs using Line tracking sensors

**AIM: Write a Program to interface Arduino with Line tracking sensors**

- **OBJECTIVE:**

  To provide a stepping stone for students to implement a robot by using Line tracking sensors.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| BAT1 | 1 | 9V Battery |
| M1, M2 | 2 | Hobby Gearmotor |
| U2 | 1 | H-bridge Motor Driver |
| U3, U4, U5, U6, U7 | 5 | IR sensor |
| U8 | 1 | 5V Regulator [LM7805] |

- **CIRCUIT DIAGRAM:**

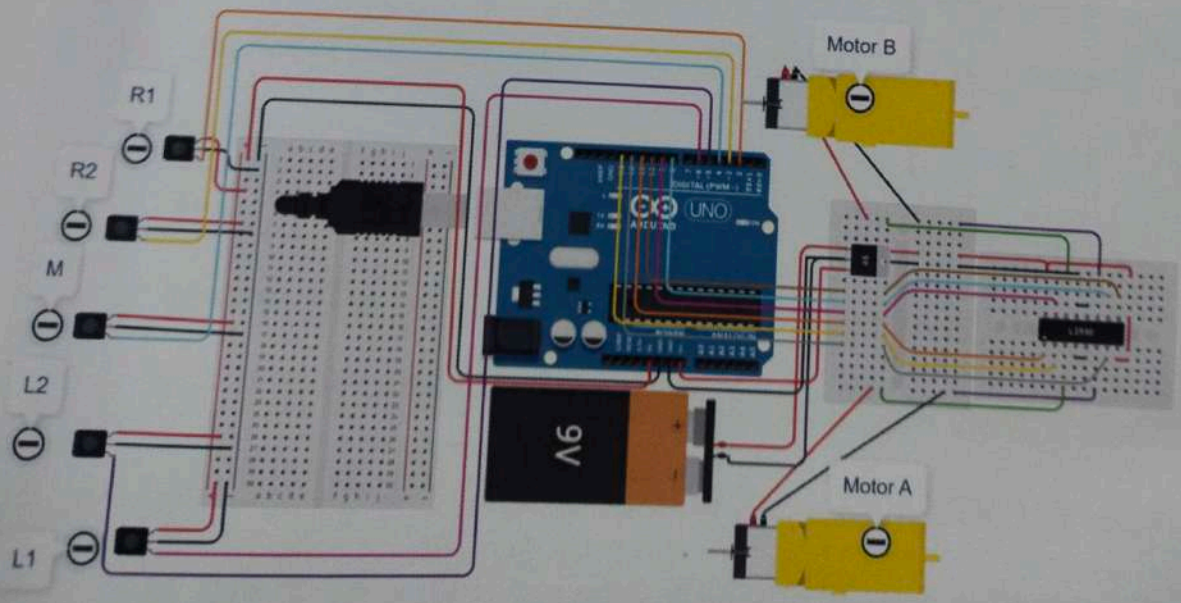



Figure 5.1: *Connection Diagram*

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| BAT1 | 1 | 9V Battery |
| M1, M2 | 2 | Hobby Gearmotor |
| U2 | 1 | H-bridge Motor Driver |
| U3, U4, U5, U6, U7 | 5 | IR sensor |
| U8 | 1 | 5V Regulator [LM7805] |

- CIRCUIT DIAGRAM:

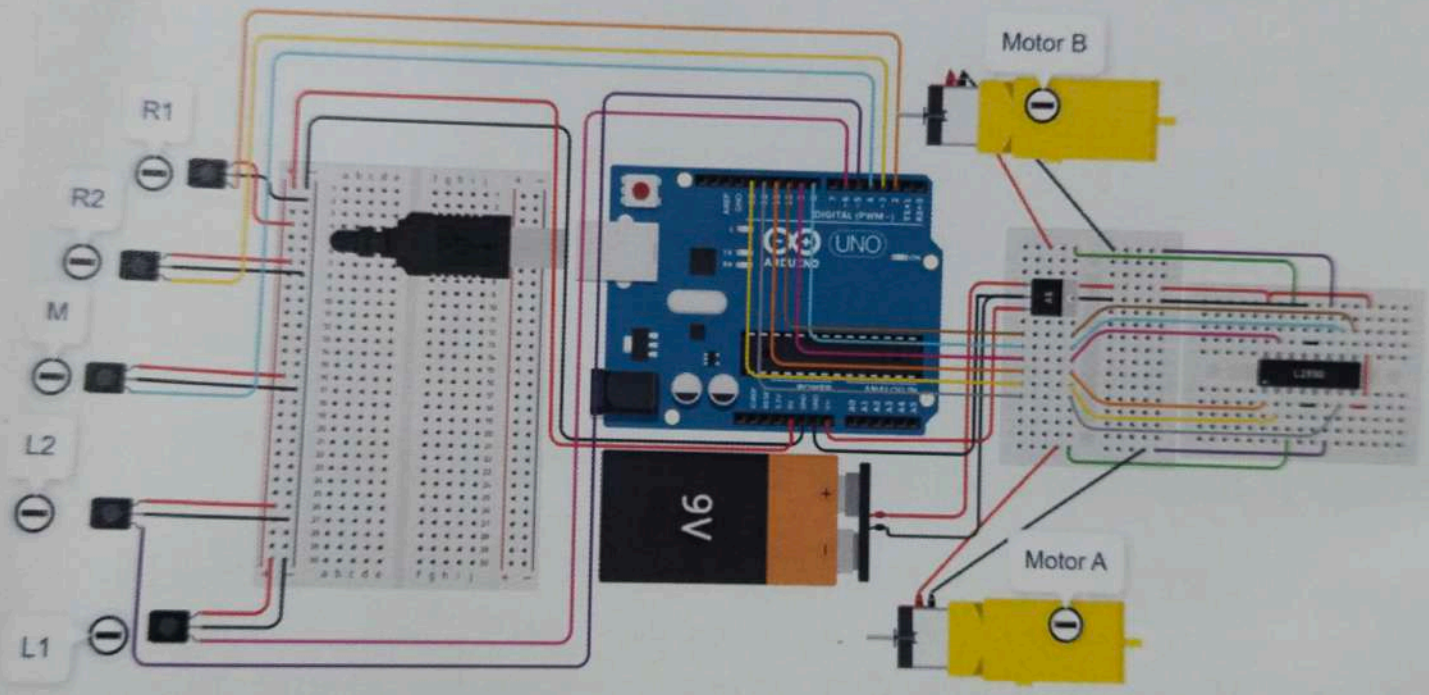



Figure 5.1: *Connection Diagram*

- PROGRAM:

```
void setup()
{
pinMode(13, OUTPUT); // Motor A +
pinMode(12, OUTPUT); // Motor A-
pinMode(11, OUTPUT); // Enable 1 & 2
pinMode(10, OUTPUT); // Enable 3 & 4
pinMode (9, OUTPUT); // Motor B -
pinMode (8, OUTPUT); // Motor B +
pinMode(6, INPUT); // L1
pinMode(5, INPUT); // L2
pinMode(4, INPUT); // M
pinMode(3, INPUT); // R2
pinMode(2, INPUT); // R1
}
void loop()
{
if((digitalRead(6)== 1) && (digitalRead(5) == 1)
&& (digitalRead(4) == 1) && (digitalRead(3)== 1)
&& (digitalRead(3) == 1));
{forward();} // calling functions
if((digitalRead(6)== 1) && (digitalRead(5) == 1)
&& (digitalRead(4) == 1) && (digitalRead(3)== 1)
&& (digitalRead(3) == 1));
{backward();}
```

```
if((digitalRead(6 )== 1) && (digitalRead(5) == 0)
&& (digitalRead(4) == 0) && (digitalRead(3)== 1)
&& (digitalRead(3) == 1));
{left_f();}
if((digitalRead(6 )== 1) && (digitalRead(5) == 0)
&& (digitalRead(4) == 1) && (digitalRead(3)== 1)
&& (digitalRead(3) == 1));
{left();}
if((digitalRead(6 )== 1) && (digitalRead(5) == 1)
&& (digitalRead(4) == 0) && (digitalRead(3)== 0)
&& (digitalRead(3) == 1));
{right_s();}
if((digitalRead(6 )== 1) && (digitalRead(5) == 1)
&& (digitalRead(4) == 0) && (digitalRead(3)== 0)
&& (digitalRead(3) == 0));
right_f();
if((digitalRead(6 )== 1) && (digitalRead(5) == 1)
&& (digitalRead(4) == 1) && (digitalRead(3)== 0)
&& (digitalRead(3) == 1));
{right();}
if((digitalRead(6 )== 0) && (digitalRead(5) == 0)
&& (digitalRead(4) == 0) && (digitalRead(3)== 0)
&& (digitalRead(3) == 0));
{stop();}
} // Declaring Functions
```

```
void forward()
{
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
digitalWrite(9, LOW);
digitalWrite(8, HIGH);
analogWrite(11, 80);
analogWrite(10, 80);
}
void backward()
{
digitalWrite(13, LOW);
digitalWrite(12, HIGH);
digitalWrite(9, LOW);
digitalWrite(8, HIGH);
analogWrite(11, 60);
analogWrite(10, 60);
}
void left_s()
{
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
digitalWrite(9, LOW);
digitalWrite(8, HIGH);
analogWrite(11, 70);
```

```
analogWrite(10, 60);
}
void left_f()
{
digitalWrite(13, LOW);
digitalWrite(12, HIGH);
digitalWrite(9, LOW);
digitalWrite(8, HIGH);
analogWrite(11, 60);
analogWrite(10, 60);
}
void left()
{
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
digitalWrite(9, LOW);
digitalWrite(8, HIGH);
analogWrite(11, 70);
analogWrite(10, 50);
}
void right_s()
{
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
digitalWrite(9, LOW);
```

```
digitalWrite(8, HIGH);
analogWrite(11, 60);
analogWrite(10, 70);
}
void right_f()
{
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
digitalWrite(9, HIGH);
digitalWrite(8, LOW);
analogWrite(11, 60);
analogWrite(10, 60);
}
void right()
{
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
digitalWrite(9, LOW);
digitalWrite(8, HIGH);
analogWrite(11, 50);
analogWrite(10, 70);
}
void stop()
{
digitalWrite(13, LOW);
```

```
digitalWrite(12, LOW);
digitalWrite(9, LOW);
digitalWrite(8, LOW);
analogWrite(11, 0);
analogWrite(10, 0);
}
```

- **OUTPUT:**

  The required output can be seen on TinkerCAD. Creating a hardware based model for the practical will be more effective for better understanding.

• CIRCUIT DIAGRAM:

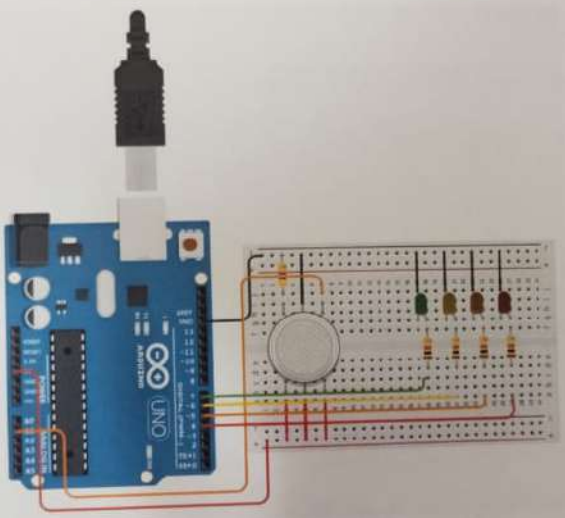


Figure 8.1: *Connection Diagram*

# Practical 6: Program Arduino using Ultrasonic Sensors

**AIM: Write a program to interface Arduino with an Ultrasonic sensor.**

- **OBJECTIVE:**

  To make students aware about the procedures for interfacing an Arduino board with an Ultrasonic sensor. This practical will provide an idea of implementing a range finding based projects.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | Red LED |
| PING1 | 1 | Ultrasonic Distance Sensor |
| R1 | 1 | 1kΩ |

- **CIRCUIT DIAGRAM:**

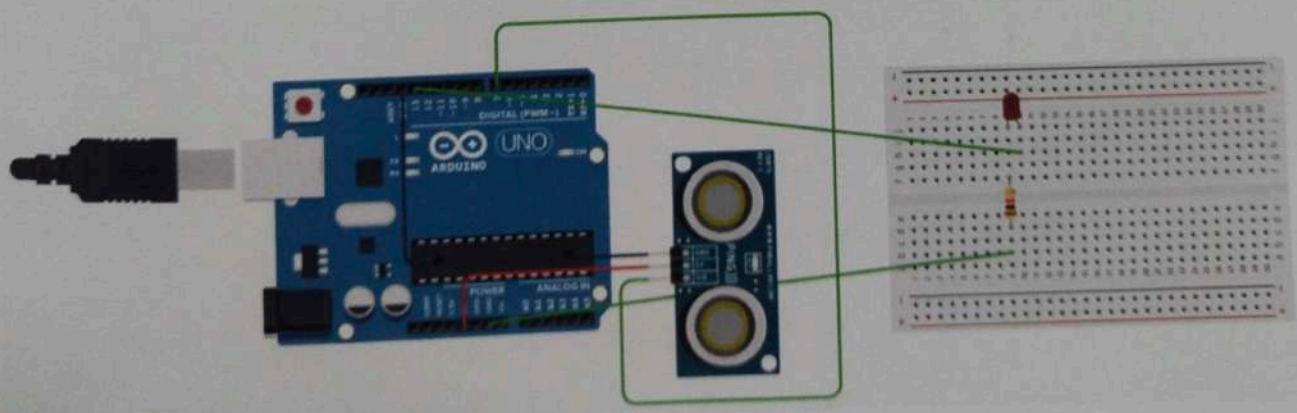



Figure 6.1: *Connection Diagram*

- **OBJECTIVE:**

  To make students aware about the procedures for interfacing an Arduino board with an Ultrasonic sensor. This practical will provide an idea of implementing a range finding based projects.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | Red LED |
| PING1 | 1 | Ultrasonic Distance Sensor |
| R1 | 1 | 1kΩ |

- **CIRCUIT DIAGRAM:**

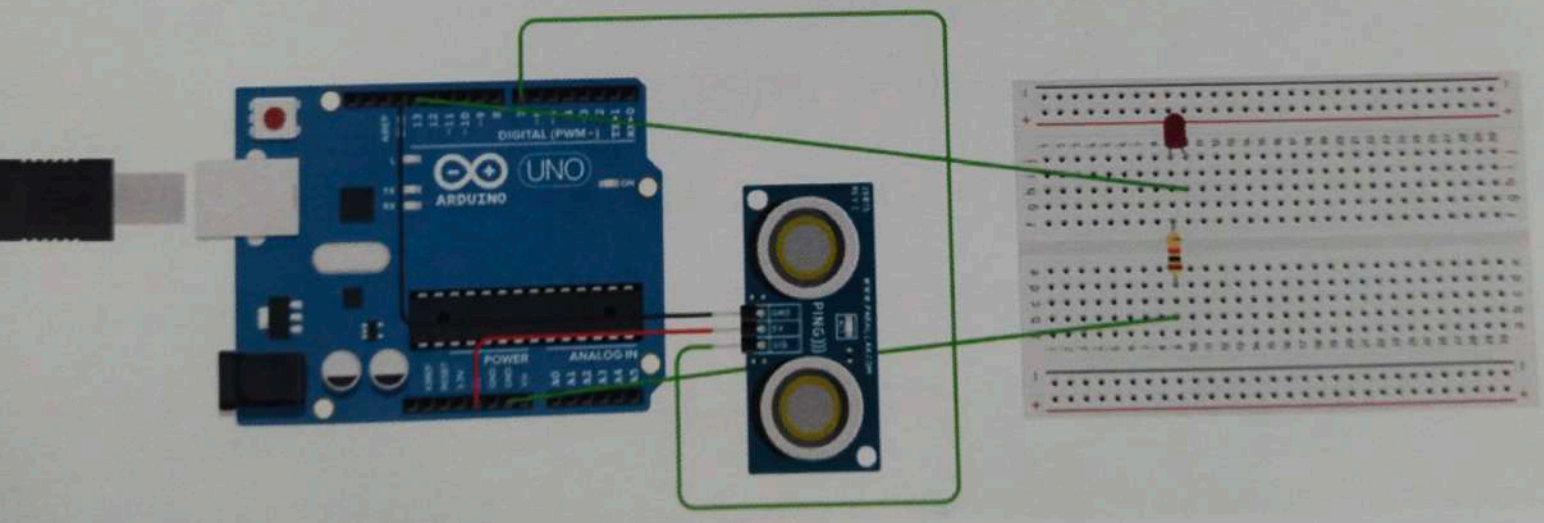



Figure 6.1: *Connection Diagram*

• PROGRAM:

```
// this constant won't change.  It's the pin number of the
sensor's output:
const int pingPin = 7;
const int ledPin = 13;
void setup() {
// initialize serial communication:
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
}
void loop() {
// establish variables for duration of the ping, and the
distance result in inches and centimeters:  long duration,
cm; The PING))) is triggered by a HIGH pulse of 2 or more
microseconds.  Give a short LOW pulse beforehand to ensure
a clean HIGH pulse:
pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(5);
digitalWrite(pingPin, LOW);
// The same pin is used to read the signal from the PING))):
//a HIGH pulse whose duration is the time (in microseconds)
//from the sending of the ping to the reception of its echo
```

• PROGRAM:

```
// This constant won't change.  It's the pin number of the
sensor's output:
const int pingPin = 7;
const int ledPin = 13;
void setup() {
// initialize serial communication:
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
}
void loop() {
// establish variables for duration of the ping, and the
distance result in inches and centimeters:  long duration,
cm; The PING))) is triggered by a HIGH pulse of 2 or more
microseconds.  Give a short LOW pulse beforehand to ensure
a clean HIGH pulse:
pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(5);
digitalWrite(pingPin, LOW);
// The same pin is used to read the signal from the PING))):
//a HIGH pulse whose duration is the time (in microseconds)
//from the sending of the ping to the reception of its echo
```

- PROGRAM:

```
// this constant won't change.  It's the pin number of the
sensor's output:
const int pingPin = 7;
const int ledPin = 13;
void setup() {
// initialize serial communication:
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
}
 void loop() {
 // establish variables for duration of the ping, and the
 distance result in inches and centimeters:  long duration,
 cm; The PING))) is triggered by a HIGH pulse of 2 or more
 microseconds.  Give a short LOW pulse beforehand to ensure
 a clean HIGH pulse:
 pinMode(pingPin, OUTPUT);
 digitalWrite(pingPin, LOW);
 delayMicroseconds(2);
 digitalWrite(pingPin, HIGH);
 delayMicroseconds(5);
 digitalWrite(pingPin, LOW);
 // The same pin is used to read the signal from the PING))):
 //a HIGH pulse whose duration is the time (in microseconds)
 //from the sending of the ping to the reception of its echo
```

```
//off of an object.
pinMode(pingPin, INPUT);
duration = pulseIn(pingPin, HIGH);
// convert the time into a distance
cm = microsecondsToCentimeters(duration);
// Print the distance
Serial.print("Distance:  ");
Serial.print(cm);
Serial.print("cm");
Serial.println();
// Turn on the LED if the object is too close:
if(cm < 100) {
digitalWrite(ledPin, HIGH);
}
else {
digitalWrite(ledPin, LOW);
}
delay(100);
}
long microsecondsToCentimeters(long microseconds) {
// The speed of sound is 340 m/s or 29 microseconds per
//centimeter.  The ping travels out and back, so to find
//the distance of the object we take half of the distance
//travelled.
return microseconds / 29 / 2;
```

}

- OUTPUT:

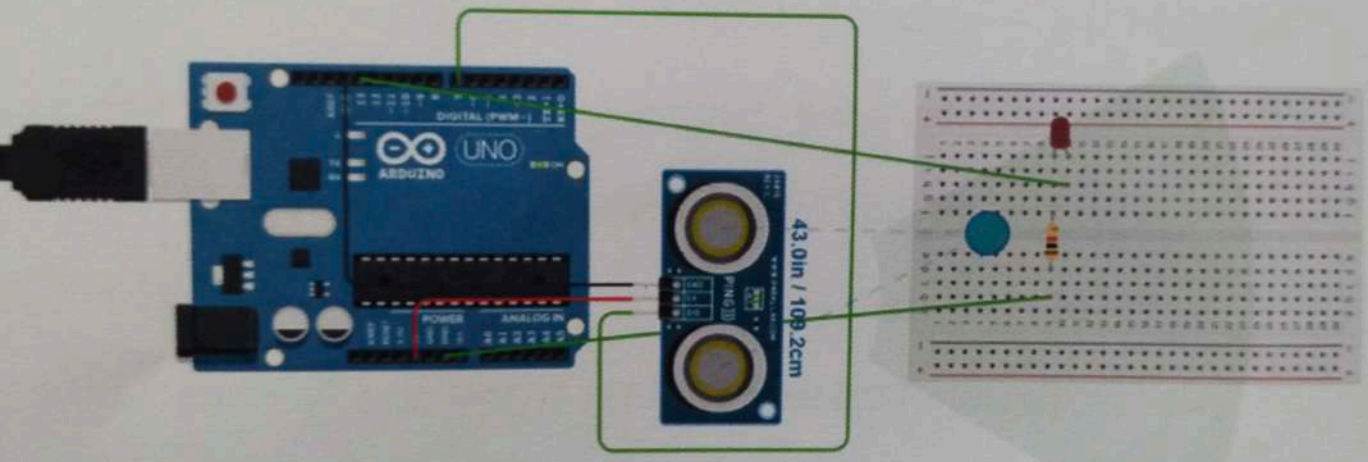



Figure 6.2: *Output during the smulation of the setup*

# Practical 7: Program Arduino using digital infrared motion sensors

**AIM: Write a program to interface Passive Infrared Sensor**

- **OBJECTIVE:**

  To make student understand the procedure used for interfacing a Digital Infrared Sensor (more specifically, PIR (Passive Infrared Sensor)). It will help students to explore various object tracking based projects.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | Red LED |
| PIR1 | 1 | PIR Sensor |
| R1 | 1 | 1kΩ |

- **CIRCUIT DIAGRAM:**

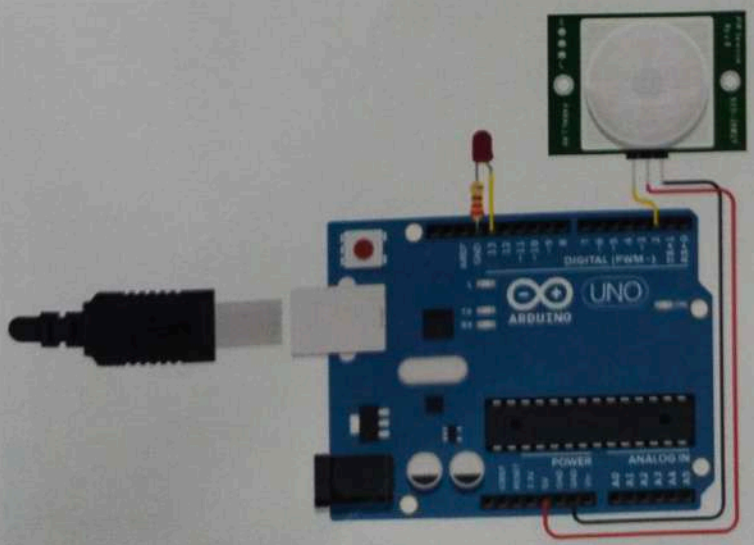



Figure 7.1: *Connection Diagram*

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | Red LED |
| PIR1 | 1 | PIR Sensor |
| R1 | 1 | 1kΩ |

# IRCUIT DIAGRAM:

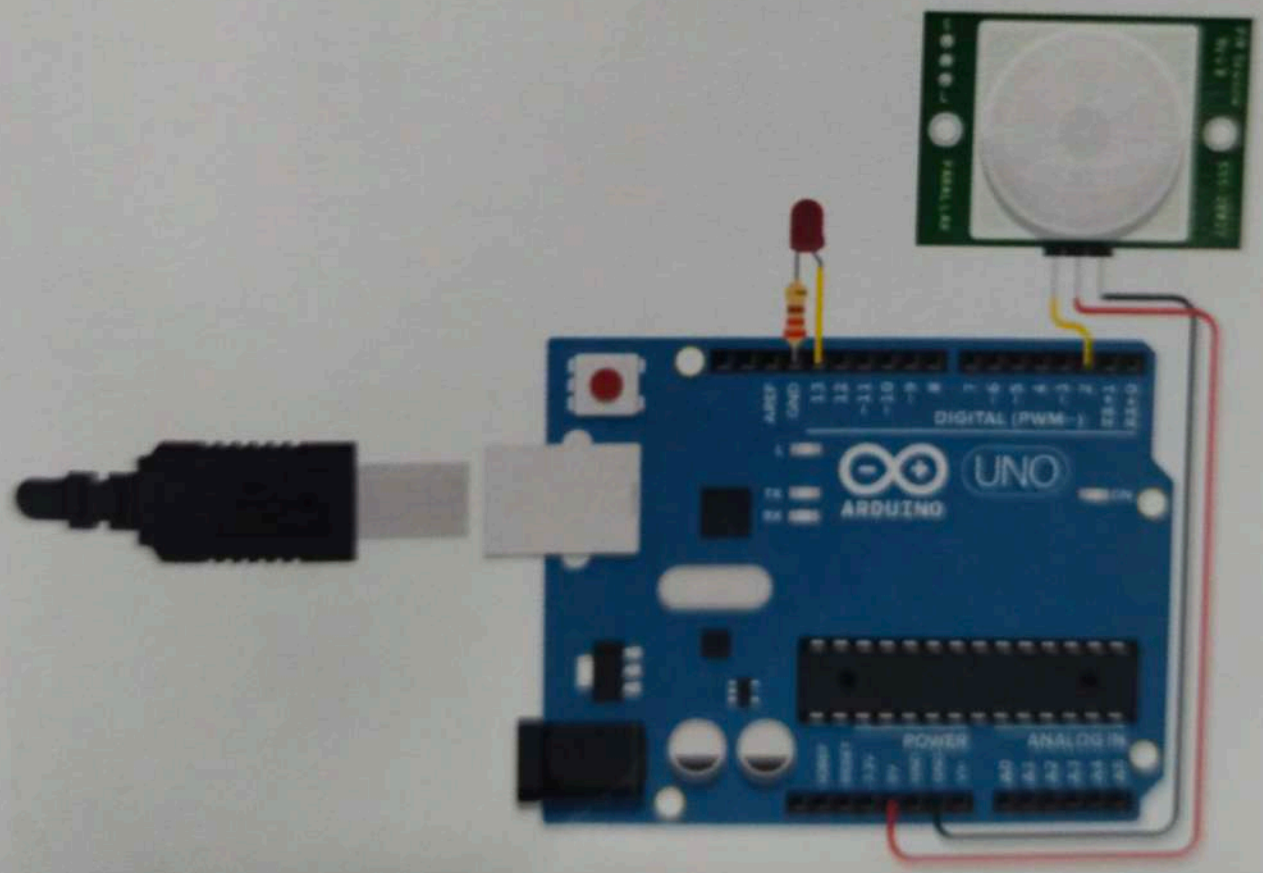



Figure 7.1: *Connection Diagram*

- **PROGRAM:**

```
int buttonState = 0;
void setup()
{
pinMode(2, INPUT); pinMode(LED_BUILTIN, OUTPUT);
}
void loop()
{
// read the state of the pushbutton
buttonState = digitalRead(2);
// check if pushbutton is pressed.  if it is, the button
//state is HIGH
if (buttonState == HIGH) {
digitalWrite(LED_BUILTIN, HIGH);
}
else {
digitalWrite(LED_BUILTIN, LOW);
}
delay(10);
//Delay a little bit to improve simulation performance
}
```

- **OUTPUT:**

The desired output can be viewed on TINKERCAD.

# Practical 8: Program Arduino using Gas Sensor

**AIM: Write a program to interface Arduino with Gas sensor.**

- **OBJECTIVE:**

  To make student understand the procedure required to interface a Gas Sensor with an Arduino. It will help students to explore various Gas leakage or smoke presence detection based projects.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | Red LED |
| D2 | 1 | Yellow LED |
| D3 | 1 | Green LED |
| D4 | 1 | Red LED |
| R2, R3, R4, R5 | 4 | 220Ω |

- **CIRCUIT DIAGRAM:**

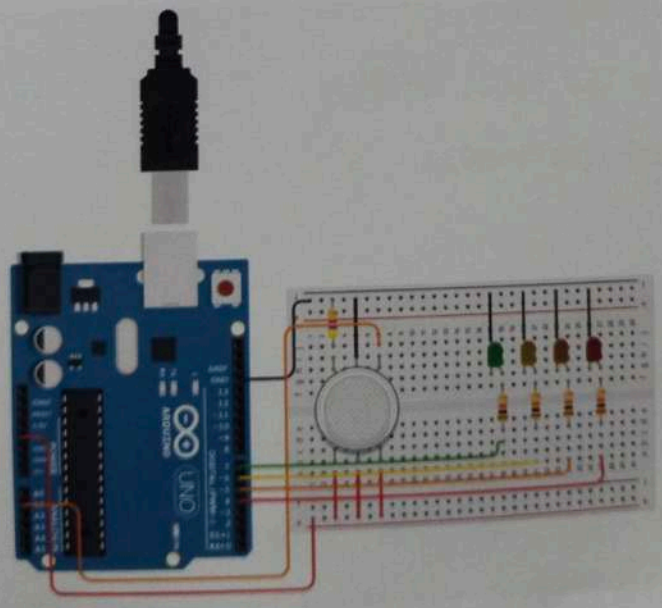


Figure 8.1: *Connection Diagram*

| | | |
|---|---|---|
| D1 | 1 | Red LED |
| D2 | 1 | Yellow LED |
| D3 | 1 | Green LED |
| D4 | 1 | Red LED |
| R2, R3, R4, R5 | 4 | 220Ω |

- CIRCUIT DIAGRAM:

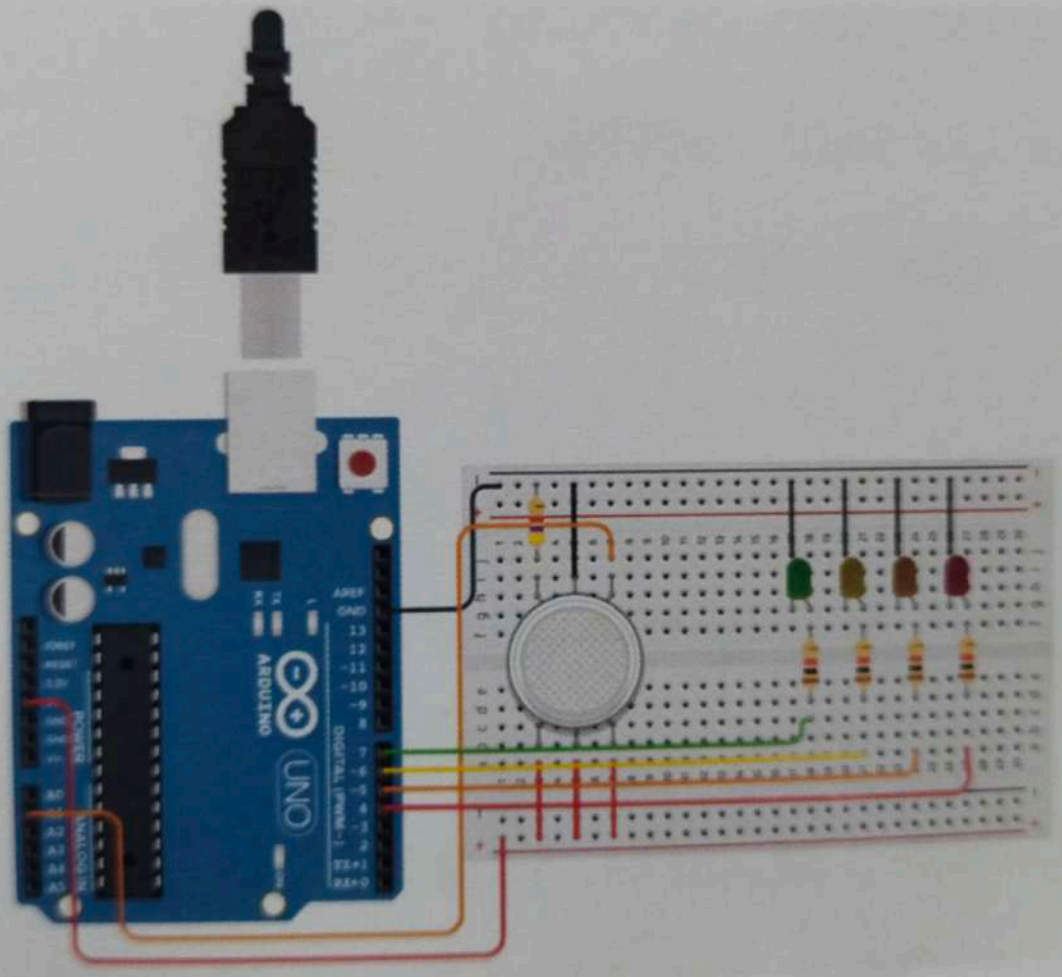



Figure 8.1: *Connection Diagram*

- PROGRAM:

```
/* ### Gas Detector ###
# Green = Safe #
# Yellow = Alert #
# Orange = Danger #
# Red = Contaminated Area #
*/
int const GAS_PIN = A1;
int LED_GREEN = 7;
int LED_YELLOW = 6;
int LED_RED1 = 5;
int LED_RED2 = 4;
void setup(){
pinMode(LED_GREEN, OUTPUT);
pinMode(LED_YELLOW, OUTPUT);
pinMode(LED_RED1, OUTPUT);
pinMode(LED_RED2, OUTPUT);
Serial.begin(9600);
}
void loop(){
int value = analogRead(GAS_PIN);
value = map(value, 300, 750, 0, 100);
digitalWrite(LED_GREEN, HIGH);
digitalWrite(LED_YELLOW, value >= 30 ?  HIGH : LOW);
digitalWrite(LED_RED1, value >= 50 ?  HIGH : LOW);
```

```
digitalWrite(LED_RED2, value >= 80 ?  HIGH : LOW);
delay(250);
}
```

- OUTPUT:

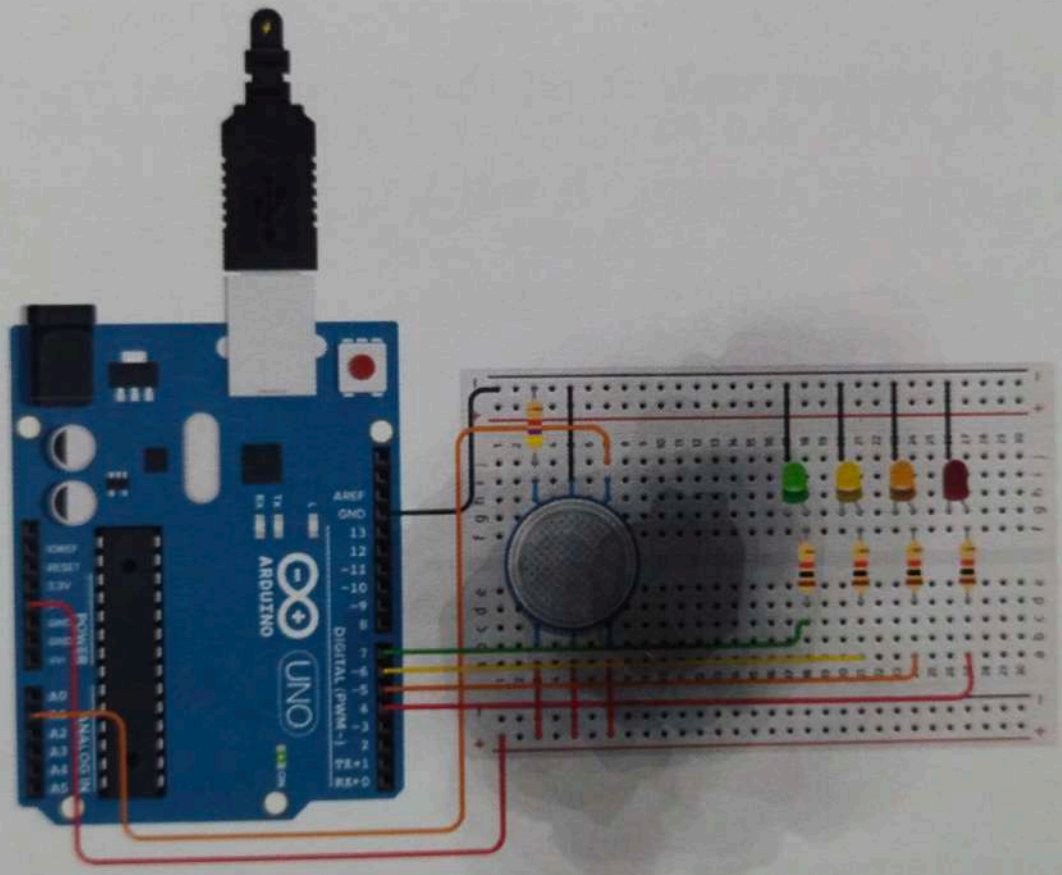



Figure 8.2: *Output during the simulation of the setup*

# Practical 9: Program Arduino using Servo Motor

**AIM: Write a program to interface an Arduino with servo motor.**

- **OBJECTIVE:**

  To make student understand the procedure to interface a Servo Motor with an Arduino. It will help students to explore various robot related automation projects.

- **COMPONENTS REQIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| SERVO1 | 1 | Positional Micro Servo |

- **COMPONENT DIAGRAM:**

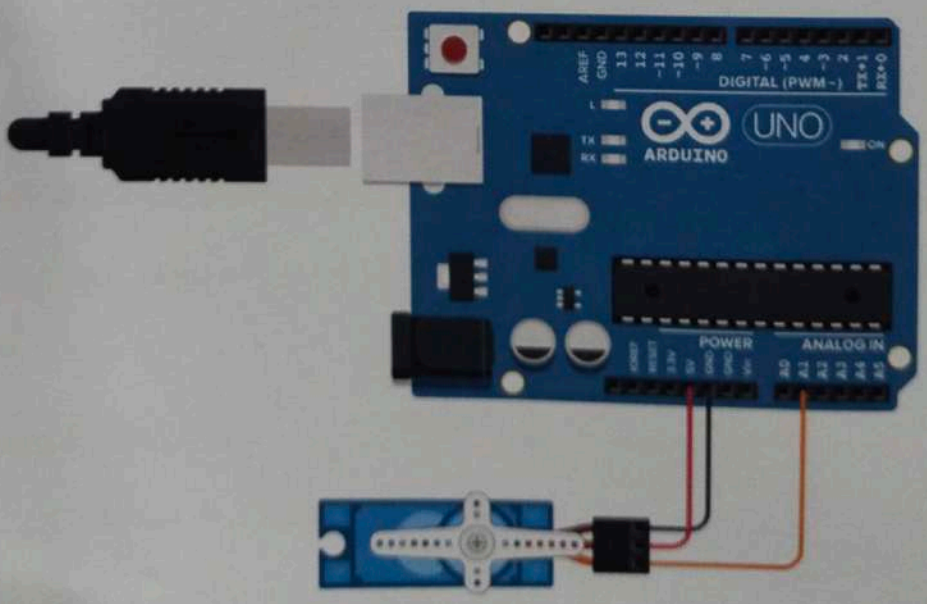



Figure 9.1: *Connection Diagram*

- PROGRAM:

```
// include the library for Servo
#include <Servo.h>
Servo servoBase;// assign a specific name
void setup() {
servoBase.attach(A1);// Pin to use for servo
servoBase.write(0);// assign 0 to the servo motor
}
void loop() {
for(int i=0; i<=180; i=i+10)
{
servoBase.write(i);
delay(2000);
}
}
```

- OUTPUT:

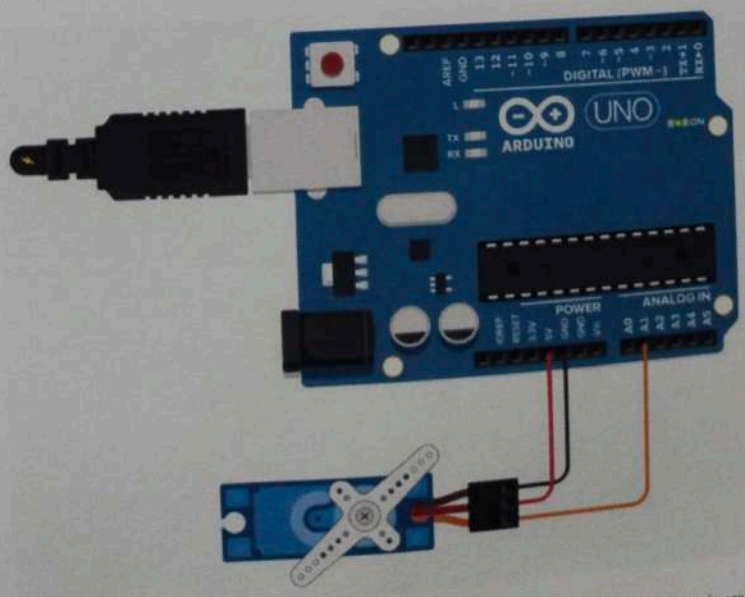



Figure 9.2: *Output during the simulation of the setup*

# Practical 10: Program to create Joystick by using an Arduino

**AIM: Write a program to design a Joystick using an Arduino Board.**

- **OBJECTIVE:**

  To make students understand the procedure to design a Joystick using Arduino. It will help students to explore various infotainment related applications related projects.

- **COMPONENTS REQUIRED:**

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 to D11 | 11 | Red LED |
| R12 | 1 | Potentiometer 10kΩ |
| R1 to R11 | 11 | 220Ω |

- **COMPONENT DIAGRAM:**

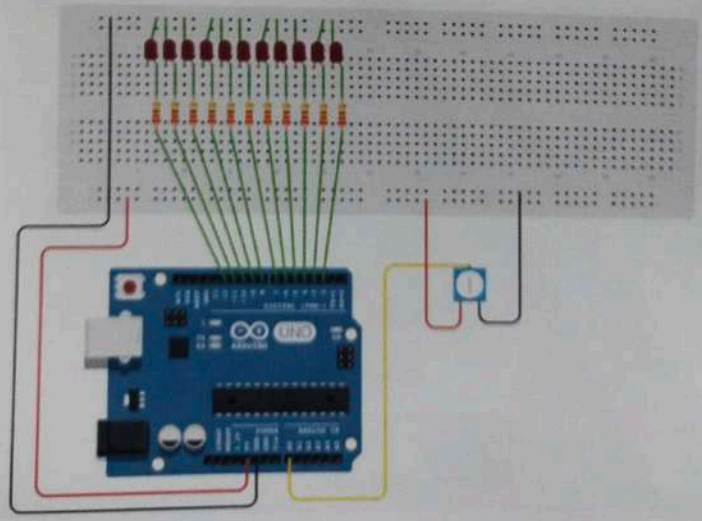


Figure 10.1: *Connection Diagram*

- COMPONENTS REQUIRED:

| Name | Quantity | Component |
|---|---|---|
| U1 | 1 | Arduino Uno R3 |
| D1 to D11 | 11 | Red LED |
| R12 | 1 | Potentiometer 10kΩ |
| R1 to R11 | 11 | 220Ω |

- COMPONENT DIAGRAM:

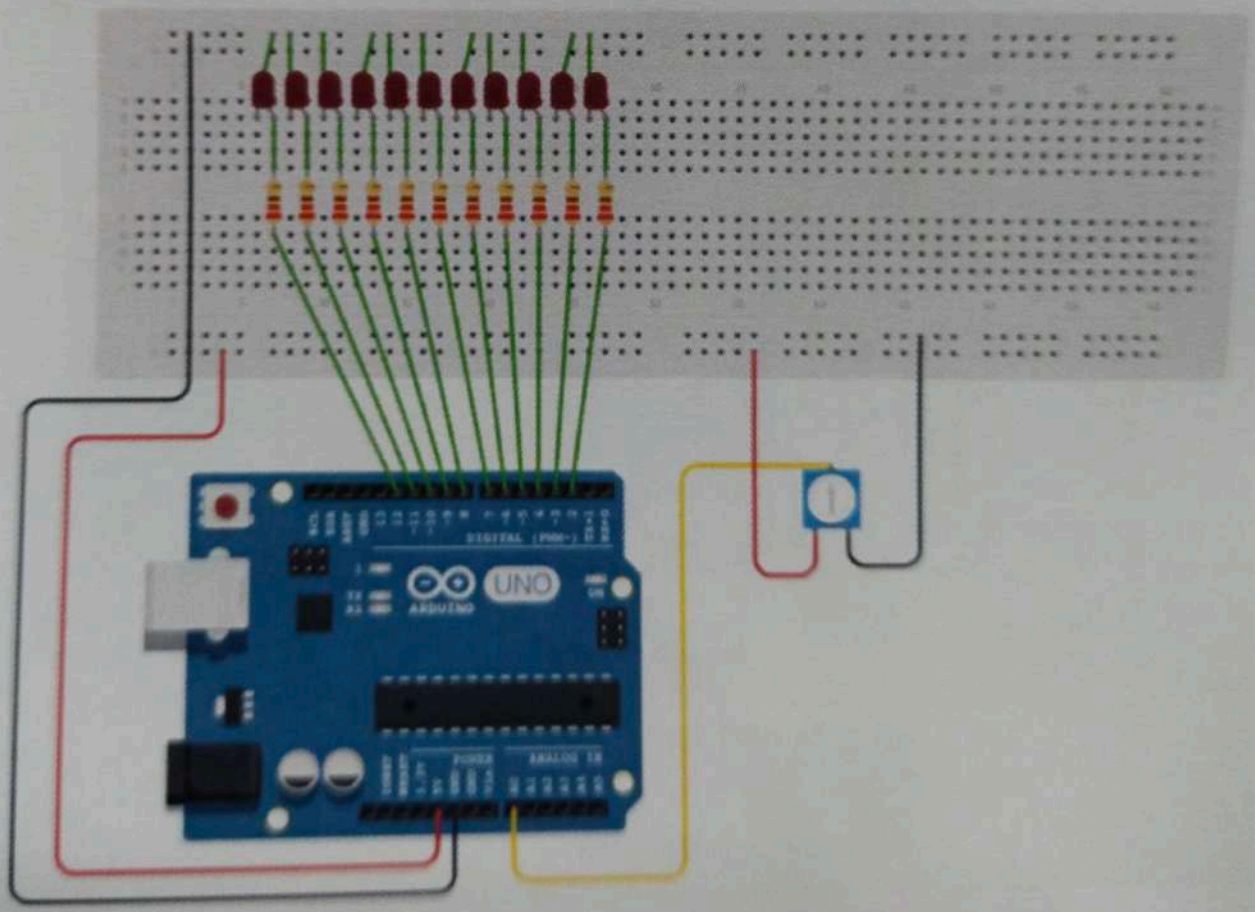



Figure 10.1: *Connection Diagram*

- PROGRAM:

```
byte led;
int joystick;
void setup() {
for(byte n=2; n<13; n++)
pinMode(n,OUTPUT);
Serial.begin(115200);
}
void loop() {
joystick= analogRead(A0); // joystick tension reading
//Serial.println(joystick);
led= map(joystick, 0 ,1023, 2, 12); // attribution of led
number
digitalWrite(led,1);
delay(10);
digitalWrite(led,0);
}
```

- OUTPUT:

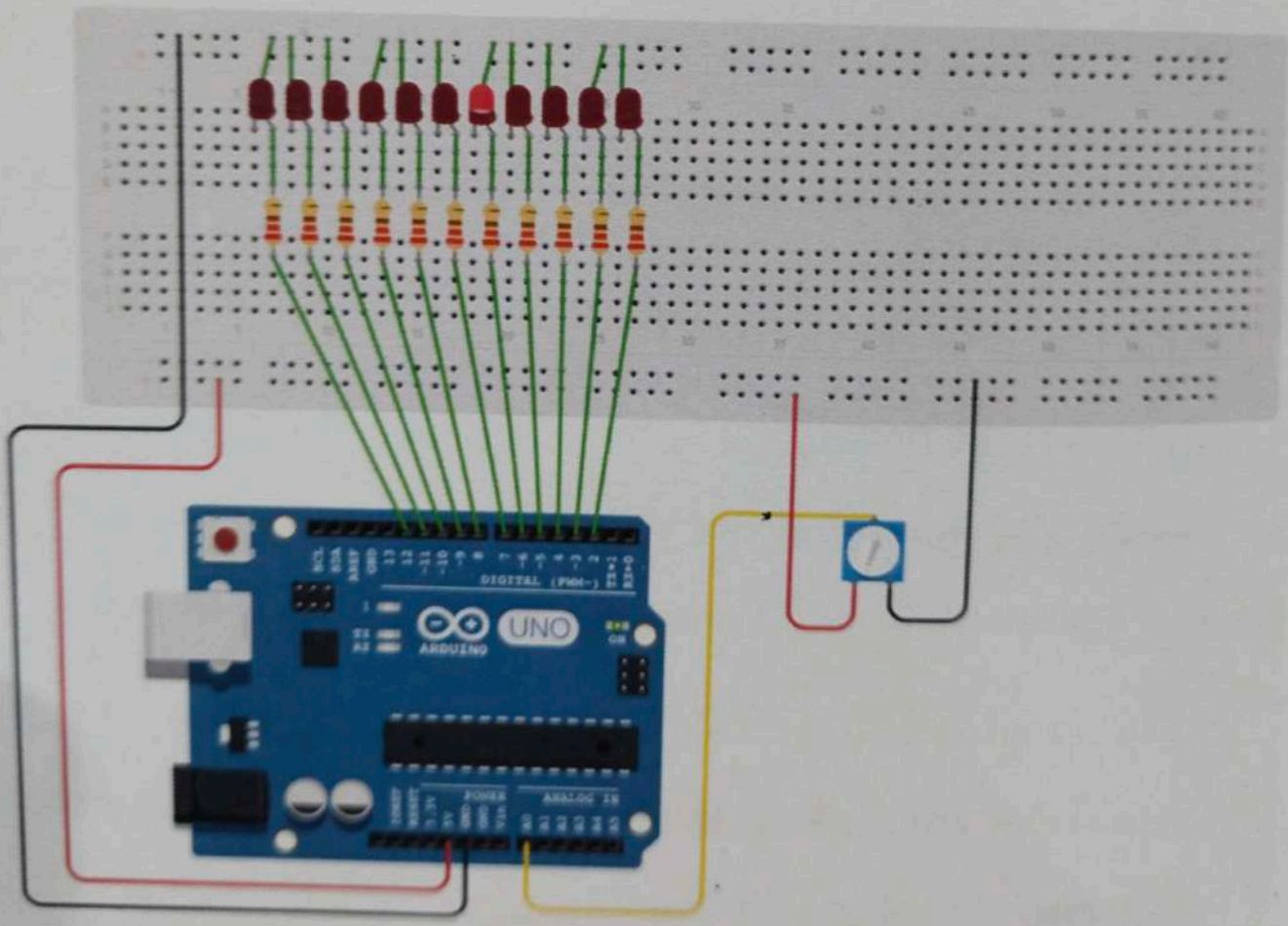



Figure 10.2: *Output during the simulation of the setup*