# Madhav Institute of Technology & Science, Gwalior

**Deemed To Be University**
**(Declared under Distinct Category by Ministry of Education, Government of India)**
**NAAC Accredited with A++ Grade**

## Department of Information Technology


# A

# Practical File

# on

# Design & Thinking Lab using Arduino (2160425)

**SUBMITTED BY**

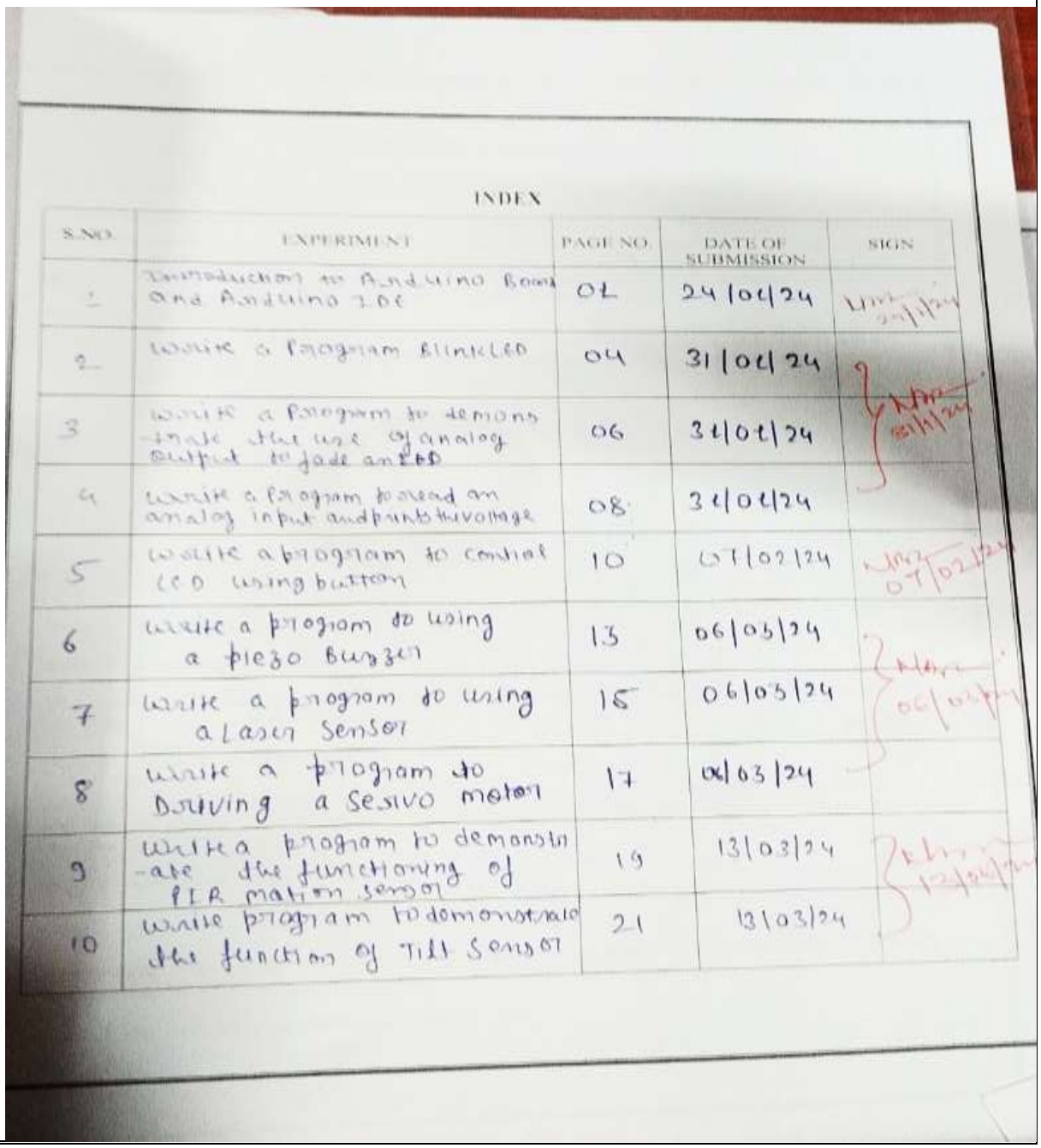
Abhishek Singh Jadoun
0901IT2333D01
IT 4TH Sem 2ND Year

**SUBMITTED TO**

**Ms. Neha Bhardwaj**

**Mr. Shubham Sharma**

**Session: January-June 2024**



INDEX

| S.NO | EXPERIMENT | PAGE NO | DATE OF SUBMISSION | SIGN |
|---|---|---|---|---|
| 1 | Introduction to Arduino Board and Arduino IDE | 02 | 24/04/24 | |
| 2 | Write a Program Blink LED | 04 | 31/04/24 | |
| 3 | Write a Program to demonstrate the use of analog output to fade an LED | 06 | 31/04/24 | |
| 4 | Write a Program to read an analog input and prints the voltage | 08 | 31/04/24 | |
| 5 | Write a program to control LED using button | 10 | 07/02/24 | |
| 6 | Write a program to using a piezo Buzzer | 13 | 06/05/24 | |
| 7 | Write a program to using a Laser Sensor | 15 | 06/05/24 | |
| 8 | Write a program to Driving a Servo motor | 17 | 06/03/24 | |
| 9 | Write a program to demonstrate the functioning of PIR motion sensor | 19 | 13/03/24 | |
| 10 | Write program to demonstrate the function of Tilt Sensor | 21 | 13/03/24 | |

# Experiment Number -01

**Aim**: Introduction to Arduino Board and Arduino IDE (Installation and Setup)

**Materials Required:** Arduino Board, Arduino IDE installed Laptop, Standard USB cable

**Introduction:** Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's designed for hobbyists, students, artists, and professionals to create interactive projects. The Arduino board serves as the brain of your project, while the Arduino Integrated Development Environment (IDE) is the software used to program the board.

## Installation and Setup process:-

**Step 1** − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug)

**Step 2** − Download Arduino IDE Software.

**Step 3 − Power up your board.**

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer

**Step 4 − Launch Arduino IDE.**

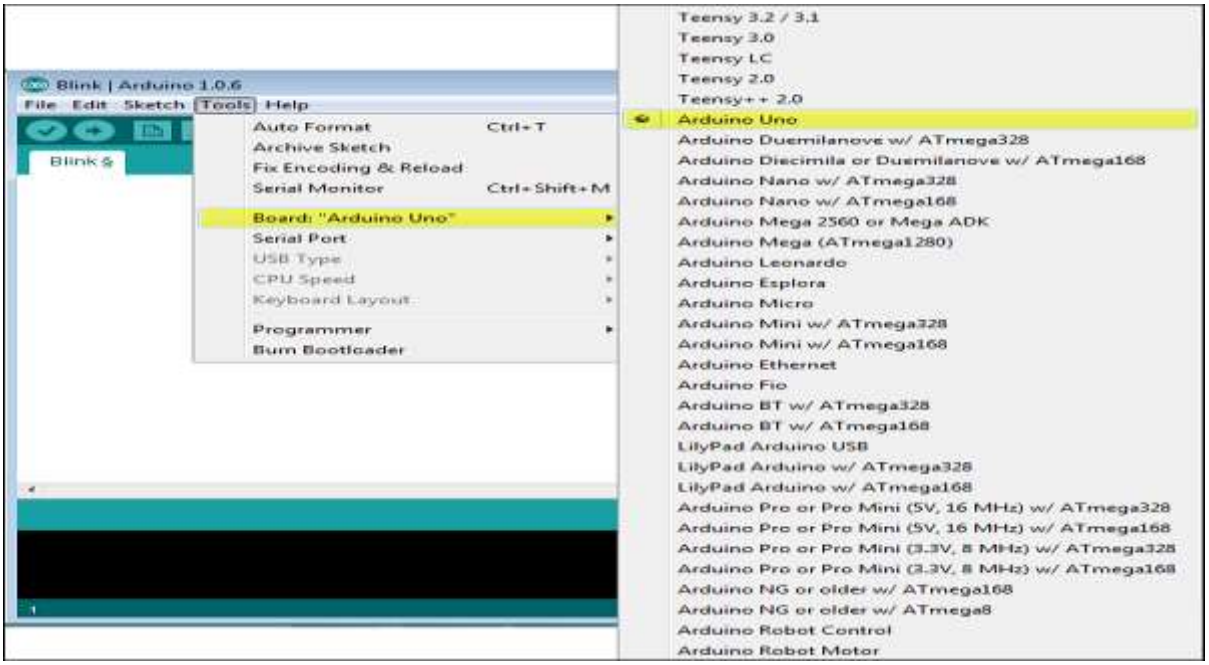
After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

**Step 5** − Open your first project**.**

**Step 6 −** Select your Arduino board**.**

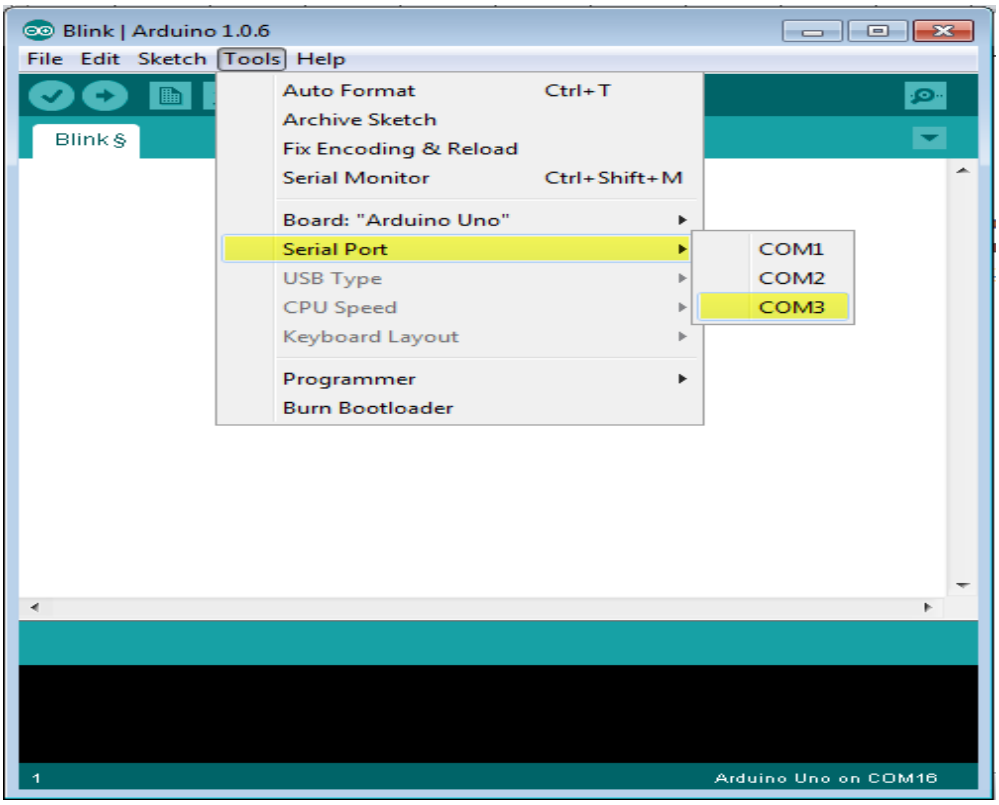
To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools → Board and select your board.




## Step 7 − Select your serial port.

Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



**Step 8 − Upload the program to your board.**

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.


**A** − Used to check if there is any compilation error.

**B** − Used to upload a program to the Arduino board.

**C** − Shortcut used to create a new sketch.

**D** − Used to directly open one of the example sketch.

**E** − Used to save your sketch.

**F** − Serial monitor used to receive serial data from the board and send the serial data to the board.

**Result: We** do the Installation and Setup of Arduino Board and Arduino IDE.

**Conclusion:** After the completion of this experiment, We are able to Installation and Setup of Arduino Board and Arduino IDE.

# Experiment Number – 02

**Aim:** Write a Program to Blink LED (Turn an LED on and off).

**Materials Required:**

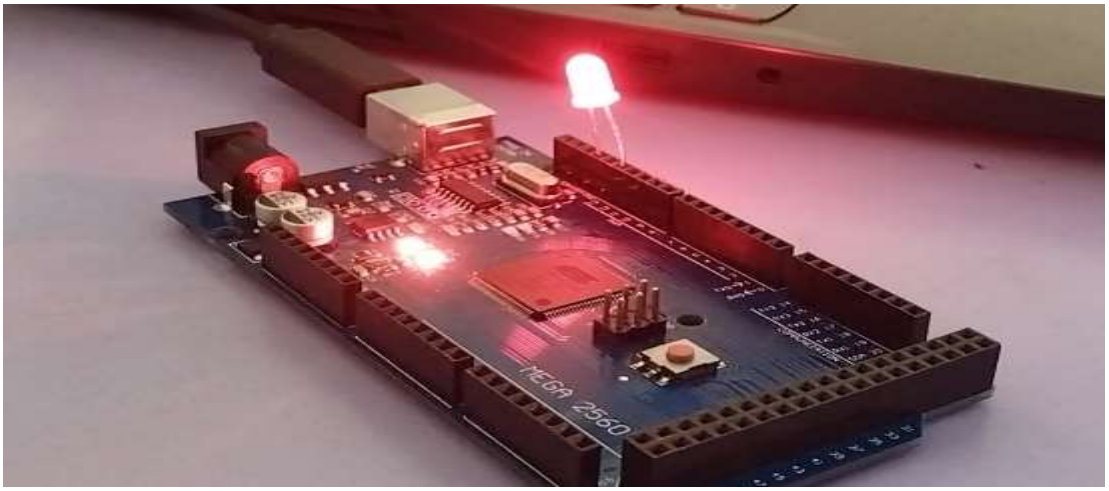
1. Arduino Mega 2560 + USB A-to-B Cable
2. **1x** Breadboard
3. **1x** LED
4. **2x** Jumper Wires

**Introduction:-** LEDs are small, powerful lights that are used in many different applications. To start off, we will work on blinking an LED, the Hello World of microcontrollers. That's right - it's as simple as turning a light on and off. It might not seem like much, but establishing this important baseline will give you a solid foundation as we work toward more complex experiments.

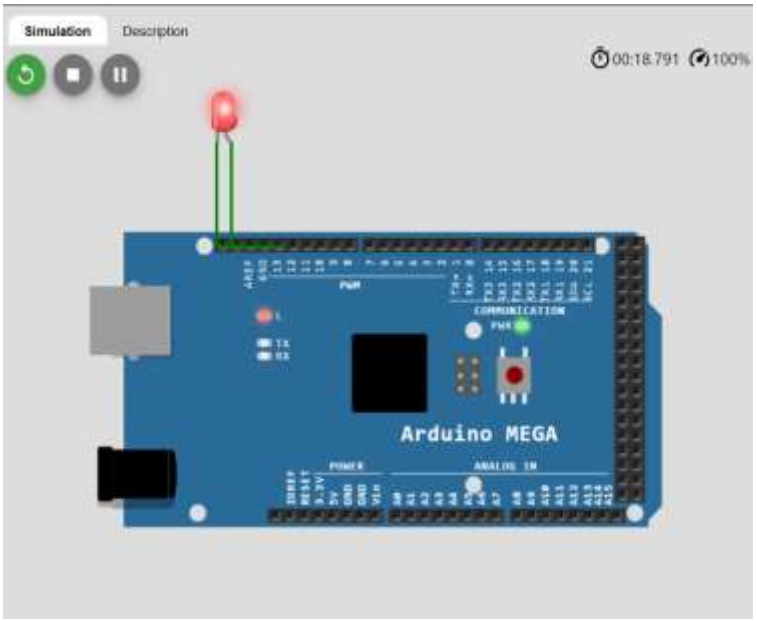
## Code:

```
void setup()
{
  pinMode(13, OUTPUT);
}
void loop()
{
  digitalWrite(13, HIGH);   // Turn on the LED
  delay(1000);              // Wait for one second
  digitalWrite(13, LOW);    // Turn off the LED
  delay(1000);              // Wait for one second
}
```

## Connection Diagram:




## Schematic Diagram:




**Result:** We constructed a Blink LED (Turn an LED on and off )

**Conclusion:** After the completion of this experiment, we were able to use delay(), digitalWrite() functions and got a better understanding of Arduino and the various implementations of an LED.

# Experiment Number - 03

**Aim:** Write a Program to demonstrate the use of analog output to fade an LED.

**Materials Required:** Arduino Board(Mega 2560), Arduino IDE installed Laptop, USB cable, LED.

**Introduction:** In order to change the brightness of an LED which makes it fade, we use a special function called "analogWrite()".
Functions of Arduino used

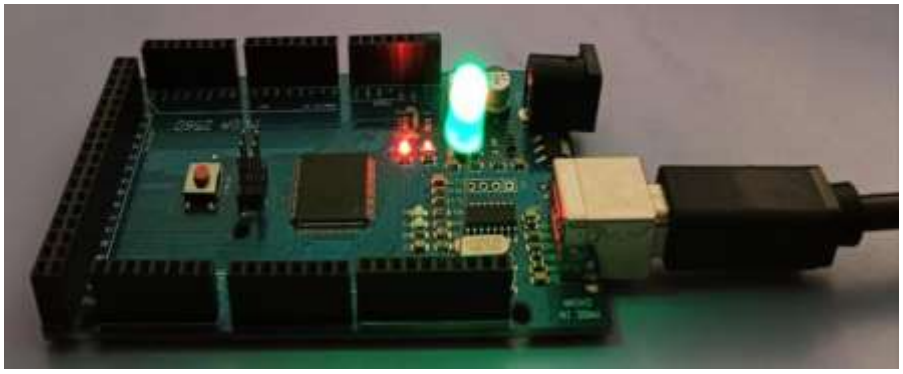
a) pinMode(pinNumber,Mode) : Used to determine usage of pin whether for Input or for Output
b) analogWrite(pinNumber,Signal) : Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds.
c) delay(time(ms)) : Used to create a time gap for CPU by passing certain amount of time to halt the execution.

## Code:

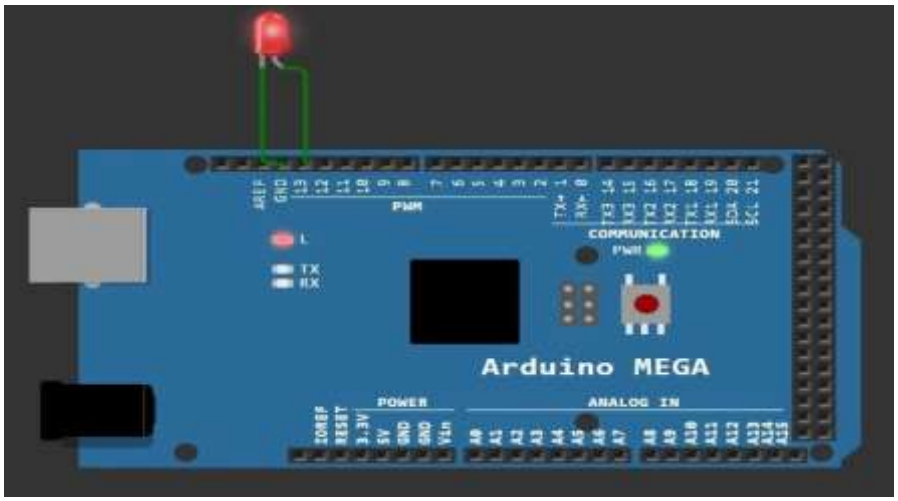
```
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);     // Declaring function of pin
}

void loop() {
  // put your main code here, to run repeatedly:
  for (int i=0; i<100; i++)     //Increasing the LED's intensity
  {
    analogWrite(13,i);
    delay(100);
  }
```

```
for (int j=100; j>0; j++)     //Decreasing the LED's intensity
{
  analogWrite(13,j);
  delay(100);
}
```

## Connection Diagram:



## Schematic Diagram:




**Result:** We constructed a fading LED project using analog output.

**Conclusion:** After the completion of this experiment, we were able to use delay(), analogWrite() functions and got a better understanding of Arduino and the various implementations of an LED.

# Experiment Number – 04

**Aim:** Write a Program to read an analog input and prints the voltage to the serial monitor.

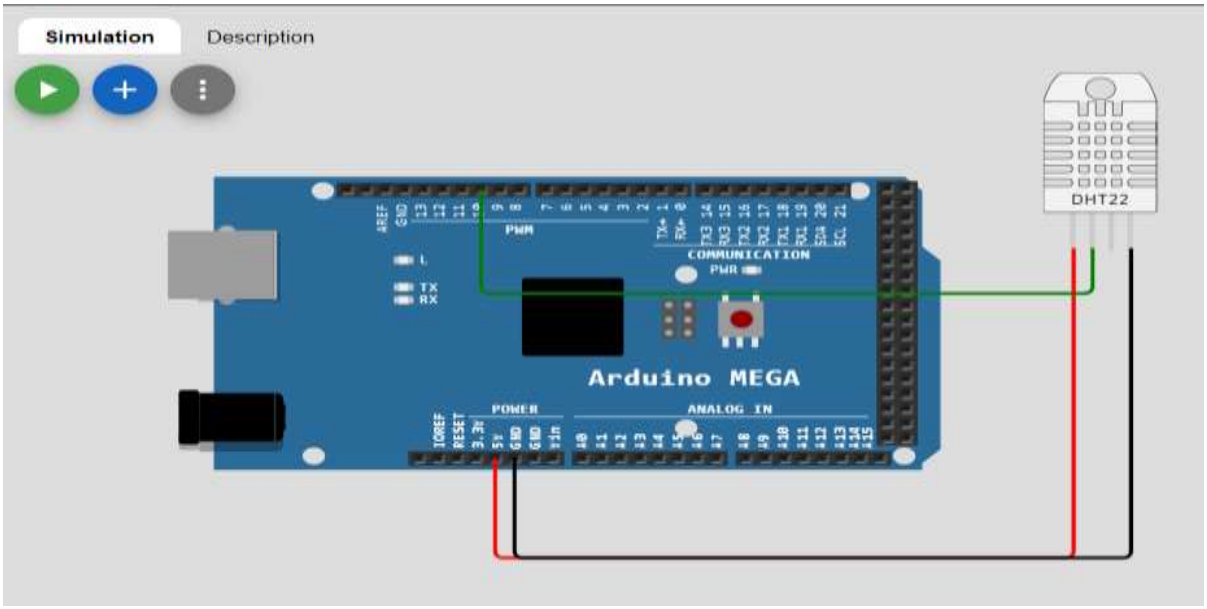
**Materials Required**: **Arduino Mega 2560 + USB A-to-B Cable, 3x Jumper Wires,** 1x Temperature Sensor **(**DHT11)

**Introduction:** A temperature sensor is exactly what it sounds like – a sensor used to measure ambient temperature. This particular sensor has three pins – a positive, a ground, and a signal. This is a linear temperature sensor. A change in temperature of one degree centigrade is equal to a change of 10 millivolts at the sensor output.

## Code:

```
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;
#define DHT11_PIN 10

void setup(){
  Serial.begin(9600);
}

void loop(){
  DHT.read(DHT11_PIN);
  Serial.print("temp:");
  Serial.print(DHT.temperature);
  Serial.print("  humi:");
  Serial.println(DHT.humidity);
  delay(1000);
}
```

## Connection Diagram:



**Result:** We constructed a Program to read an analog input and prints the voltage to the serial monitor


**Conclusion**: After the completion of this experiment, The experiment successfully demonstrated the capability to read analog inputs and accurately convert them into voltage readings, which were then displayed on the serial monitor. This foundational process enables real-time monitoring and analysis of analog signals, facilitating various applications in electronics and sensor-based systems.

# Experiment Number – 05

**Aim:** Write a Program to Control an LED using Button.

**Materials Required**: Arduino board, LED, Push button, Ohm resistor, A bunch of male to male wires

## Introduction:

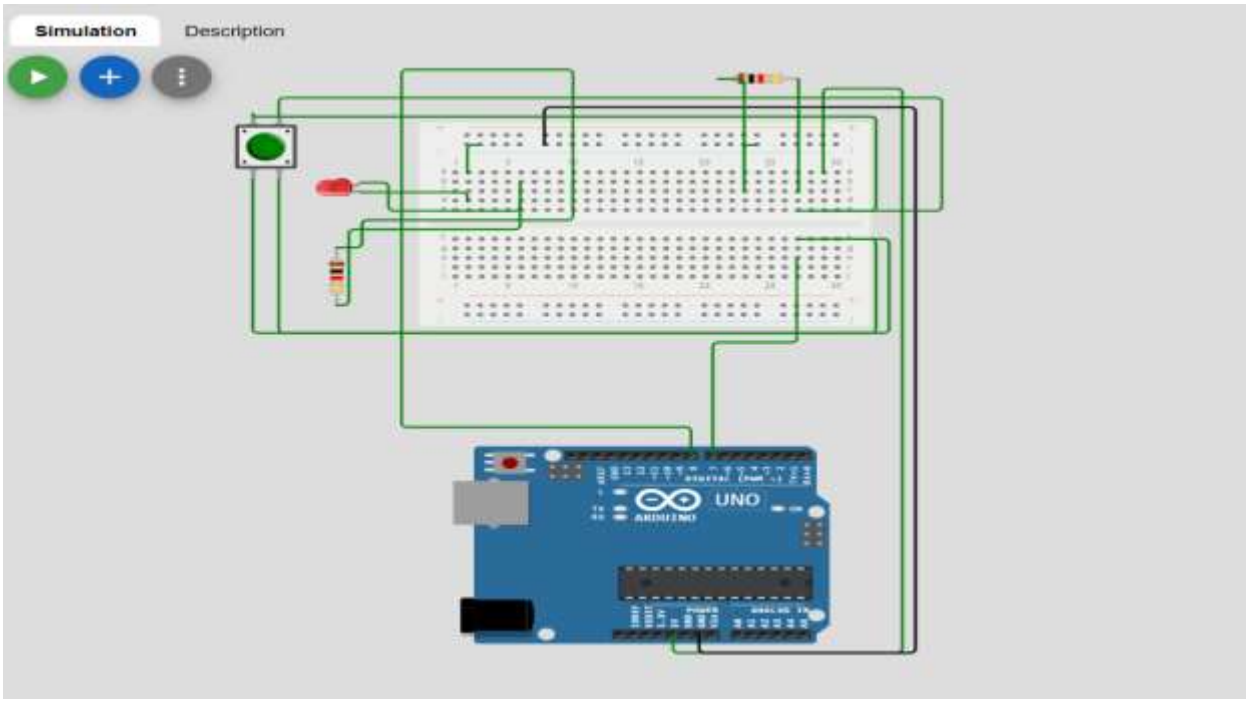
Step by step instructions to build the circuit :

- Plug a black wire between the blue line of the breadboard and a ground (GND) pin on the Arduino board.
- Plug the LED. You can notice that the LED has a leg shorter than the other. Plug this shorter leg to the ground (blue line here) of the circuit.
- Connect the longer leg of the LED to a digital pin (here pin no 8, you can change it). Add a 220 Ohm resistor in between to limit the current going through the LED.
- Add the push button to the breadboard, like in the picture.
- Connect one leg of the button to the ground, and put a 10k Ohm resistor in between. This resistor will act as a "pull down" resistor, which means that the default button's state will be LOW.
- Add a red wire between another leg of the button and VCC (5V).
- Finally, connect a leg of the button (same side as the pull down resistor) to a digital pin (here 7).

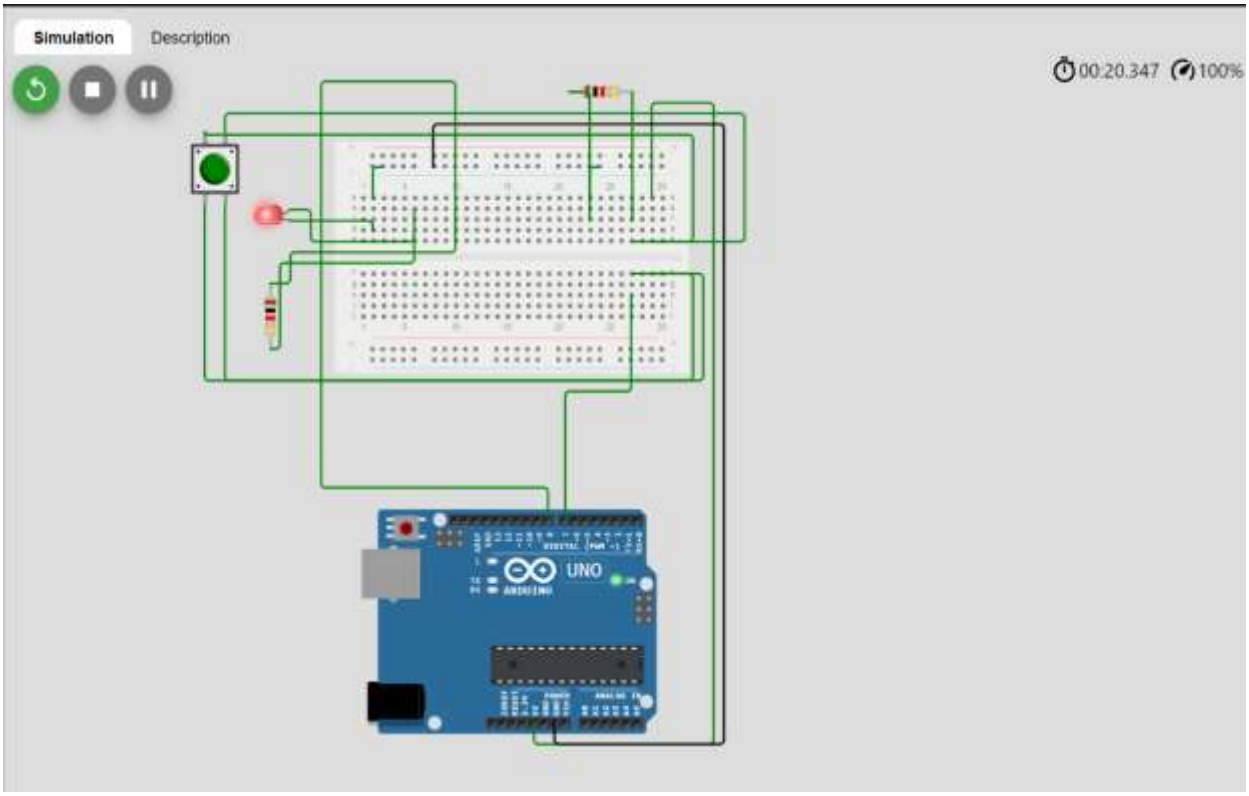
## Code:

```
#define LED_PIN 8
#define BUTTON_PIN 7
byte lastButtonState = LOW;
byte ledState = LOW;
unsigned long debounceDuration = 150; // millis
unsigned long lastTimeButtonStateChanged = 0;
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}
void loop() {
  if (millis() - lastTimeButtonStateChanged > debounceDuration) {
    byte buttonState = digitalRead(BUTTON_PIN);
    if (buttonState != lastButtonState) {
      lastTimeButtonStateChanged = millis();
      lastButtonState = buttonState;
      if (buttonState == LOW) {
        ledState = (ledState == HIGH) ? LOW: HIGH;
        digitalWrite(LED_PIN, ledState);
      }
    }
  }
}
```

## Connection Diagram:




## Schematic Diagram:



**Result:** We constructed a Program in which led is control by using button .

## Conclusion:

After completing the experiment to control an LED using a button, several key observations and conclusions can be drawn:

1. **Functionality Confirmation**: The experiment successfully demonstrated the ability to control an LED using a button. When the button was pressed, the LED turned on, and when the button was released, the LED turned off. This confirms that the hardware setup and coding logic were implemented correctly.
2. **Circuit Reliability:** The reliability of the circuit connections and components was crucial for the experiment's success. A stable and well-connected circuit ensured that the button press was accurately detected, and the LED responded promptly.
3. **Programming Logic:** The programming logic implemented in the microcontroller or microprocessor played a significant role. The code should efficiently detect the button state changes and toggle the LED accordingly.

# Experiment Number – 06

**Aim:** Write a program to Using a Piezo Buzzer

**Materials Required**: Arduino board, Piezo Buzzer**,** 3x Jumper Wires.

## Introduction:

In this circuit, we'll again bridge the gap between the digital world and the analog world. We'll be using a piezo buzzer that makes a small "click" when you apply voltage to it (try it!). By itself that isn't terribly exciting, but if you turn the voltage on and off hundreds of times a second, the piezo buzzer will produce a tone. And if you string a bunch of tones together, you've got music! This circuit and sketch will play a classic tune.

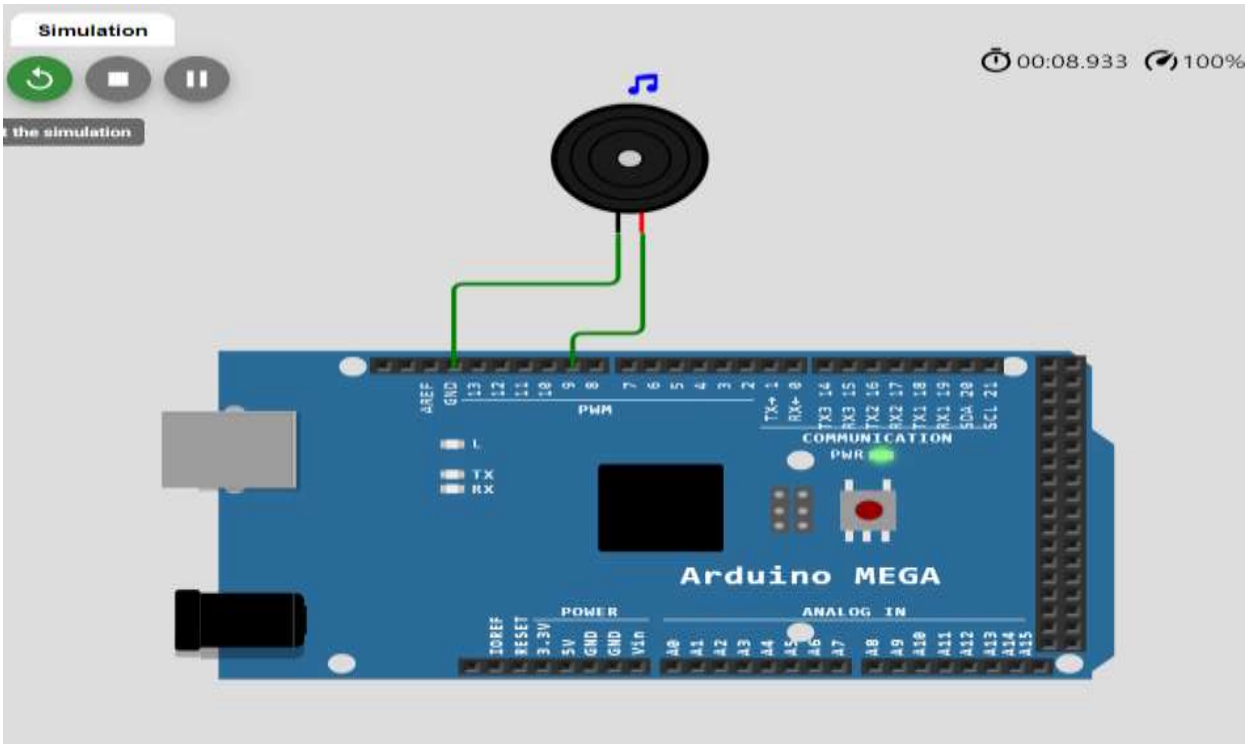
Note: Dowload pitches.h library.

## Code:

```
 #include "pitches.h"

int melody[] = {

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

};

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

int noteDurations[] = {

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

};


void setup() {

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

    int noteDuration = 1000 / noteDurations[thisNote];


    tone(8, melody[thisNote], noteDuration);


    int pauseBetweenNotes = noteDuration * 1.30;


    delay(pauseBetweenNotes);
```

```
    // stop the tone playing:


    noTone(8);
  }
}
void loop() {
  // no need to repeat the melody.
}
```

## Connection Diagram:




**Result:** We constructed a Program in which Piezo Buzzer make melody sound

**Conclusion:** In conclusion, the utilization of a piezo buzzer in this project has proven to be highly effective in generating audible alerts or tones. Through programming, we were able to control the frequency, duration, and pattern of the sound produced by the buzzer, allowing for versatile applications in various projects such as alarms, notifications, or interactive feedback systems.

# Experiment Number – 07

**Aim:** Write a program to Using a Laser Sensor

**Materials Required:** Arduino, Laser Sensor, 3XMale to Female Jumper Wires.

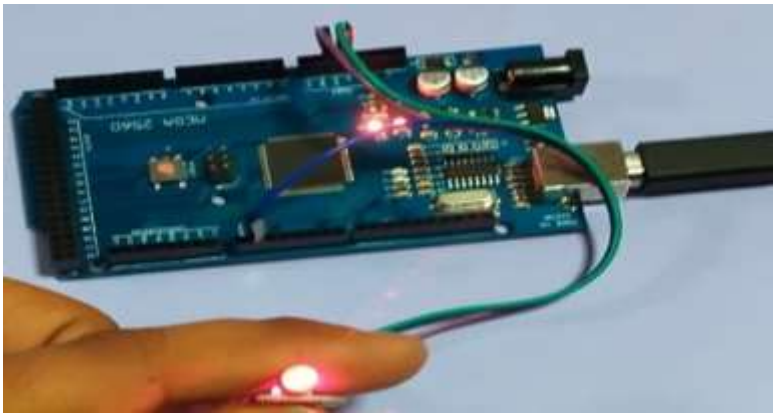
## Introduction:

Laser sensors are used where small objects or precise positions are to be detected. They are designed as through-beam sensors, retro-reflective sensors or diffuse reflection sensors. Laser light consists of light waves of the same wave length with a fixed phase ratio (coherence).

## Code:

```
int Laser = 2;
int voltage = 0;

void setup()

{
Serial.begin(9600);
pinMode (Laser,OUTPUT);
digitalWrite(Laser,LOW);
}

void loop()
{
  digitalWrite(Laser,HIGH);
  voltage = analogRead(A0);
  float voltage1 = voltage * (5.0 / 1023.0);
  Serial.print("the laser is ON and the voltage on the center pin is ");
  Serial.println(voltage1);
  Serial.println();
    delay(2000);
  digitalWrite(Laser,LOW);
  voltage = analogRead(A0);
  float voltage2 = voltage * (5.0 / 1023.0);
  Serial.print("the laser is OFF and the voltage on the center pin is ");
  Serial.println(voltage2);
  Serial.println();
    delay(2000);
}
```

## Connection Diagram:



**Result:** We constructed a Program in which Laser Light can No & Off.

**Conclusion:** After completing the experiment involving turning the laser light on and off using Arduino, several conclusions can be drawn:

1. **Functionality Validation:** The successful on/off control of the laser light using Arduino confirms the functionality of the hardware setup and the programming logic.

2. **Arduino's Capabilities:** The experiment demonstrates Arduino's ability to interface with external devices, such as the laser light, and control them based on programmed instructions.

3. **Programming Proficiency:** Participants or researchers involved in the experiment have likely enhanced their programming proficiency, particularly in Arduino programming, by writing code to control the laser light.

# Experiment Number – 08

**Aim:** Write a program to Driving a Servo Motor

**Materials Required**: Arduino, Servo Motor, 3X Jumper Wires.

## Introduction:

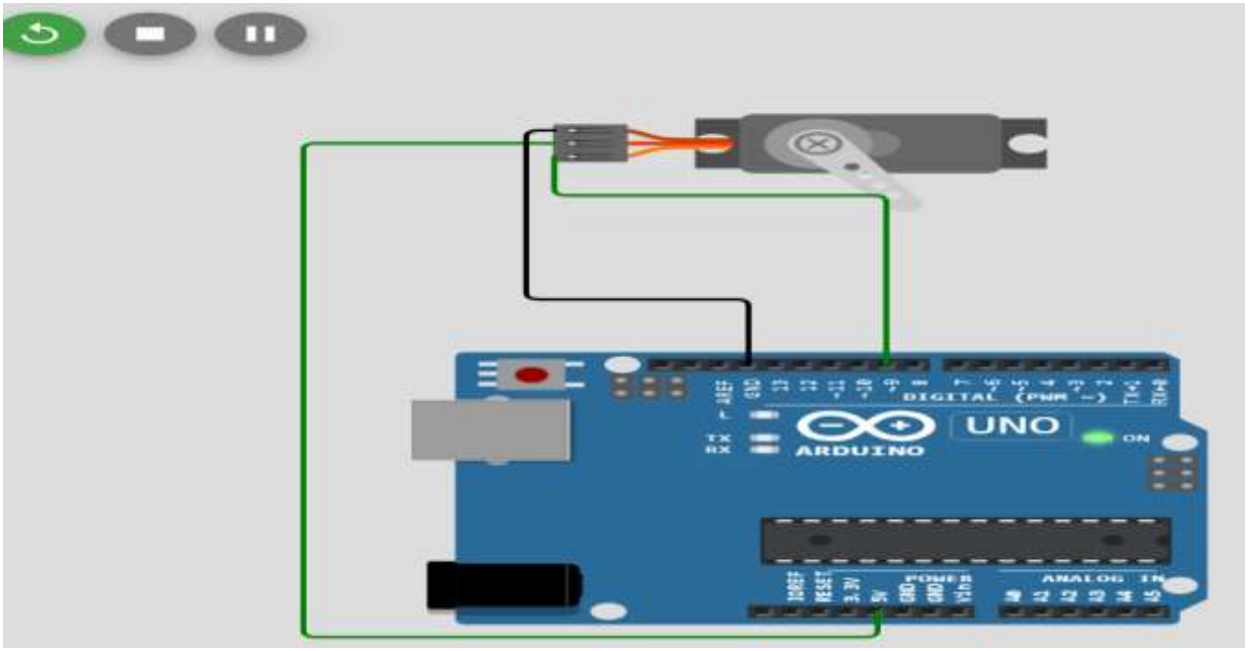
Servos are ideal for embedded electronics applications because they do one thing very well that motors cannot – they can move to a position accurately. By varying the pulse width of the output voltage to a servo, you can move a servo to a specific position. For example, a pulse of 1.5 milliseconds will move the servo 90 degrees. In this circuit, you'll learn how to use PWM (pulse width modulation) to control and rotate a servo.

## Code:

```
#include <Servo.h>  // servo library


Servo servo1;  // servo control object


void setup()
{

  servo1.attach(9, 900, 2100);
}


void loop()
{
  int position;
  servo1.write(90);

  delay(1000);

  servo1.write(180);

  delay(1000);

  servo1.write(0);

  delay(1000);

  for(position = 0; position < 180; position += 2)
  {
    servo1.write(position);
    delay(20);
  }
```

```
  for(position = 180; position >= 0; position -= 1)
  {
    servo1.write(position);
    delay(20);
  }
}
```

## Connection Diagram:




**Result:** We constructed a Program in which we Drive a Servo Motor.

**Conclusion:** After completing the experiment involving the servo motor, several conclusions can be drawn based on the observations and data collected:

1. Servo Motor Functionality: The experiment reaffirmed the functionality and versatility of servo motors in various applications. Servo motors are highly precise and can be controlled with great accuracy, making them suitable for tasks requiring specific angles or positions.

2. Control Interface: The control interface used to manipulate the servo motor proved effective in commanding its movement. Whether through PWM (Pulse Width Modulation) signals or other control mechanisms, the servo responded reliably to the input provided.

3. Response to Commands: The servo motor demonstrated prompt responsiveness to commands, indicating low latency in its operation. This characteristic is crucial in applications where real-time adjustments or precise control is necessary.

# Experiment Number -09

**Aim:** Write a program to demonstrate the functioning of PIR Motion Sensor.

**Materials Required**: Arduino, PIR Motion Sensor,LED.

## Introduction:

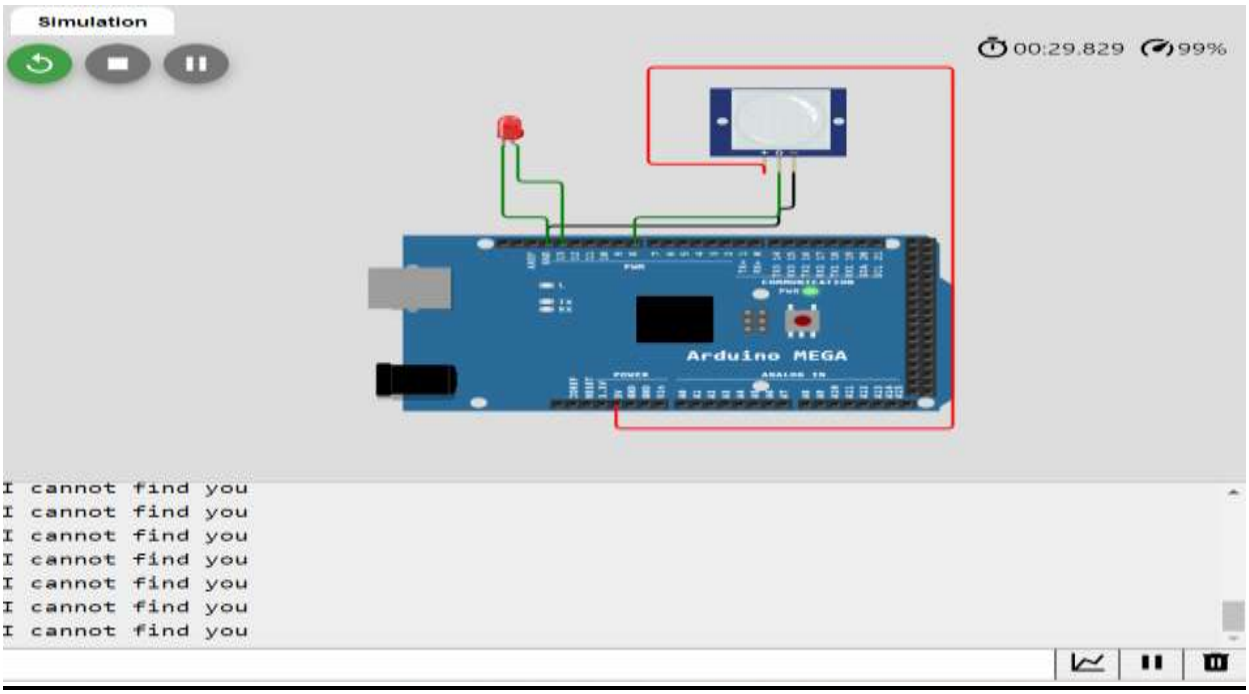
We will use a *PIR motion sensor* in this project. All objects (having temperature higher than absolute zero) emit radiations from the generated heat. These radiations cannot be detected by a human eye. Hence, electronic devices such as motion sensors, etc. are used for the detection of these radiations.

The Passive Infra-Red sensors or PIR sensors detect motion or movement of an object that detect infrared radiations, such as the human body. Hence, the use of sensors is very common.

## Code:

```
int LEDpin = 13;   // LED pin
int PIRpin = 8;    // The pin of Arduino connected to the PIR output
int PIRvalue = 0;  // It specifies the status of PIR sensor
void setup() {
pinMode(LEDpin, OUTPUT);
pinMode(PIRpin, INPUT);
// the output from the sensor is considered as input for Arduino
Serial.begin(9600);
}
void loop()
{
PIRvalue = digitalRead(PIRpin);
if (PIRvalue == HIGH)
{
digitalWrite(LEDpin, HIGH);
// turn ON LED if the motion is detected
Serial.println("hello, I found you...heyyy..");
}
else
 {
digitalWrite(LEDpin, LOW);
// LED will turn OFF if we have no motion
Serial.println("I cannot find you");
delay(1000);
}
}
```

## Connection Diagram:




**Result:** We constructed a Program in which we use PIR Motion Sensor.

**Conclusion:** This experiment successfully demonstrates the functionality of a PIR motion sensor using a program. The program reads the sensor output and controls an LED to indicate motion detection. By observing the LED and serial monitor messages, you can verify that the sensor detects motion and triggers the programmed response.

This is a basic example, and you can expand on this concept to explore more advanced functionalities like:

- Adjusting PIR sensor sensitivity for different detection ranges.
- Implementing timers for controlling light duration after motion detection.
- Connecting the PIR sensor to control other devices like alarms or cameras.

# Experiment Number -10

**Aim:** Write a program to demonstrate the functioning of Tilt Sensor.

**Materials Required**: Arduino, Tilt Sensor, Bread Board .

## Introduction:

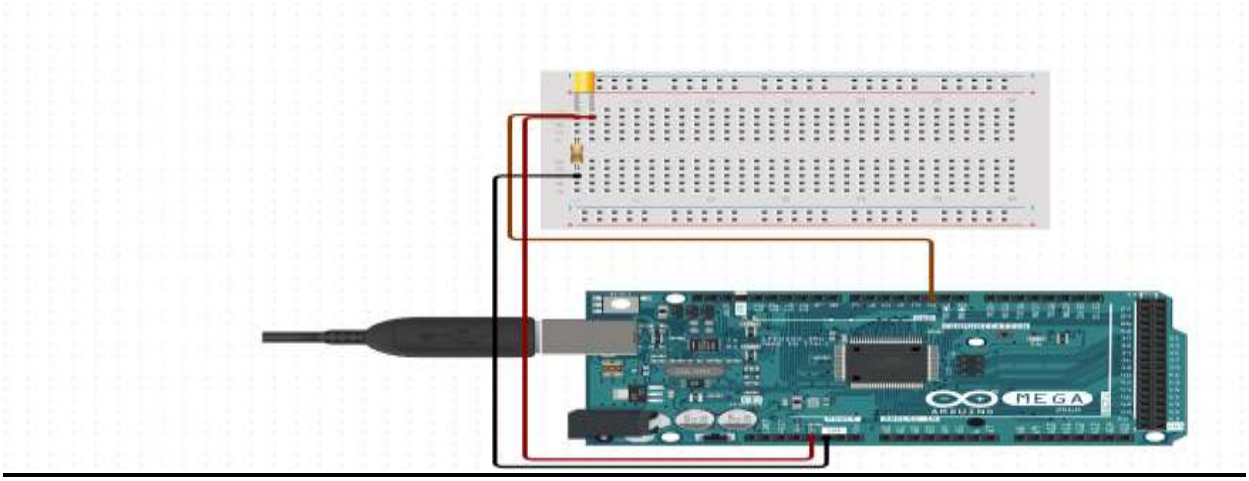
Tilt sensors are essential components in security alarm systems today. Standalone tilt sensors sense tilt angle or movement. Tilt sensors can be implemented using mercury and roller ball technology, and can be mounted using mechanical threading, magnets, or adhesives, depending on what type of surface they are being mounted to.

## Code:

```
#include "Arduino.h"
#include "Button.h"
#define TILTSWITCH_PIN_2
Button TiltSwitch(TILTSWITCH_PIN_2);
const int timeout = 10000;       //define timeout of 10 sec
char menuOption = 0;
long time0;
void setup()
{
   Serial.begin(9600);
   while (!Serial) ;
   Serial.println("start");
   TiltSwitch.init();
   menuOption = menu();
}
void loop()
{
   if(menuOption == '1') {
   bool TiltSwitchVal = TiltSwitch.read();
   Serial.print(F("Val: ")); Serial.println(TiltSwitchVal);
   }
   if (millis() - time0 > timeout)
   {
      menuOption = menu();
   }
}
char menu()
{
   Serial.println(F("\nWhich component would you like to test?"));
   Serial.println(F("(1) Tilt Sensor - AT407"));
   Serial.println(F("(menu) send anything else or press on board reset button\n"));
   while (!Serial.available());
   while (Serial.available())
   {
      char c = Serial.read();
      if (isAlphaNumeric(c))
      {
```

```
      if(c == '1')
        Serial.println(F("Now Testing Tilt Sensor - AT407"));
          else
          {
             Serial.println(F("illegal input!"));
             return 0;
          }
          time0 = millis();
          return c;
      }
   }
}
```

**Connection Diagram:**



**Result:** We constructed a Program in which we use Tilt Sensor.

**Conclusion:** In this experiment, we successfully designed and implemented a program to demonstrate the functioning of a TILT sensor. The TILT sensor, also known as a tilt switch, is a simple device that detects orientation or tilt.

Through our program, we were able to read the output of the TILT sensor and interpret it to determine the orientation of the sensor relative to the vertical axis. This information allowed us to detect when the sensor was tilted beyond a certain threshold angle.