

Step 1: Go to the Arduino.cc Website

Go to the website www.arduino.cc in order to download the software.

Hover over the 'Software' tab and click on 'Downloads'.

Step 2: Scroll down until you see the link that says 'Windows installer' and click on it.

Step 3: Begin the Download

After clicking on the download link you'll be redirected to the donation page, here you can donate or skip it if you like by clicking on the 'Just download' link.

Step 4: Begin the Installation Process

Open the downloaded file.

A new window will open asking you to agree to the license agreement.

Click on 'I agree' to continue.

Step 5: Select What to Install

Now you'll see all the available options to install the software with.

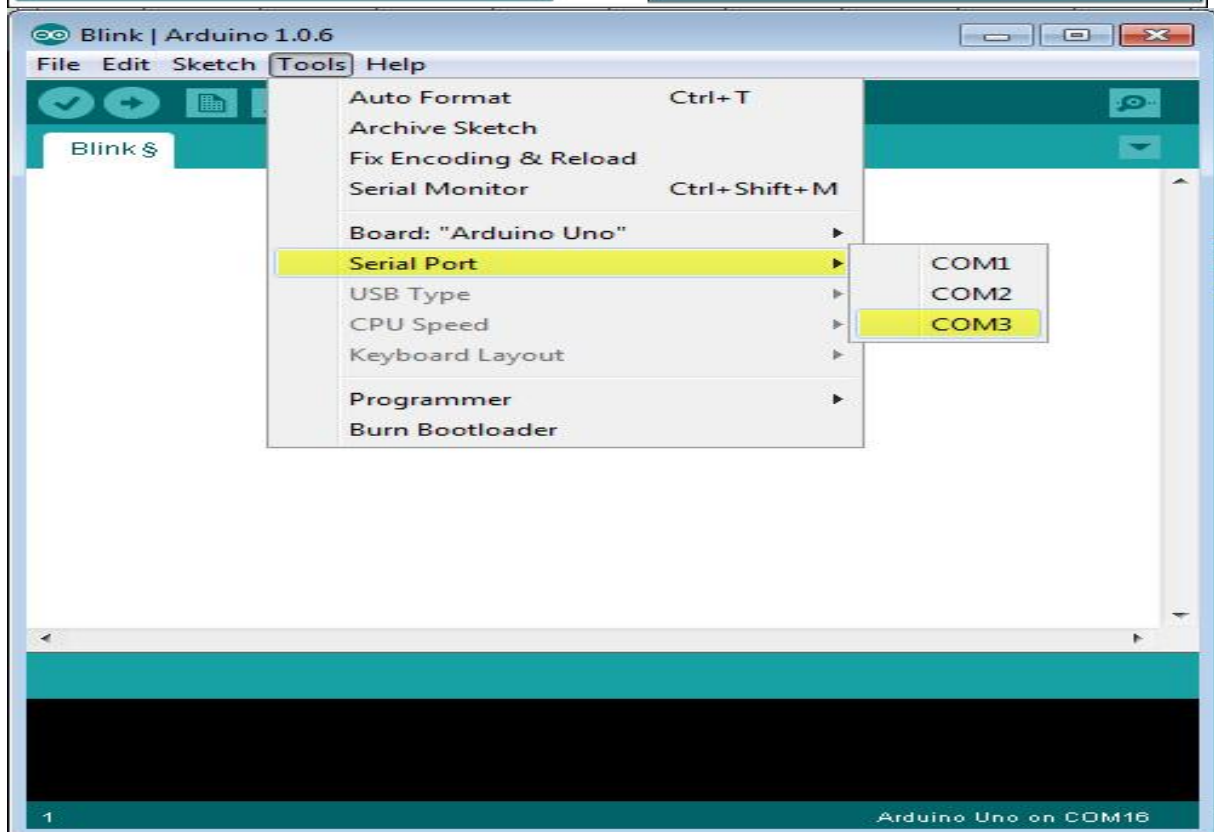
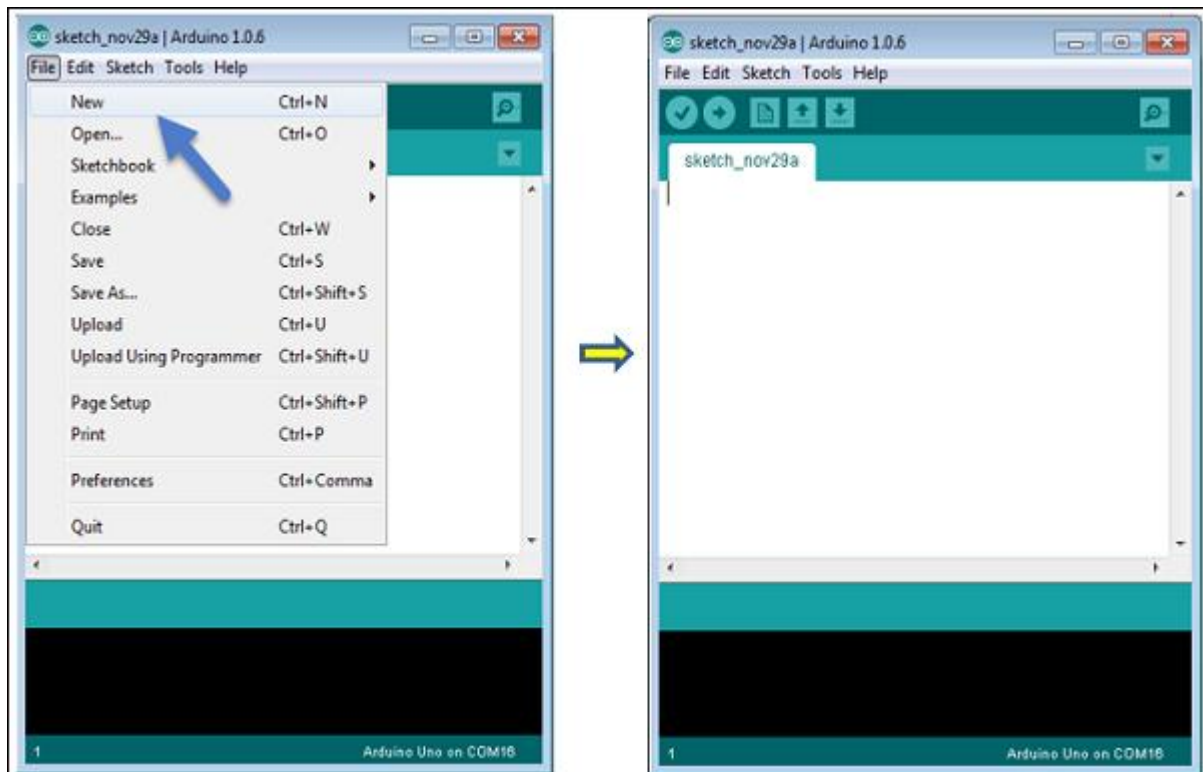
If you don't know what you need, it is best to keep everything checked as you can change it later when the installation has finished.

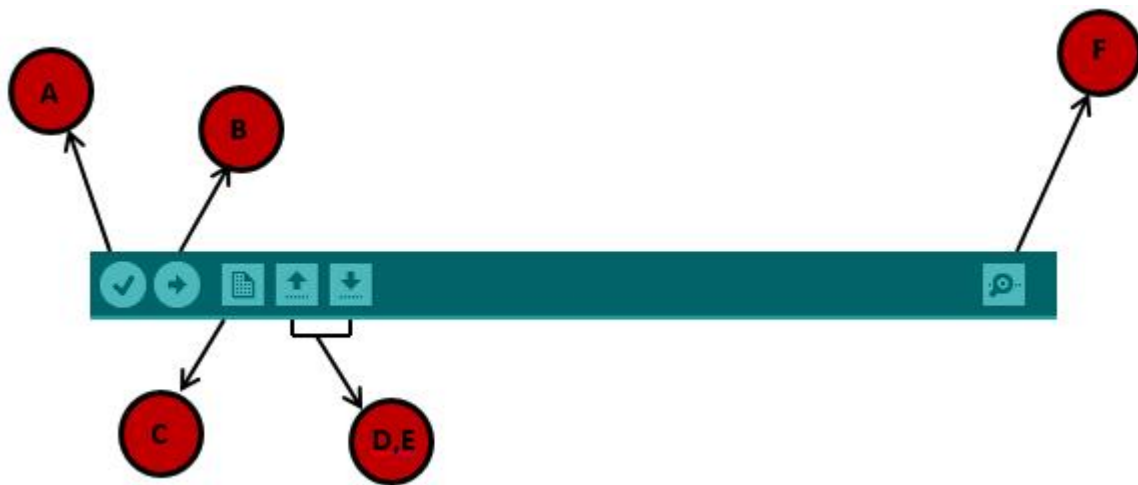
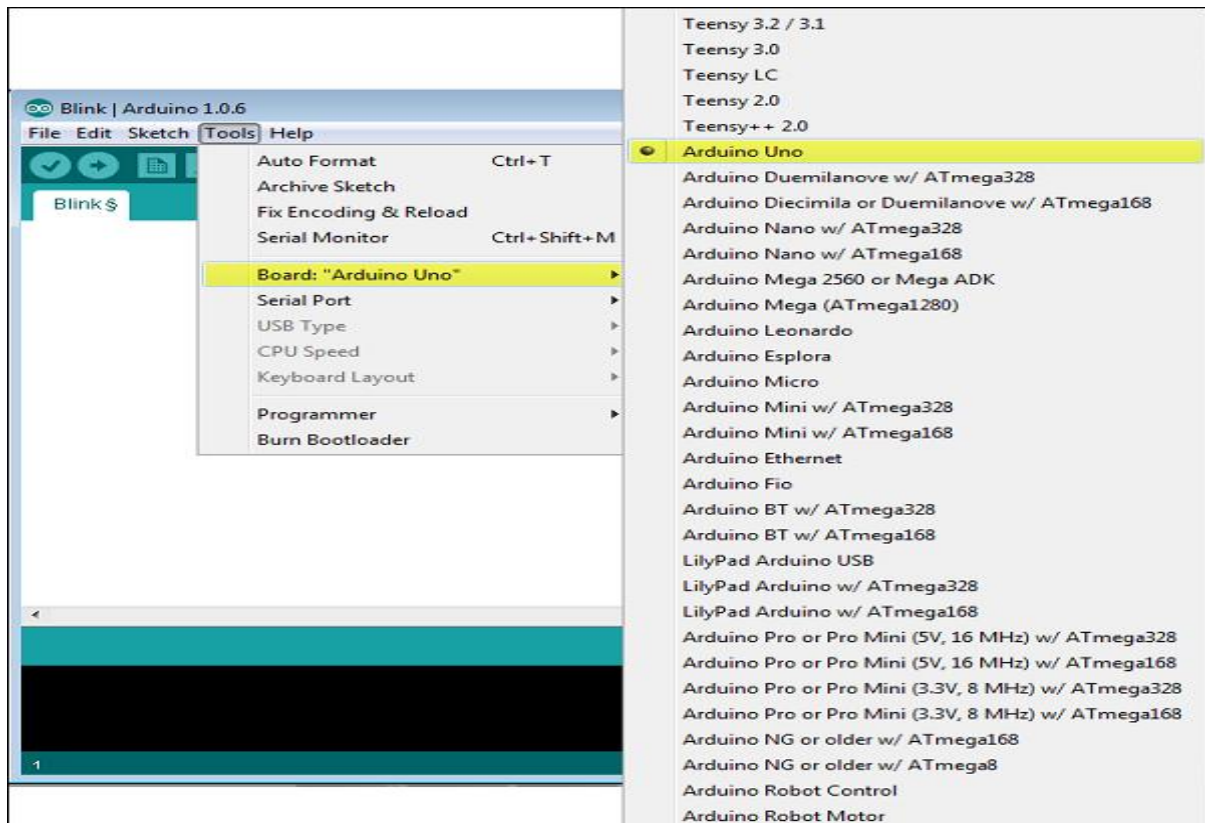
Click on 'Next' to continue.

Step 6: Choose the Installation Path

Step 7: Finish the Installation

Step 8: Launch the Arduino IDE





LED Blinking with Arduino

Aim :

To blink an LED connected to pin 13 on an Arduino board at one-second intervals.

INTRODUCTION

LED blinking refers to the process of continuously turning an LED (Light Emitting Diode) and off in a repetitive pattern. It is a simple and common demonstration in electronics and microcontroller-based projects.

Procedure:

Step 1 :Define Pin Mode: Define pin 13 as an output pin.

Step 2 :Setup Function: Set up pin 13 as an output pin in the setup() function.



Step 3 :Loop Function: Continuously turn the LED on and off in the loop() function with one-second delays.

Step 4 :Test LED connected to pin 13 blinks on and off at one-second intervals as intended.

Step 5 :Stop Program and Disconnect: Halt program execution and disconnect components.

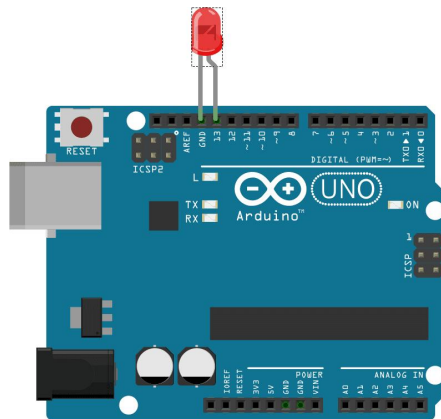
HARDWARE

CONNECTION:

DIN		D13
GND		GND

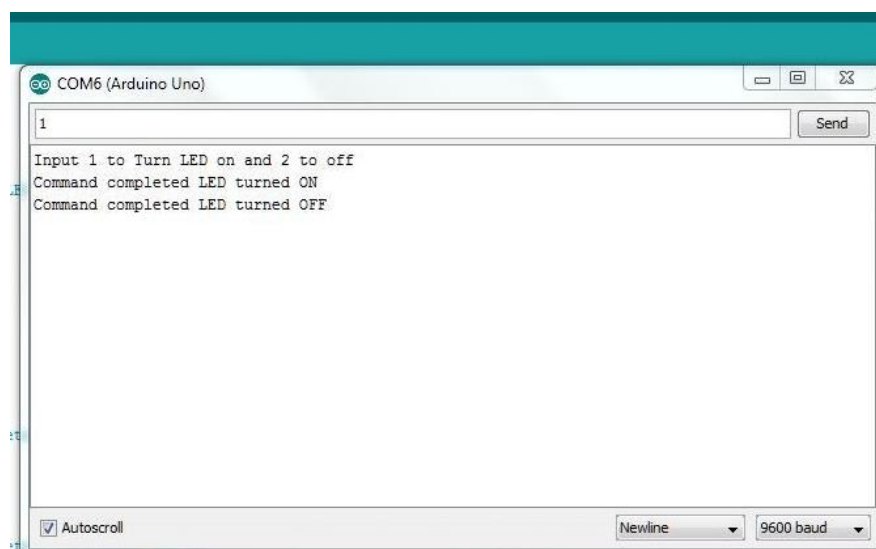
Result

The LED blinking experiment with Arduino has been successfully executed, demonstrating the basic operation of digital output pins.



```
int pinMode=13;  
void setup ()  
{  
  pinMode(13, OUTPUT);  
}  
void loop()   
{  
  digitalWrite(13, HIGH);  
  delay(1000);  
  digitalWrite(13, LOW);  
  delay(1000);  
}
```

Output:



DETECT THE PRESENCE OF AN OBJECT USING IR SENSOR

Aim:

The aim of the provided Arduino program is to interface an IR (Infrared) sensor with an Arduino Uno board and use it to detect the presence of an object.

Introduction:

- IR sensor is an electronic device, that emits the light in order to sense some object of the surroundings.
- An IR sensor can measure the heat of an object as well as detects the motion.
- Usually, in the infraraed, all the objects radiate some form of thermal radiation.

Procedure

Step 1: Open Arduino IDE.

Step 2: Create a New Sketch:

Start a new sketch by selecting File > New in the Arduino IDE.

Step 3: Define Pin Constants

Define constants for the pins connected to the IR sensor output and the LED:

Step 4: Setup Function

In the setup() function, configure the pins

Step 5: In the loop() function,

continuously read the state of the IR sensor and control the LED based on its output

Step 6: Verify your code for any syntax errors by clicking on the verify button (checkmark icon) in the Arduino IDE.

Step 7: Power up your Arduino Uno board and test the IR sensor with objects to observe the LED behavior.

Step 8: Stop the Program and Disconnect the Components.

Hardware connections:

1. Connect the wire between the Arduino Uno board and IR Sensor as given below


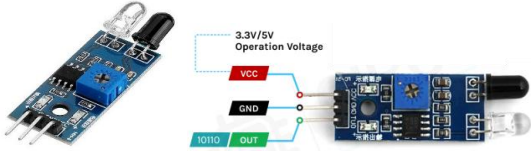


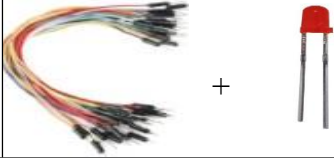
VCC	→	+5v
GND	→	Gnd
Dout	→	D2
LED	→	D13

2. Connect the USB Cable between Arduino Uno and PC.
3. Integrate the program into the arduino Uno board.

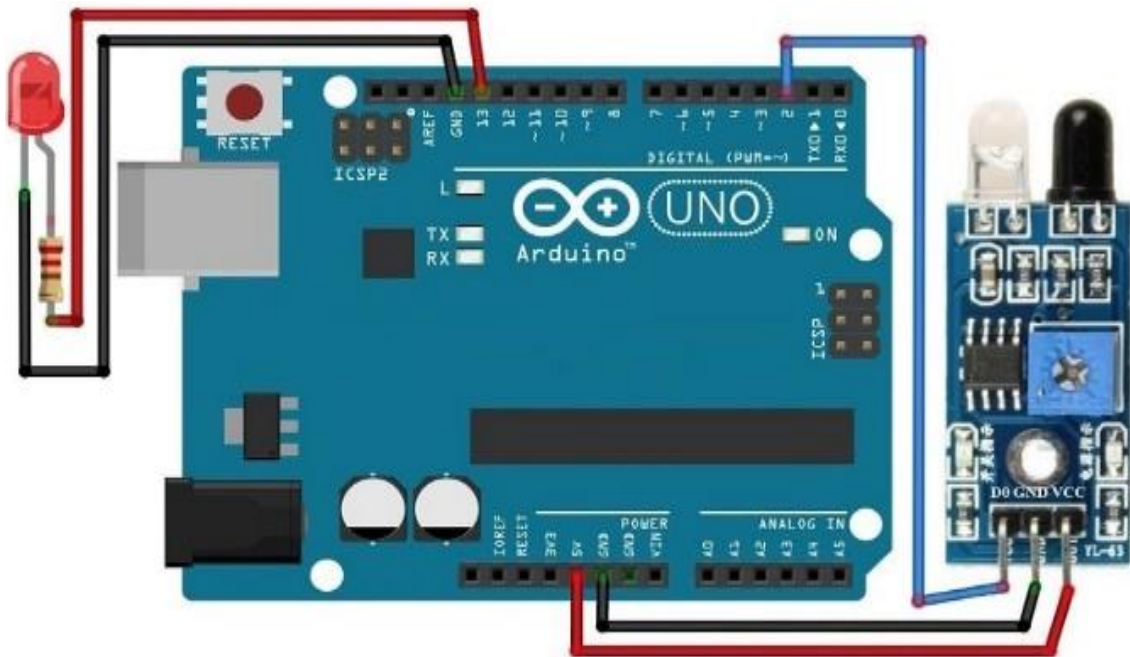
Result :

Successfully developed and uploaded Arduino code to the Arduino Uno board and connected the IR sensor, LED to the Arduino Uno the program resulted in a functional IR sensor-based object detection system integrated with an Arduino Uno, providing real-time feedback through an LED indicator.

Parts required:

<u>Sl.no</u>	<u>Description</u>	<u>Image</u>	<u>Quantity</u>
1	Arduino Uno board		1
2	IR Sensor		1
4	External power source		1
5	Connecting power cable		1
6	Connecting wires		1

Circuit diagram:



```
// IR Sensor CODE
```

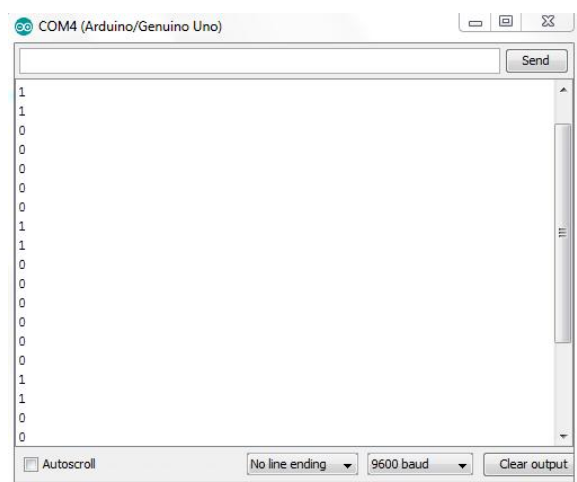
```
intSensorPin = 2;  
intOutputPin = 13;
```

```
void setup()  
{  
  pinMode(OutputPin, OUTPUT);  
  pinMode(SensorPin, INPUT);  
  Serial.begin(9600);  
}
```

```
void loop()  
{  
  intSensorValue = digitalRead(SensorPin);
```

```
  Serial.print("SensorPin Value: ");  
  Serial.println(SensorValue);  
  delay(1000);
```

```
  if (SensorValue==LOW)  
  {  
    digitalWrite(OutputPin, HIGH);  
  }  
  else  
  {  
    digitalWrite(OutputPin, LOW);  
  }  
}
```



Basic Touch Sensor System

Aim

To create a basic touch sensor system using an Arduino board, where the LED serves as an indicator of touch sensor activation, and serial communication provides additional status information.

Introduction:

- Touch sensors are also called as tactile sensors and are sensitive to touch, force or pressure.
- Touch sensors are finding their way into many applications, from mobile phones to remote controls and appliance control panels.
- Mechanical button and switch replacement continues to be implemented in a wide variety of application

Procedure

Step 1: Open Arduino IDE.

Step 2: Create a New Sketch:

Start a new sketch by selecting File > New in the Arduino IDE.

Step 3: Define Pin Constants

Define constants for the pins connected to the Touch sensor (tsPin=7) output and the LED (ledPin = 3)

Step 4: Setup Function

In the setup() function, configure the pins

Step 5: In the loop() function,

continuously read the state of the Touch sensor and control the LED based on its output Transmit status information ("TOUCHED" or "not touched") via serial communication.

Step 6: Verify your code for any syntax errors by clicking on the verify button (checkmark icon) in the Arduino IDE.

Step 7: Power up your Arduino Uno board and test the Touch sensor and Transmit status information ("TOUCHED" or "not touched")

Step 8: Stop the Program and Disconnect the Components.

Hardware connections:

Connect the wire between the Arduino Uno board and Touch Sensor as given below


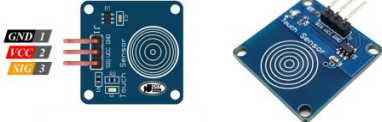


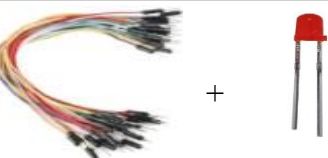
VCC	→	+5v
GND	→	Gnd
SIG	→	D7
LED	→	D3

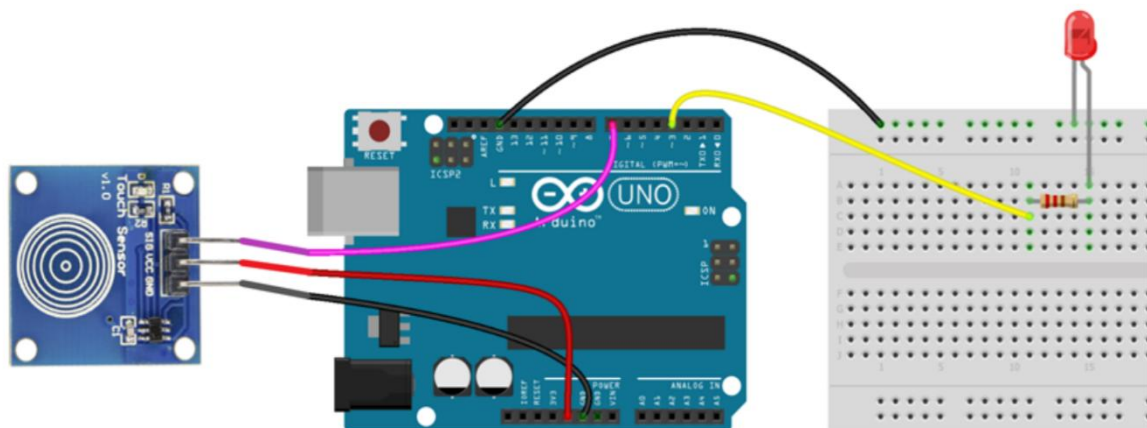
Connect the USB Cable between Arduino Uno and PC.

Result:

The touch sensor system has been successfully implemented with Arduino, providing real-time touch detection and status feedback through LED indicators and serial communication.

Parts required:

<u>Sl.no</u>	<u>Description</u>	<u>Image</u>	<u>Quantity</u>
1	Arduino Uno board		1
2	Touch Sensor		1
4	External power source		1
5	Connecting power cable		1
6	Connecting wires		1



```
#define tsPin7 // touch sensor pin
```

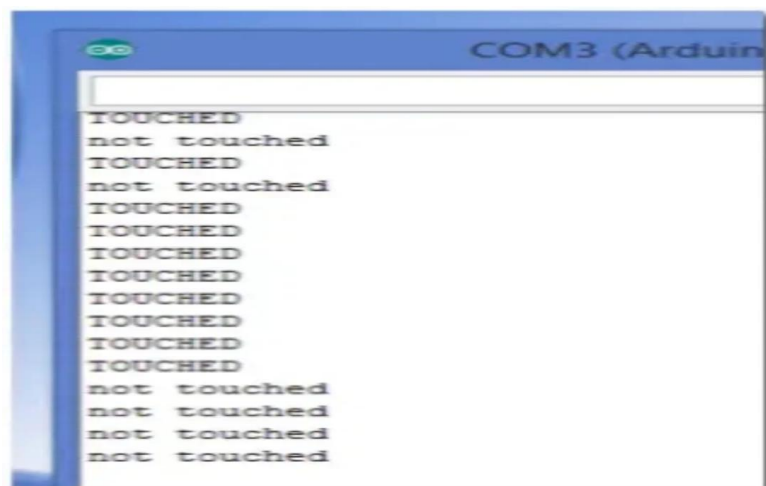
```

int ledPin = 3; // led pin

void setup()
{
  Serial.begin(9600);    //sensor baud rate
  pinMode(ledPin, OUTPUT); // led
  pinMode(tsPin, INPUT); // touch sensor
}
void loop()
{
  int tsValue = digitalRead(tsPin);
  delay(100);

  if (tsValue == HIGH)
  {
    digitalWrite(ledPin, HIGH); // led on
    Serial.println("TOUCHED");
  }
  else
  {
    digitalWrite(ledPin, LOW); // led off
    Serial.println("not touched");
  }
}

```



Object Distance Measurement with Ultrasonic Sensor and Arduino

Aim :

To create a distance measurement system using an ultrasonic sensor with an Arduino board, capable of measuring distances in both inches and centimeters.

INTRODUCTION:

- The ultrasonic sensor is a non-contact type of sensor used to measure an object's distance and velocity.
- This sensor operates on sound wave property to measure the velocity and distance of the object (
- Sounds with a frequency of 20 kHz and higher are referred to as ultrasound (or ultrasonic sound).

Procedure

Step 1 :Open Arduino IDE: Launch the Arduino IDE.

Step 2 :Create New Sketch: Start a new sketch.

Step 3 :Define Pin Constants: Define constants for trigger (pingPin) and echo (echoPin) pins.

Step 4 :Setup Function: Initialize serial communication in the setup() function.

Step 5 :Loop Function: Continuously measure distances in inches and centimeters in the loop() function.

Step 6: Conversion Functions: Define functions to convert microseconds to inches and centimeters.

Step 7 :Verify Code: Verify code for any syntax errors.





Step 8 :Upload Code: Upload the code to the Arduino board.

Step 9 :Monitor Serial Output: Open the serial monitor to observe distance measurements.

Step 10 : Test: Test the system by placing objects at different distances.

Step 11 :Stop Program and Disconnect: Halt program execution and disconnect components.





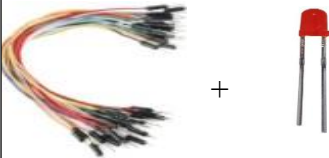
HARDWARE CONNECTION:

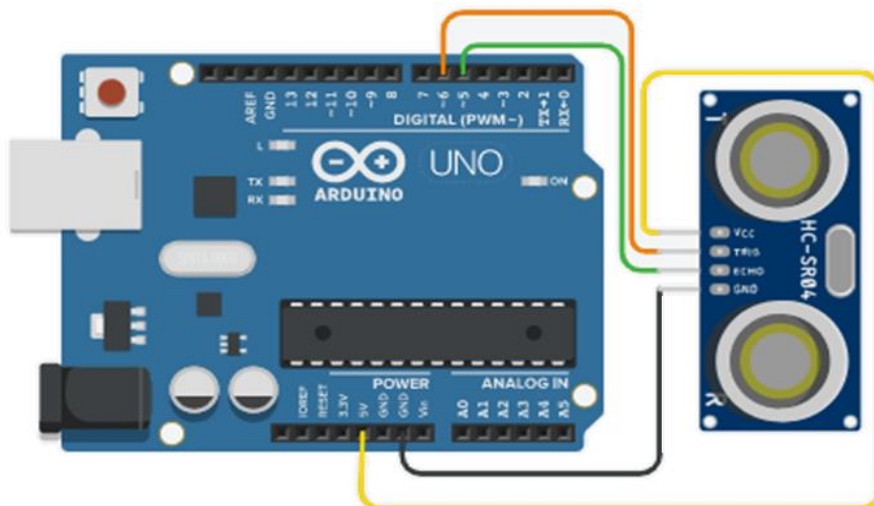
VCC		5V
GND		GND
TRIG		D6
ECHO		D5

Result

The ultrasonic distance sensor system has been successfully implemented with Arduino, providing accurate distance measurements and real-time feedback.

Parts required:

Sl.no	Description	Image	Quantity
1	Arduino Uno board		1
2	Ultrasonic Sensor		1
4	External power source		1
5	Connecting power cable		1
6	Connecting wires		1



```

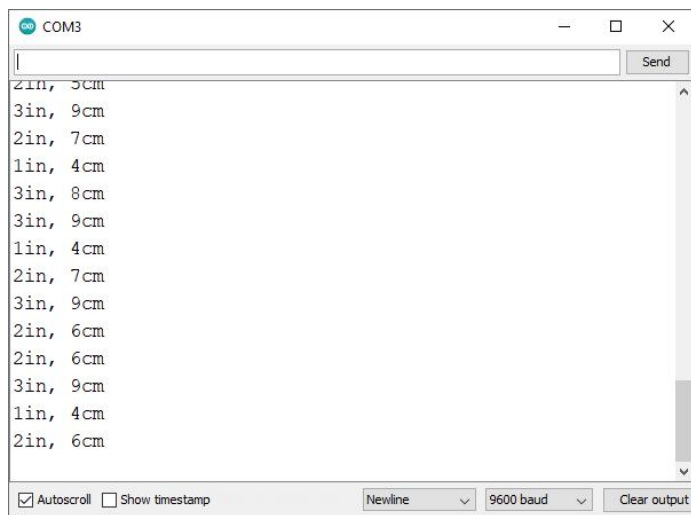
int pingPin = 6;
int echoPin = 5;
void setup()
{
  Serial.begin(9600); // Starting Serial Terminal
}

void loop()
{
  long duration, inches, cm;
  pinMode(6, OUTPUT);
  digitalWrite(6, 0);
  delayMicroseconds(2);
  digitalWrite(6, 1);
  delayMicroseconds(10);
  digitalWrite(6, 0);
  pinMode(5, INPUT);
  duration = pulseIn(5, HIGH);
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);
  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(100);
}

long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}

```



Vibration Sensor with LED Indicator

Aim :

To create a system using an Arduino board that detects vibrations using a vibration sensor and indicates them by blinking an LED.

Introduction

- Vibration sensors, also known as accelerometers or shake sensors, are electronic devices designed to detect and measure vibrations, oscillations, or movements in various objects or structures.
- They play a crucial role in monitoring and analyzing mechanical systems, industrial machinery, vehicles, buildings, and even human activities.
- These sensors work on the principle of detecting changes in acceleration, which are typically converted into electrical signals proportional to the magnitude of the vibration

Procedure

Step 1 :Open Arduino IDE: Launch the Arduino IDE.

Step 2 :Define Pin Variables: Define variables for the vibration sensor pin (vib_pin) and LED pin (led_pin).

Step 3: Setup Function: Configure the vibration sensor pin as INPUT and the LED pin as OUTPUT in the setup() function.

Step 4: Loop Function: Continuously read the state of the vibration sensor in the loop() function.





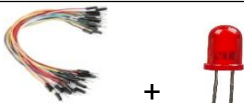
Step 5: If vibration is detected (sensor output is HIGH), turn on the LED for one second and then turn it off for one second.

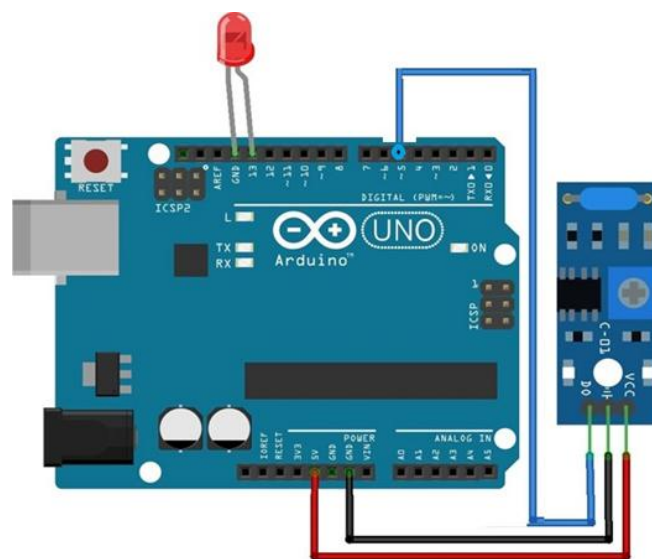
If no vibration is detected (sensor output is LOW), keep the LED off.

Step 6: Test the system

Step 7: Halt program execution and disconnect components.

Parts required:

<u>Sl.no</u>	<u>Description</u>	<u>Image</u>	<u>Quantity</u>
1	Arduino board		1
2	Vibration Sensor		1
4	External power source		1
5	Connecting power cable		1
6	Connecting wires and led		1



CODE:

```
intvib_pin=5;
intled_pin=13;

voidsetup()
{
  pinMode(vib_pin,INPUT);
  pinMode(led_pin,OUTPUT);
}

voidloop()
{
  intval;
  val=digitalRead(vib_pin);
  if(val==1)
  {
    digitalWrite(led_pin,HIGH);
    delay(1000);
    digitalWrite(led_pin,LOW);
    delay(1000);
  }
  else
    digitalWrite(led_pin,LOW);
}
```

PIR Sensor

PIR Motion Sensor Interface with Arduino

Aim:

To interface a PIR (Passive Infrared) motion sensor with an Arduino board and detect motion within its field of view.

Introduction

The PIR motion sensor is ideal to detect movement. PIR stand for “Passive Infrared”. Basically, the PIR motion sensor measures infrared light from objects in its field of view.

So, it can detect motion based on changes in infrared light in the environment. It is ideal to detect if a human has moved in or out of the sensor range.

Procedure

Step 1: Define the pin connected to the output of the PIR sensor
(e.g., `int pirPin = 2;`).

Step 2: In the `setup()` function, initialize serial communication and set the PIRpin as an input.

Step 3: In the `loop()` function, continuously read the state of the PIR sensor:

Step 4: Use `digitalRead()` to read the state of the PIR pin.

Step 5: If motion is detected (HIGH state), print "Motion detected!" to the serial monitor.

Step 6: If no motion is detected (LOW state), print "No motion detected." to the serial monitor.

Step 7: Add a small delay to prevent continuous readings (e.g., `delay(500);`).

Step 8: Upload Code:

Upload the Arduino code to the Arduino UNO board using the Arduino IDE.





Step 9: Testing: Open the Arduino Serial Monitor to observe the real-time motion detection status.

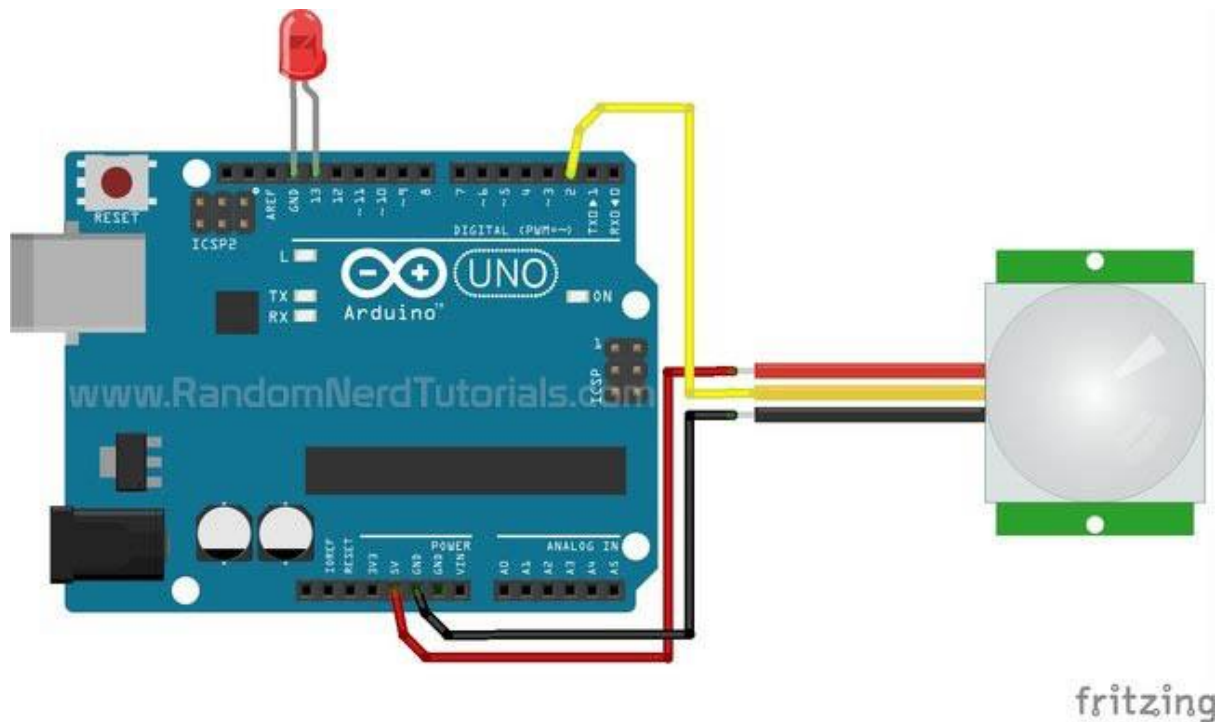
Step 10: Move within the field of view of the PIR sensor to test motion detection.

Result:

The PIR motion sensor interface with Arduino has been successfully implemented, demonstrating its capability to detect motion reliably.

Parts required:

<u>Sl.no</u>	<u>Description</u>	<u>Image</u>	<u>Quantity</u>
1	Arduino UNO Board		1
2	PIR Motion Sensor (HCSR501)		1
3	Connecting Power cable		1
4	JUMPER CABLE		1



Hardware connections:

1. Connect the wire between the Arduino UNO board and PIR Sensor detection sensor

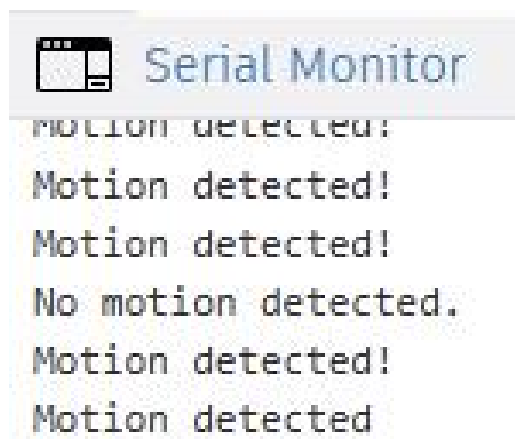
2. GND – connect to ground
3. OUT – connect to an Arduino digital pin
4. 5V – connect to 5V

Module Pin	Arduino Pin
GND	GND
OUT	D1
5V	5V

CODE:

```
// Define the pin connected to the output of the PIR sensor
int pirPin = 2;
void setup()
{
  Serial.begin(9600); // Initialize serial communication
  pinMode(pirPin, INPUT); // Set the PIR pin as an input
}

void loop()
{
  // Read the state of the PIR sensor
  int pirState = digitalRead(pirPin);
  if (pirState == HIGH)
  {
    Serial.println("Motion detected!");
  }
  else
  {
    Serial.println("No motion detected.");
  }
  // Add a small delay to prevent continuous readings
  delay(500);
}
```



Soil Moisture Measurement with Arduino

Aim:

To interface a soil moisture sensor with an Arduino UNO board and measure soil moisture levels.

Introduction:

Soil moisture sensors play a crucial role in agriculture, gardening, and environmental monitoring by measuring the water content in soil. This project aims to interface a soil moisture sensor with an Arduino UNO board to accurately measure soil moisture levels.

Procedure :

Step 1 : Define the pin connected to the soil moisture sensor

(e.g., `const int soilMoisturePin = A0;`).

Step 2 : In the `setup()` function, initialize serial communication for debugging.

Step 3 : In the `loop()` function, continuously read the analog value from the soil moisture sensor using `analogRead()`.

Step 4 : Map the analog value to a percentage value representing the soil moisture level using `map()`.

Step 5 : Print the soil moisture percentage to the serial monitor using `Serial.print()` and `Serial.println()`.

Step 6 : Add a delay to prevent spamming the serial monitor using `delay()`.

Step 7 : Upload Code: Upload the Arduino code to the Arduino UNO board using the Arduino IDE.

Step 8 : Testing: Open the Arduino Serial Monitor to observe real-time soil moisture percentage readings.

Step 9:

Hardware connections:

Connect the soil moisture sensor to the Arduino UNO board:

GND of soil moisture sensor to GND of Arduino


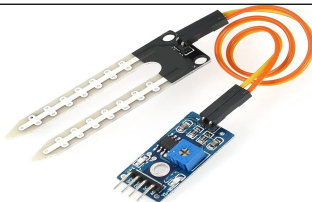



OUT of soil moisture sensor to analog pin A0 of Arduino

5V of soil moisture sensor to 5V of Arduino

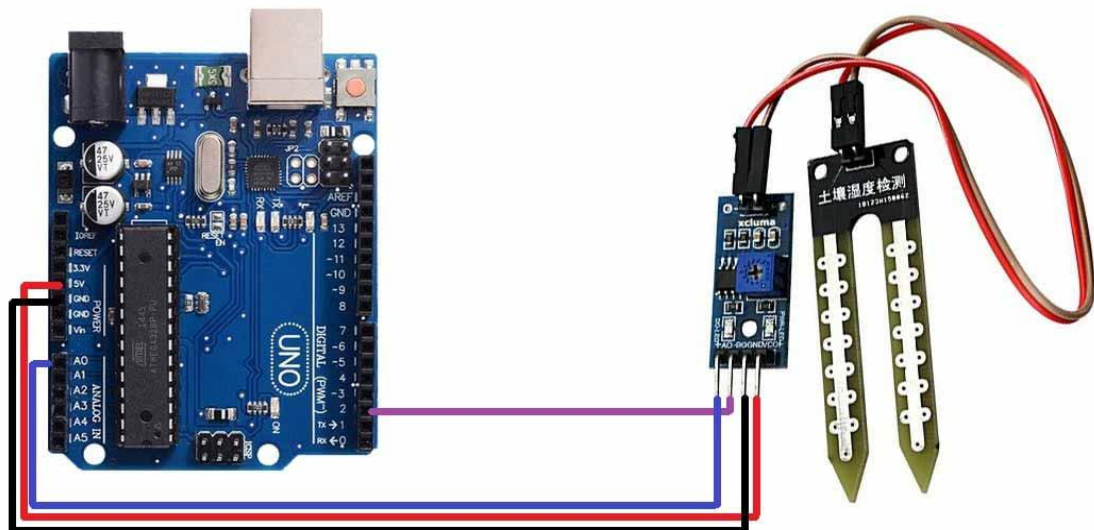
5V	VCC
GND	GND
A0	A0

Result : The soil moisture measurement system with Arduino has been successfully implemented, providing a reliable method for monitoring soil moisture levels.

Parts required:

<u>Sl.no</u>	<u>Description</u>	<u>Image</u>	<u>Quantity</u>
1	Arduino board		1
2	Soil Moisture Senor		1
4	External power source		1
5	Connecting power cable		1
6	Connecting wires		1

Circuit diagram:




```

// Define the pin connected to the soil moisture sensor
const int soilMoisturePin = A0;

void setup()
{
  // Initialize serial communication for debugging
  Serial.begin(9600);
}

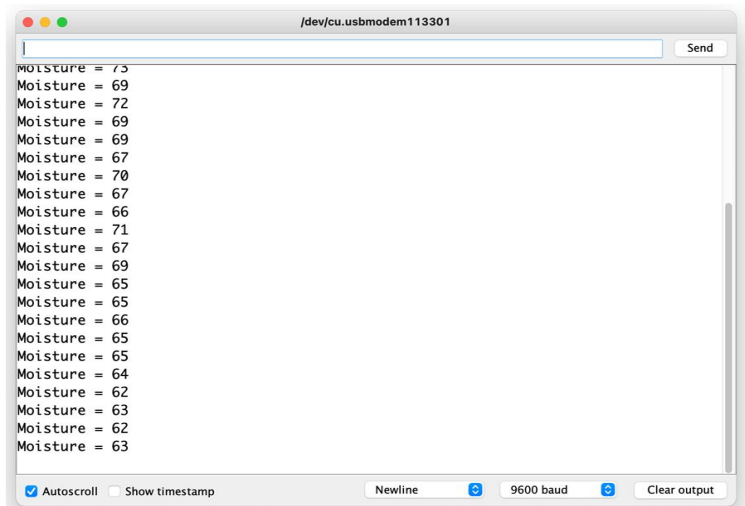
void loop()
{
  // Read the analog value from the soil moisture sensor
  int soilMoistureValue = analogRead(soilMoisturePin);

  // Map the analog value to a percentage value (0-100%)
  // Adjust these values based on your sensor and soil conditions
  int soilMoisturePercent = map(soilMoistureValue, 0, 1023, 0, 100);

  // Print the soil moisture percentage to the serial monitor
  Serial.print("Soil Moisture: ");
  Serial.print(soilMoisturePercent);
  Serial.println("%");

  // Add a delay to prevent spamming the serial monitor
  delay(1000); // Adjust delay as needed
}

```



Water Level Detection with Arduino

Aim:

To interface a water level sensor with an Arduino UNO board and detect the water level in a container or reservoir.

Introduction:

- A water level sensor is a vital component used to detect the presence and level of water in various applications, ranging from industrial processes to home automation systems.
- This program aims to interface a water level sensor with an Arduino UNO board to accurately detect water levels in a container or reservoir.

Step 1: Arduino to Sensor Connections:

- 1.1 Connect the water level sensor to the Arduino UNO board:
- 1.2 Connect the sensor's power pin to digital pin 7 (POWER_PIN).
- 1.3 Connect the sensor's signal pin to analog pin A5 (SIGNAL_PIN).
- 1.4 Connect the sensor's ground pin to the ground (GND) of the Arduino.

Step 2: Define the pin constants for power pin and signal pin (e.g., `#define POWER_PIN 7` and `#define SIGNAL_PIN A5`).

Step 3: Initialize serial communication for debugging purposes using `Serial.begin(9600)`.

Step 4: Set the pin mode of the power pin as OUTPUT and initialize it LOW using `pinMode()` and `digitalWrite()` functions.

Step 5: In the `loop()` function, activate the sensor by setting the power pin HIGH, wait for a brief period, read the analog value from the sensor using `analogRead()`, and store it in a variable.

Step 6: Deactivate the sensor by setting the power pin LOW after reading the analog value.

Step 7: Upload Code: Upload the Arduino code to the Arduino UNO board using the Arduino IDE.

Step 8: Testing: Open the Arduino Serial Monitor to observe the analog readings indicating the water level.

Step 9: Stop the Program

Hardware connections

1. For basic hardware connections, simply link the water level sensor's output to a microcontroller's analog input pin, and connect the sensor's power and ground pins to the appropriate power source.






5V	+
GND	-
A0	S

2. Connecting a water level sensor to hardware typically involves interfacing the sensor with a microcontroller or a development board. Here's a general guide on how to do it.

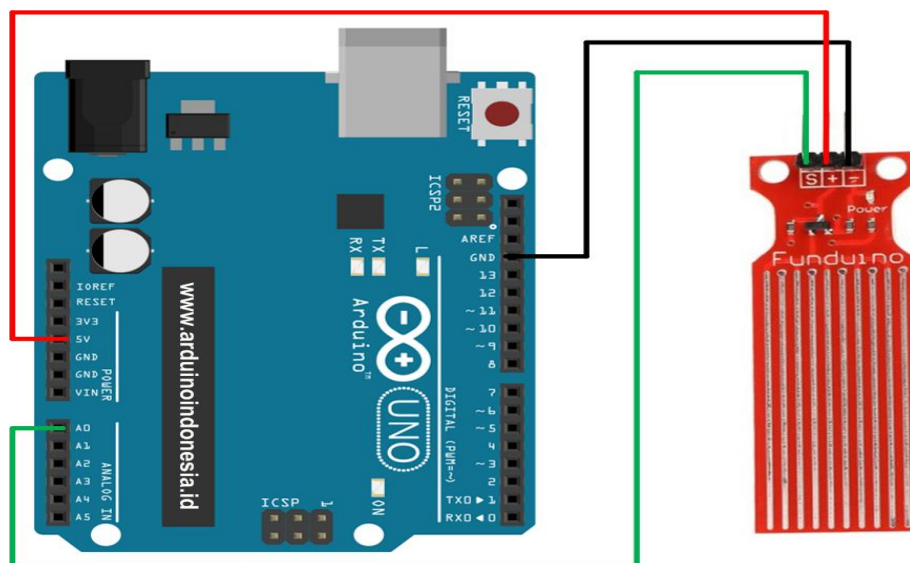
Result :

The water level detection system with Arduino has been successfully implemented, providing a reliable method for monitoring water levels in containers or reservoirs.

Parts required:

<u>Sl.no</u>	<u>Description</u>	<u>Image</u>	<u>Quantity</u>
1	Arduino board		1
2	Water Level indicator Sensor		1
4	External power source		1
5	Connecting power cable		1
6	Connecting wires		1

Circuit diagram



// CODE

```
#define POWER_PIN 7
#define SIGNAL_PIN A5

int value = 0;

void setup()
{
    Serial.begin(9600);
    pinMode(POWER_PIN, OUTPUT);
    digitalWrite(POWER_PIN, LOW);
}

void loop()
{
    digitalWrite(POWER_PIN, HIGH);
    delay(10);
    value = analogRead(SIGNAL_PIN);
    digitalWrite(POWER_PIN, LOW);
}
```

Water level: 0		The sensor was dry
Water level: 0		
Water level: 0		
Water level: 0		
Water level: 80		The sensor was partially immersed in water
Water level: 130		
Water level: 260		
Water level: 390		
Water level: 411		The sensor was fully immersed in water
Water level: 420		
Water level: 435		
Water level: 448		
Water level: 485		
Water level: 511		
Water level: 521		
Water level: 524		
Water level: 533		

Temperature Measurement Using LM35 Sensor with Arduino

Aim:

To interface an LM35 temperature sensor with an Arduino UNO board and measure temperature accurately.

Introduction:

- The LM35 temperature sensor is a precision integrated-circuit temperature device that provides an output voltage linearly proportional to the Centigrade temperature.
- This program aims to interface an LM35 temperature sensor with an Arduino UNO board to measure temperature and display the readings on the serial monitor.

Procedure

Step 1: Hardware Connections

- 1.1 Connect the LM35 temperature sensor to the Arduino UNO board:
- 1.2 Connect the sensor's middle pin to analog pin A0.
- 1.3 Connect the sensor's other pins to the appropriate power (5V) and ground (GND) sources.

Step 2: Define the analog pin connected to the sensor (e.g., `int sensor = A0;`).

Step 3: Initialize serial communication for debugging purposes using `Serial.begin(9600)`.

Step 4: In the `loop()` function, read the analog output from the LM35 sensor using `analogRead()`.

Step 5: Convert the analog reading to degrees Celsius using the formula provided in the code.

Step 6: Print the temperature information on the serial monitor using `Serial.print()` and `Serial.println()`.

Step 7: Add a delay to prevent spamming the serial monitor using `delay()`.

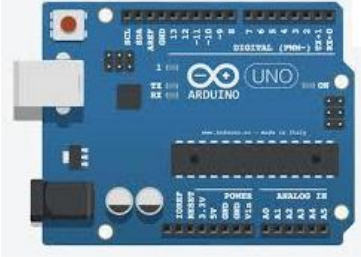
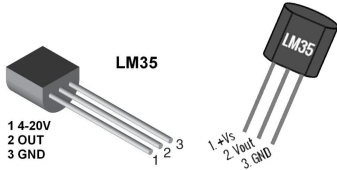



Step 8: Upload the Arduino code to the Arduino UNO board using the Arduino IDE.

Step 9: Testing: Open the Arduino Serial Monitor to observe real-time temperature readings in degrees Celsius.

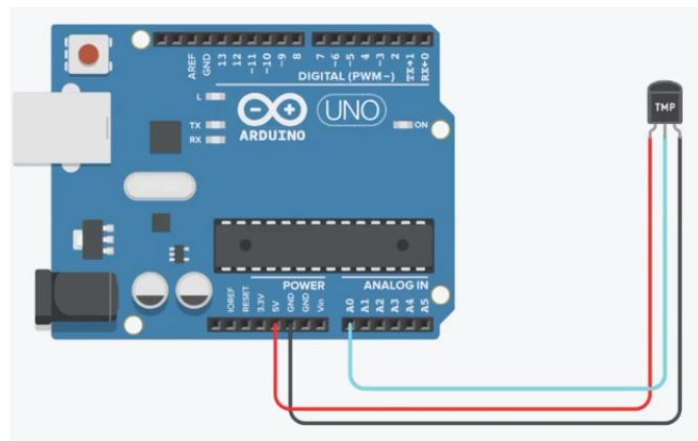
Result :

Real-time display of temperature readings in degrees Celsius on the Arduino Serial Monitor. Accurate measurement of temperature using the LM35 sensor interfaced with the Arduino UNO board.

Parts required:

<u>Sl.no</u>	<u>Description</u>	<u>Image</u>	<u>Quantity</u>
1	Arduino board		1
2	LM35 Temperature Sensor		1
4	External power source		1
5	Connecting power cable		1
6	Connecting wires		1

Circuit diagram



The screenshot shows the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The title bar reads 'LM35_TEMPERATURE_SENSOR | Arduino 1.6.4'. The main editor area contains the following code:

```

// LM35_TEMPERATURE_SENSOR

// LM35 TEMPERATURE SENSOR

float temp; //Define the temp float variable
int sensor = 0; // sensor middle pin on analog pin 0

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

void loop()
{
  temp = analogRead(sensor); //assigning the analog output to temp
  temp = temp * 0.48620125; //converting volts to degrees celsius ----- 0.48620125 = [ (5V

  //print information on the serial monitor
  Serial.print("The temperature is :");
  Serial.print(temp);
  Serial.println("deg. Celsius");

  //wait 1 second
  delay(1000);
}

```

The status bar at the bottom indicates 'Global variables use 238 bytes (11%) of dynamic memory, leaving 1,810 bytes for local variables. M'. The serial monitor on the right shows the output of the program, displaying the temperature in degrees Celsius, which fluctuates between approximately 25.39 and 25.88 degrees Celsius.

Smoke Detector/ Gas Sensor With Arduino

Aim:

To build a Smoke Detector Circuit using a gas sensor (MQ-2) interfaced with an Arduino board to detect the presence of smoke in the air and display the sensor readings in parts per million (PPM).

Introduction:

- Smoke detectors are essential safety devices used to detect smoke or fire in buildings, ensuring timely evacuation and fire suppression.
- This Program involves building a Smoke Detector Circuit using a gas sensor (specifically the MQ-2) interfaced with an Arduino board.
- The circuit not only senses smoke in the air but also reads and displays the level of smoke in parts per million (PPM).

Procedure:

Step 1: Hardware Connections:

- 1.1 Connect the gas sensor to the Arduino board:
- 1.2 Connect the sensor's signal pin to analog pin A0 (MQ2pin).
- 1.3 Connect the sensor's power and ground pins to the appropriate power and ground sources.

Step 2: Define the analog pin connected to the sensor (e.g., #define MQ2pin A0).

Step 3: Initialize serial communication for debugging purposes using Serial.begin(9600).

Step 4: In the setup() function, print a message indicating the gas sensor warming up and delay for 20 seconds to allow the MQ-2 sensor to warm up.

Step 5: In the loop() function, read the analog output from the gas sensor using analogRead() and store the value in a variable.

Step 6: Check if the sensor value exceeds a certain threshold (e.g., 200) to determine if smoke is detected.

Step 7: Print the sensor value and a smoke detection message to the serial monitor using Serial.print() and Serial.println().

Step 8: Add a delay to prevent rapid readings and provide stability to the system using delay().

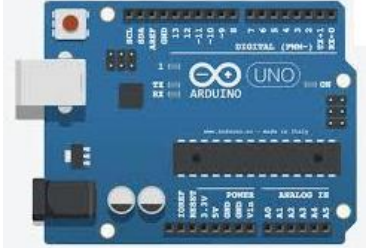




Step 9: Upload Code: Upload the Arduino code to the Arduino board using the Arduino IDE.

Step 10: Testing: Open the Arduino Serial Monitor to observe real-time gas sensor readings.

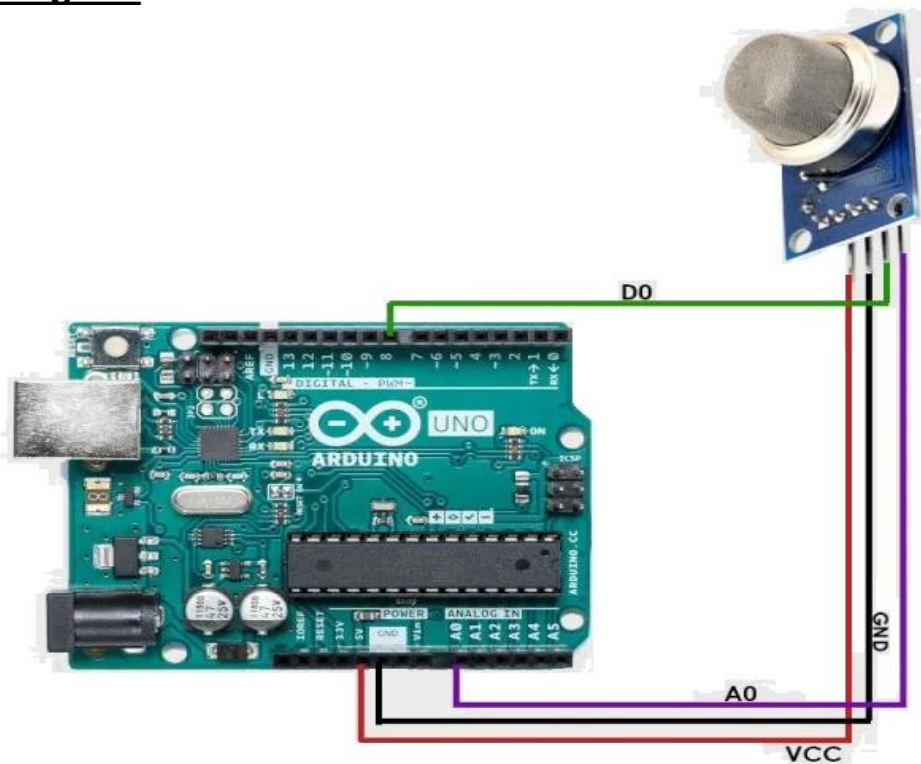
Result :

The Smoke Detector Circuit with Gas Sensor and Arduino has been successfully implemented, demonstrating its capability to detect smoke in the air.

Parts required:

<u>Sl.no</u>	<u>Description</u>	<u>Image</u>	<u>Quantity</u>
1	Arduino board		1
2	Gas Sensor		1
4	External power source		1
5	Connecting power cable		1
6	Connecting wires		1

Circuit diagram

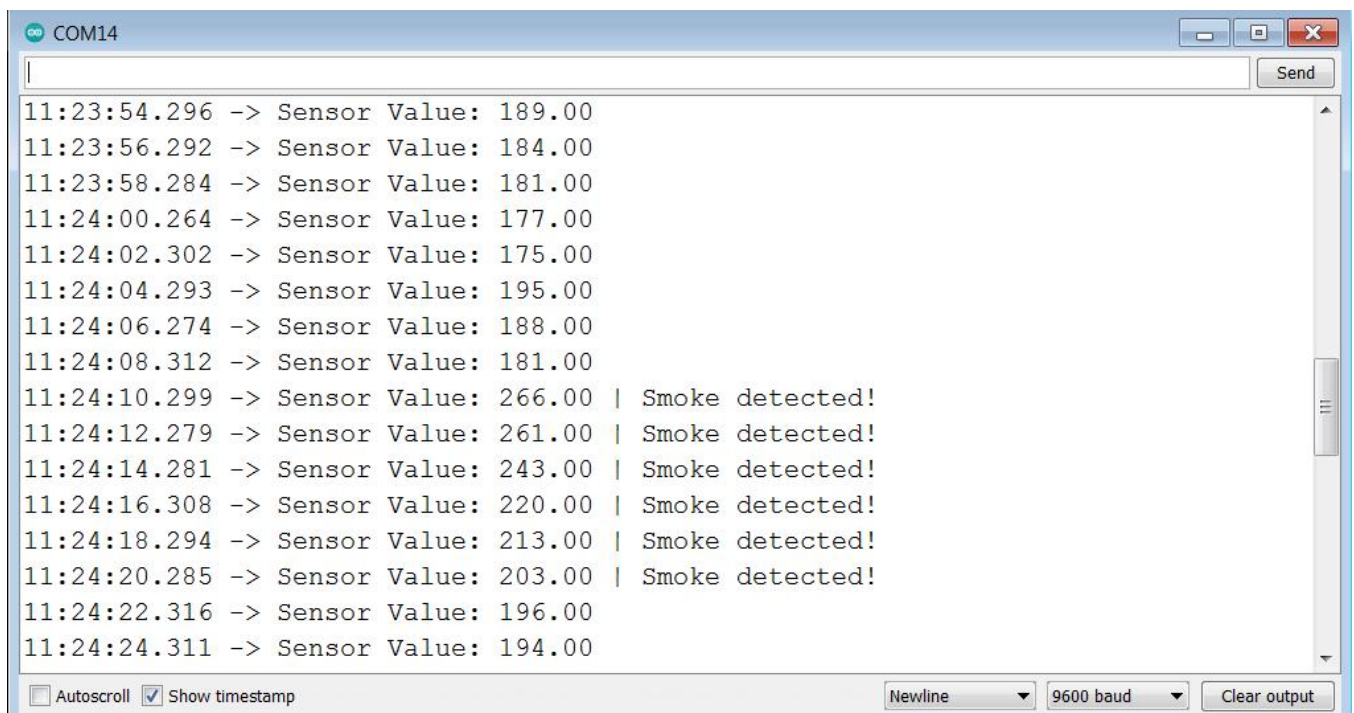


```
#define MQ2pin 0
float sensorValue; //variable to store sensor value
void setup()
{
  Serial.begin(9600); // sets the serial port to 9600
  Serial.println("Gas sensor warming up!");
  delay(20000); // allow the MQ-2 to warm up
}

void loop()
{
  sensorValue = analogRead(MQ2pin); // read analog input pin 0
  Serial.print("Sensor Value: ");
  Serial.print(sensorValue);

  if(sensorValue > 200)
  {
    Serial.print(" | Smoke detected!");
  }

  Serial.println("");
  delay(2000); // wait 2s for next reading
}
```



The screenshot shows a serial monitor window titled "COM14" with a "Send" button in the top right corner. The main area displays the following output:

```
11:23:54.296 -> Sensor Value: 189.00
11:23:56.292 -> Sensor Value: 184.00
11:23:58.284 -> Sensor Value: 181.00
11:24:00.264 -> Sensor Value: 177.00
11:24:02.302 -> Sensor Value: 175.00
11:24:04.293 -> Sensor Value: 195.00
11:24:06.274 -> Sensor Value: 188.00
11:24:08.312 -> Sensor Value: 181.00
11:24:10.299 -> Sensor Value: 266.00 | Smoke detected!
11:24:12.279 -> Sensor Value: 261.00 | Smoke detected!
11:24:14.281 -> Sensor Value: 243.00 | Smoke detected!
11:24:16.308 -> Sensor Value: 220.00 | Smoke detected!
11:24:18.294 -> Sensor Value: 213.00 | Smoke detected!
11:24:20.285 -> Sensor Value: 203.00 | Smoke detected!
11:24:22.316 -> Sensor Value: 196.00
11:24:24.311 -> Sensor Value: 194.00
```

At the bottom of the window, there are checkboxes for "Autoscroll" (unchecked) and "Show timestamp" (checked). To the right of these are dropdown menus for "Newline" and "9600 baud", and a "Clear output" button.