JAGAN INSTITUTE OF MANAGEMENT STUDIES

Sector – 5, Rohini, New Delhi

(Affiliated to)

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

SECTOR – 16 C, DWARKA, NEW DELHI

INTERNET OF THINGS PRACTICAL FILE

Submitted To:

Ms. Deepika

Submitted By:

Name: Akshat Gangi

Roll No: 04450402021

BCA-VI, IInd shift

INDEX

S.NO	PRACTICAL	DATE	SIGN
1	To study IOT and Arduino.	16/02/2024	
2	To study and install IDE of Arduino.	16/02/2024	
3	Write a program using Arduino to blink the LED.	01/03/2024	
4	Write a program using Arduino to run a DC motor.	01/03/2024	
5	Make a circuit to print your name on LCD.	15/03/2024	
6	Make a circuit to measure and display temperature using a TMP36 temperature sensor.	15/03/2024	
7	Write a program to show how to fade a LED using analogWrite () function.	22/03/2024	
8	Write a program to for Arduino by using ultrasonic sensor for distance measurement	22/03/2024	
9	Write a program using for detecting objects using PIR sensor in Arduino.	12/04/2024	
10	Write a program to interface 1 digit 7 segment display with Arduino.	12/04/2024	
11	Hardware implementation of IOT Experiments	19/04/2024	
12	To install libraries in Arduino IDE	19/04/2024	
13	Perform an Experiment in Raspberry Pi	26/04/2024	
14	Perform an Experiment in Raspberry Pi	26/04/2024	
15	Application Based Experiment	03/05/2024	

PRACTICAL 1

AIM: To study IOT and ARDUINO

INTRODUCTION TO IOT

IOT stands for Internet of Things. It refers to the interconnectedness of physical devices, such as appliances and vehicles that are embedded with software, sensors, and connectivity which enables these objects to connect and exchange data.

This technology allows for the collection and sharing of data from a vast network of devices, creating opportunities for more efficient and automated systems.

Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a few of the categorical examples where IOT is strongly established.

CHARACTERSTICS OF IOT:

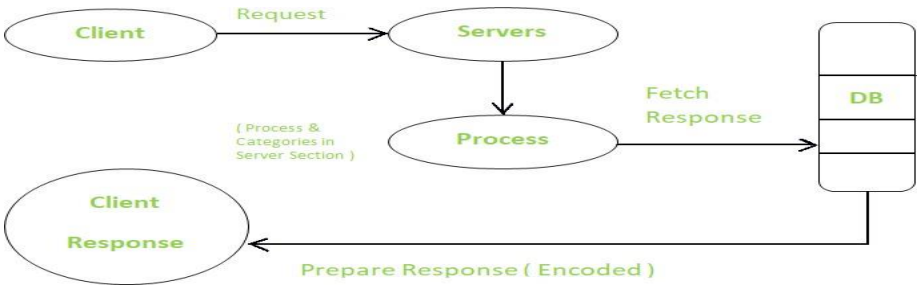
- Dynamic and self-adapting
- Self-configuring
- Unique identity
- Interconnectivity
- Heterogeneity
- Things related services
- Safety and privacy
- Interoperable communication protocols

COMMUNICATION IN IOT:

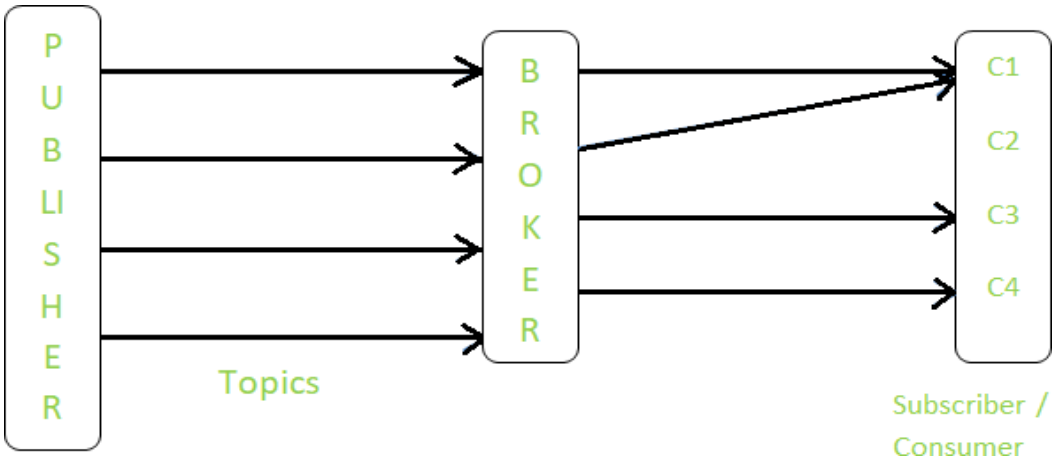
IoT devices are found everywhere and will enable circulatory intelligence in the future. For operational perception, it is important and useful to understand how various IoT devices communicate with each other.

Types of Communication Model:

1. **Request & Response Model –**
   In **Request-Response** communication model client sends a request to the server and the server responds to the request. When the server receives the request, it decides how to respond, fetches the data retrieves resources, and prepares the response, and sends it to the client.

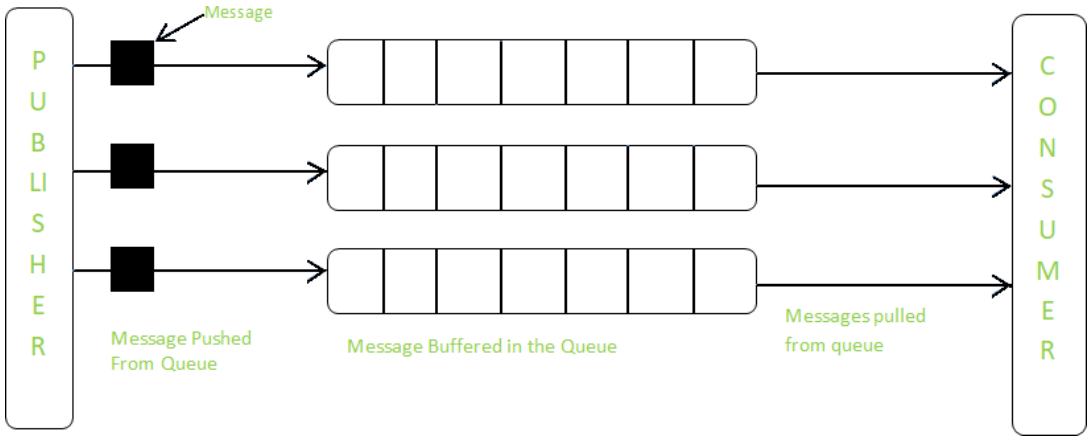


2. **Publisher-Subscriber Model –**
   This model comprises three entities: Publishers, Brokers, and Consumers.
   - **Publishers** are the source of data. It sends the data to the topic which are managed by the broker. They are not aware of consumers.
   - **Consumers** subscribe to the topics which are managed by the broker.
   - Hence, **Brokers** responsibility is to accept data from publishers and send it to the appropriate consumers. The broker only has the information regarding the consumer to which a particular topic belongs to which the publisher is unaware of.

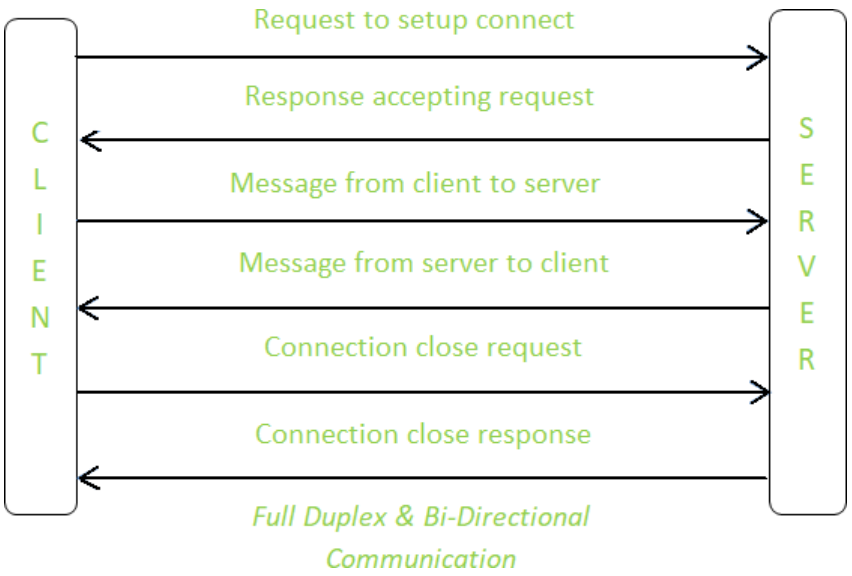


3. **Push-Pull Model** –
The push-pull model constitutes data publishers, data consumers, and data queues.

- **Publishers** and **Consumers** are not aware of each other.
- Publishers publish the message/data and push it into the queue. The consumers, present on the other side, pull the data out of the queue. Thus, the queue acts as the buffer for the message when the difference occurs in the rate of push or pull of data on the side of a publisher and consumer.
- **Queues** help in decoupling the messaging between the producer and consumer. Queues also act as a buffer which helps in situations where there is a mismatch between the rate at which the producers push the data and consumers pull the data.



4. **Exclusive Pair-**
   - **Exclusive Pair** is the bi-directional model, including full-duplex communication among client and server. The connection is constant and remains open till the client sends a request to close the connection.
   - The **Server** has the record of all the connections which has been opened.
   - This is a state-full connection model and the server is aware of all open connections.
   - Web Socket based communication API is fully based on this model.



APPLICATIONS OF IOT:

1. Wearables

Wearable technology is the hallmark of IoT applications and one of the earliest industries to deploy IoT. We have fit bits, heart rate monitors and smartwatches these days.

Guardian glucose monitoring device has been developed to help people with diabetes. It detects glucose levels in our body, uses a small electrode called the glucose sensor under the skin, and relates it to a radiofrequency monitoring device.

2. Smart Home Applications

The smart home is probably the first thing when we talk about the IoT application. The example we see the AI home automation is employed by **Mark Zuckerberg. Alan Pan's** home automation system, where a string of musical notes uses in-house functions.

3. Smart Cities

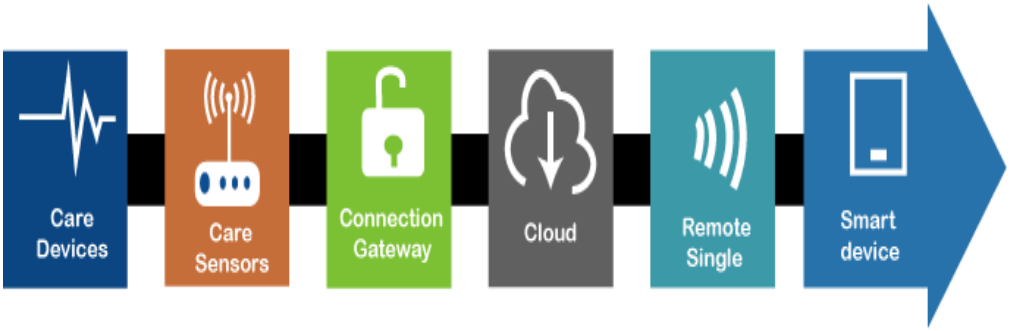
Governments and engineers use the Internet of Things to analyse the complex factors of town and each city. IoT applications help in the area of water management, waste control and emergencies.

Example of a smart city - Palo Alto.

Palo Alto, San Francisco, is the first city to acquire the traffic approach. He realized that most cars roam around the same block on the streets in search of parking spots. It is the primary cause of traffic congestion in the city. Thus, the sensors were installed at all parking areas in the city. These sensors pass occupancy status to the cloud of each spot.

4. Health care

IoT applications can transform reactive medical-based systems into active wellness-based systems. Resources that are used in current medical research lack important real-world information. It uses controlled environments, leftover data, and volunteers for clinical trials. The **Internet of Things** improves the device's **power, precision** and **availability**. IoT focuses on building systems rather than just tools. Here's how the IoT-enabled care device works.

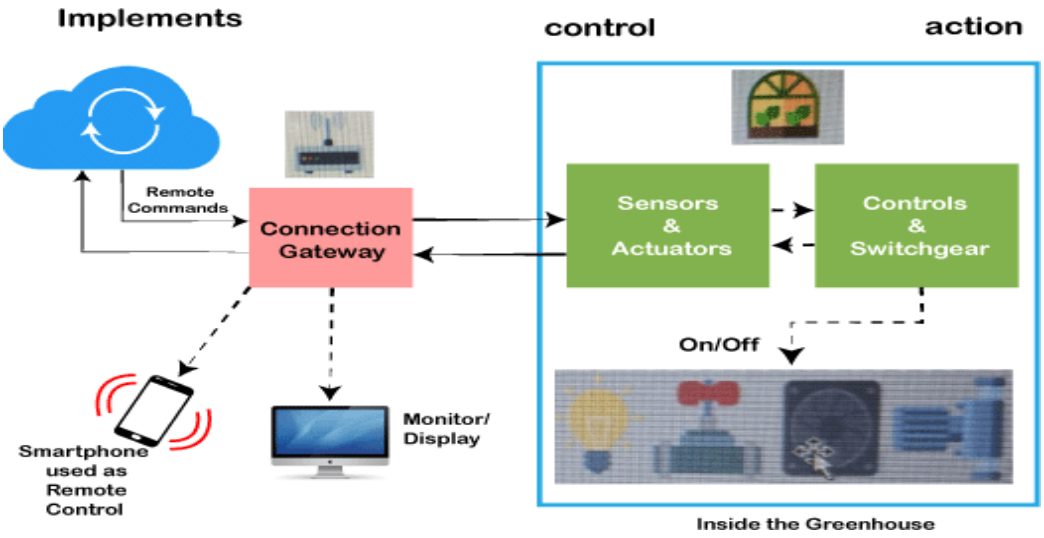


5. Agriculture

By the year **2050**, the world's growing population is estimated to have reached about 10 billion. To feed such a large population, agriculture needs to marry technology and get the best results. There are many possibilities in this area. One of them is Smart Greenhouse.

Farming techniques grow crops by **environmental parameters**. However, manual handling results in production losses, energy losses and labor costs, making it less effective.

The greenhouse makes it easy to monitor and enables to control the climate inside it.



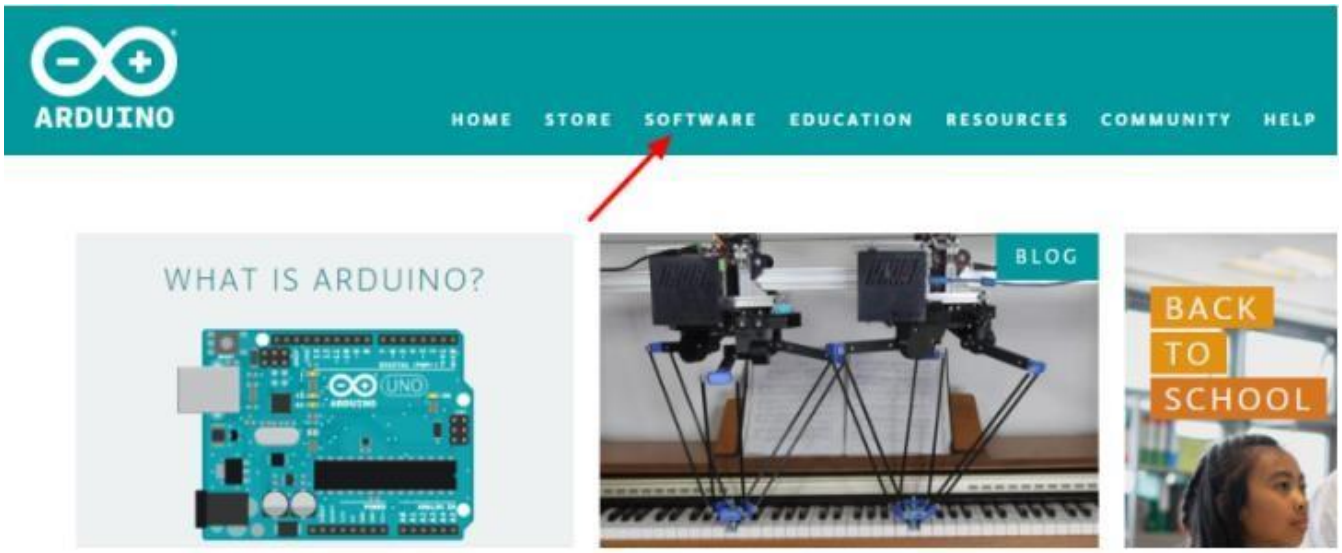
PRACTICAL 2

AIM: To study and install IDE of ARDUINO

STEP 1

Open the browser and search Arduino software on the Google search engine. You will find the official software link to download. Or you can directly visit link https://www.arduino.cc.

Click on the **software** menu option.

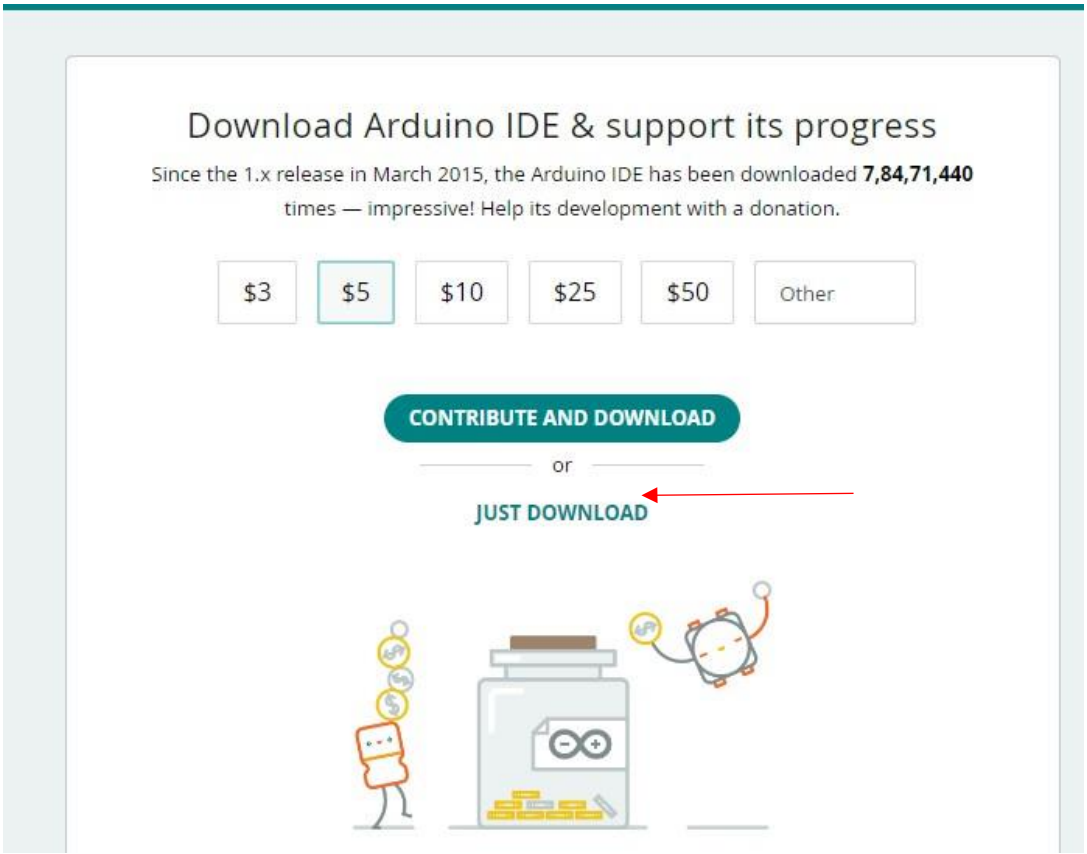


STEP 2

Select the **Download** submenu option under the **software** menu.
Based on your operating system, choose the installer.

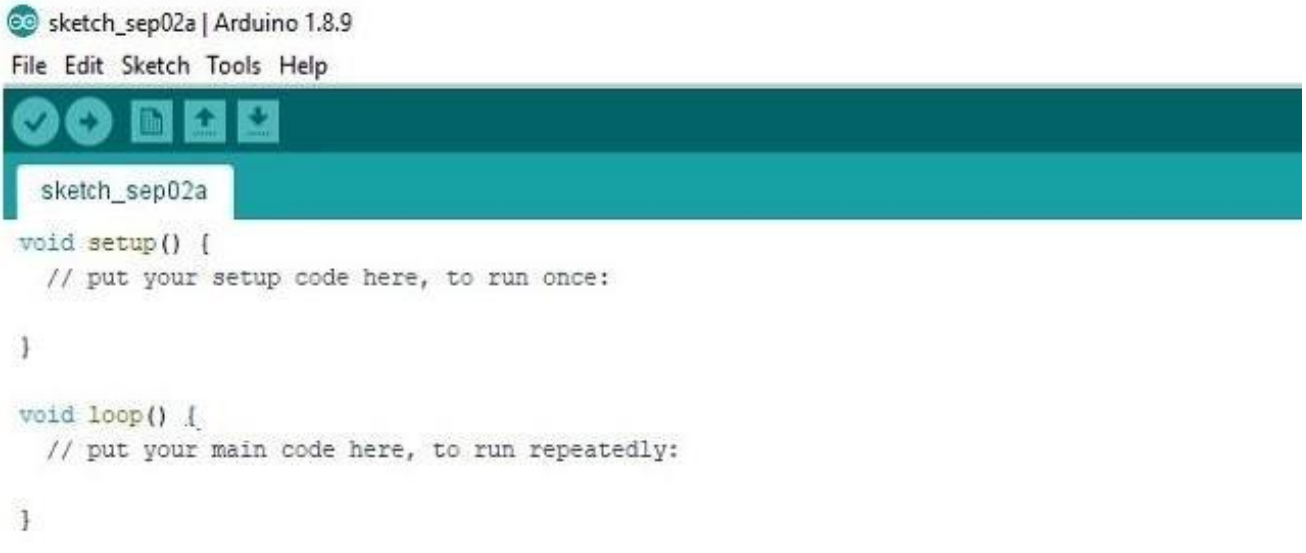
STEP 3

If you want to use it for FREE, click on **JUST DOWNLOAD** option.



STEP 4

After the successful software installation, open the installed Arduino software. This will look like as below picture where you can edit code.



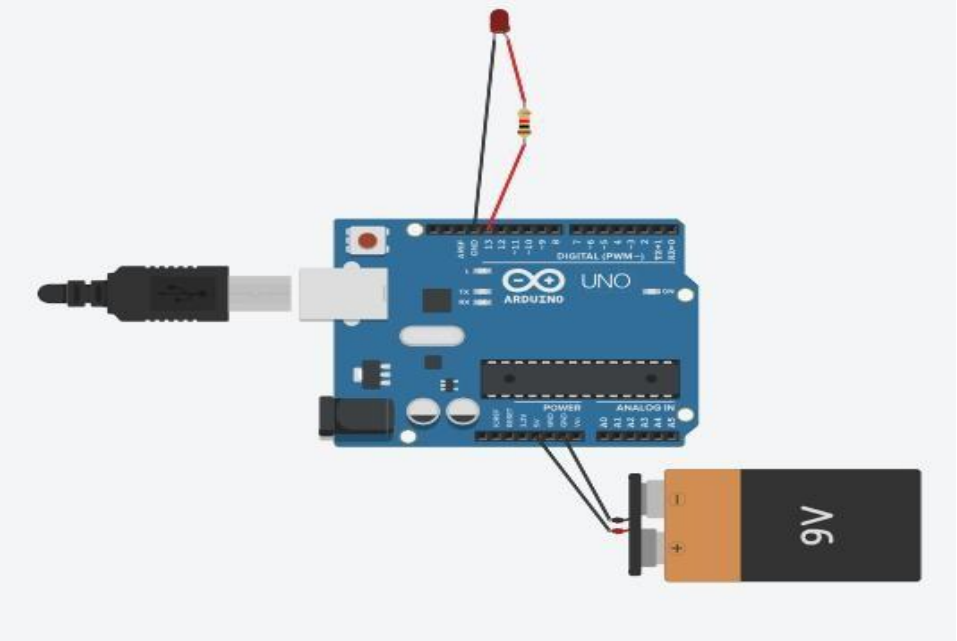
PRACTICAL 3

AIM: Write a program using Arduino to blink the LED

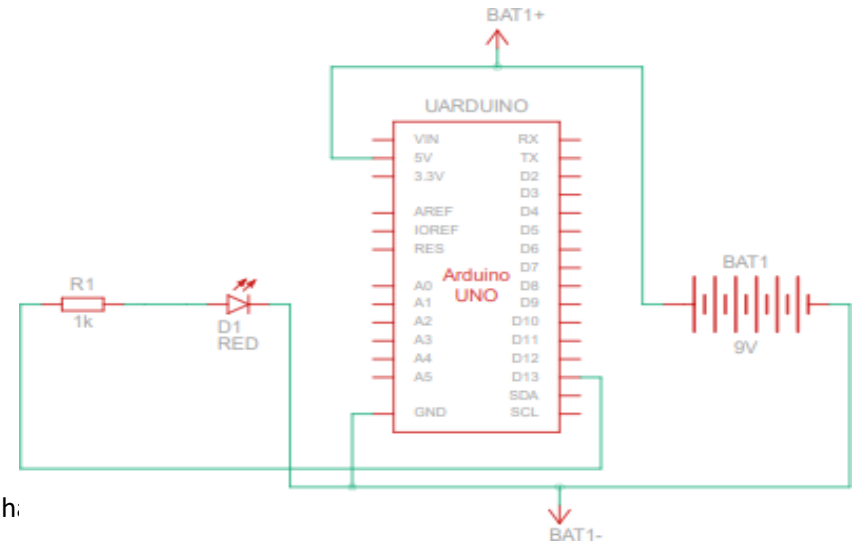
INTRODUCTION

Using the Arduino platform, a versatile and accessible microcontroller system, we can easily create programs to control various electronic components, including LEDs. This introductory project will serve as a foundation for learning more complex tasks and experimenting with different sensors and actuators.

CIRCUIT DIAGRAM



SCHEMATIC DIAGRAM



CODE

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

void loop ()
{
 digitalWrite(LED_BUILTIN, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
 digitalWrite(LED_BUILTIN, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
}
```

KEY COMPONENTS

Name	Quantity	Component
Uarduino	1	Arduino Uno R3
D1	1	Red LED
R1	1	1 kΩ Resistor
BAT1	1	9V Battery

STEPS OF WORKING

1. Assemble the Circuit: Connect the LED to the Arduino board, with the longer leg (anode) to pin 13 and the shorter leg (cathode) to ground (GND).

2. Set up the Arduino IDE: Install and open the Arduino IDE. Ensure the correct board and port are selected.

3. Write the Code: Create a new sketch and write code to blink the LED using digitalWrite() and delay() functions.

4. Upload the Code: Connect the Arduino to your computer and click the upload button in the IDE to compile and upload the code.

5. Observe the LED: Once uploaded, the LED will blink on and off at 1-second intervals.

6. Experiment: Modify the code to change the blink rate or try different patterns. Double-check connections if the LED doesn't blink as expected.

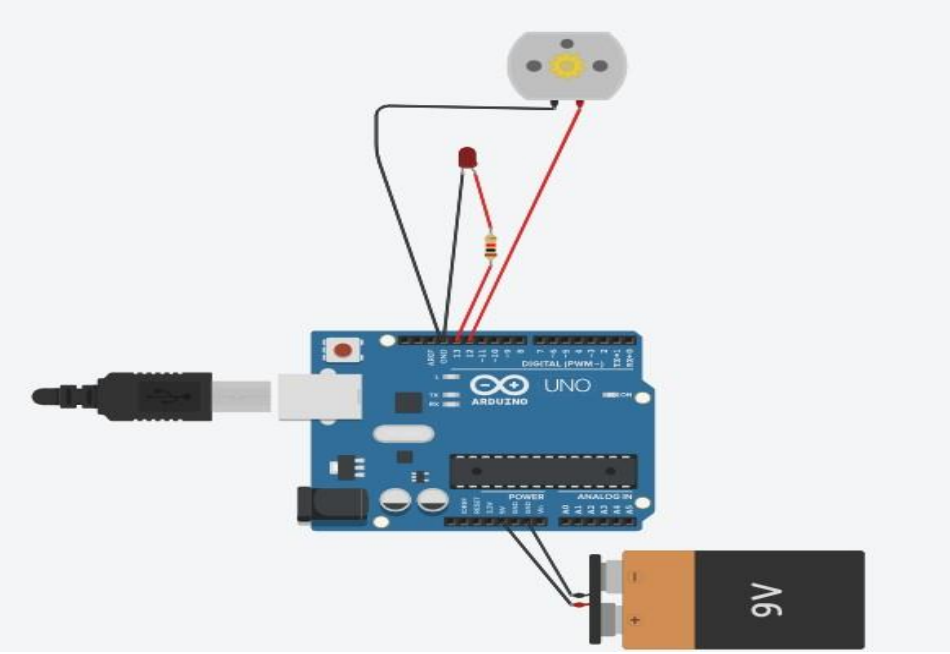
PRACTICAL 4

AIM: Write a program using Arduino to run a DC Motor

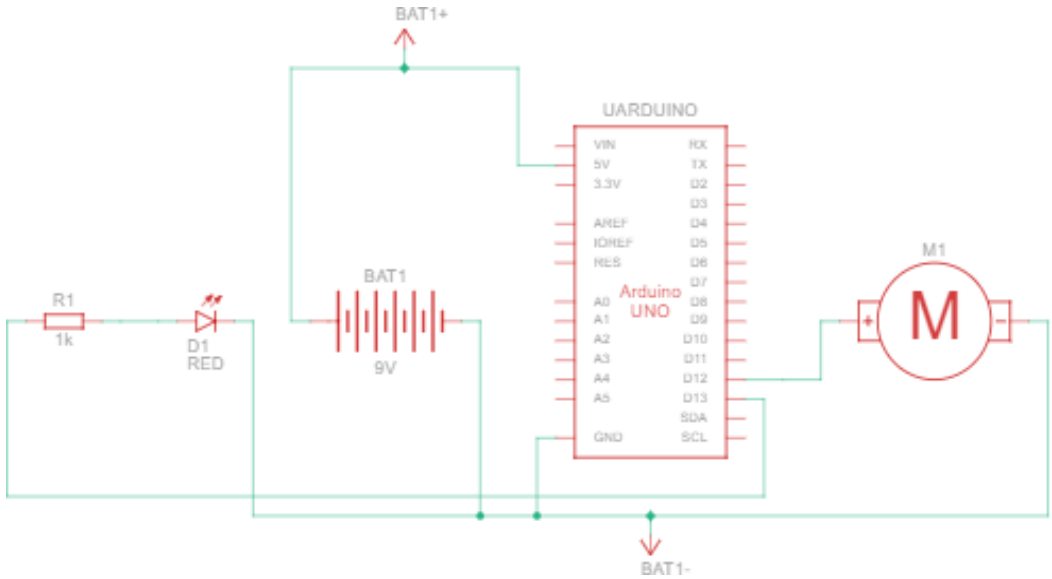
INTRODUCTION

Using the Arduino platform, a versatile and accessible microcontroller system, we can easily create programs to control various electronic components, including DC Motor. This introductory project will serve as a foundation for learning more complex tasks and experimenting with different sensors and actuators.

CIRCUIT DIAGRAM



SCHEMATIC DIAGRAM



CODE

```
void setup()
{
 pinMode(LED_BUILTIN, OUTPUT);
 pinMode(12, OUTPUT);
}
void loop()
{
 digitalWrite(LED_BUILTIN, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
 digitalWrite(LED_BUILTIN, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
 digitalWrite(12, HIGH);
 delay(2000); // Wait for 1000 millisecond(s)
 digitalWrite(12, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
}
```

KEY COMPONENTS

Name	Quantity	Component
Uarduino	1	Arduino Uno R3
D1	1	Red LED
R1	1	1 kΩ Resistor
BAT1	1	9V Battery
M1	1	DC Motor

STEPS OF WORKING

1. Assemble the Circuit: Connect the DC motor to your Arduino board using an H-bridge motor driver. Ensure proper connections between the motor, motor driver, and Arduino according to the motor driver's datasheet.

2. Set up the Arduino IDE: Install the Arduino IDE on your computer if you haven't already. Open the IDE and ensure you have selected the correct board and port from the Tools menu.

3. Write the Code: In the Arduino IDE, create a new sketch (File > New). In the code, you'll need to define the pins connected to the motor driver's input pins and the pin connected to the motor driver's enable pin. You'll also need to write code to control the direction and speed of the motor by sending appropriate signals to the motor driver.

4. Upload the Code: Connect your Arduino board to your computer using a USB cable. Click the upload button in the Arduino IDE to compile and upload the code to the Arduino board.

5. Observe the Motor: Once the code is uploaded successfully, the DC motor will start rotating according to the specified direction and speed in the code.

6. Experiment: Modify the code to change the motor direction or speed. You can also add additional features such as controlling the motor based on sensor inputs or integrating it into a larger project.

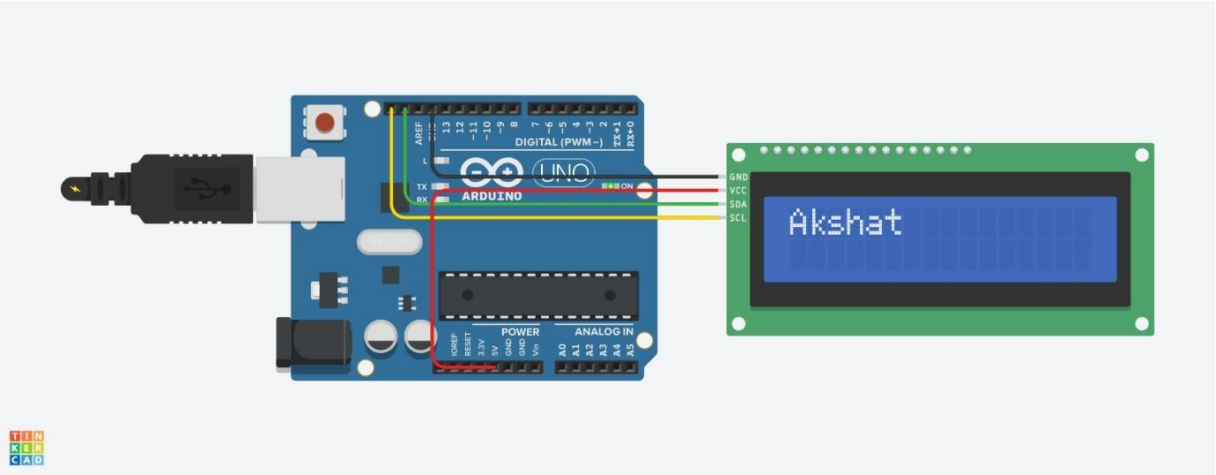
PRACTICAL 5

AIM: Make a circuit to print your name on LCD

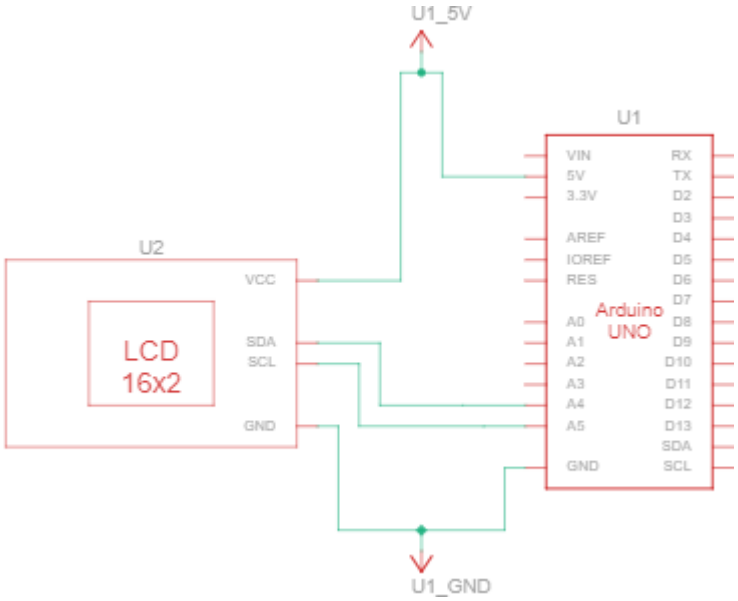
INTRODUCTION

we aim to create a circuit that prints your name on an LCD (Liquid Crystal Display) using an Arduino board. LCDs are widely used for displaying text and graphics in various electronic devices, making them a fundamental component in many projects. By interfacing an LCD with an Arduino, we can control what is displayed on the screen, allowing us to showcase personalized messages, data, or even sensor readings.

CIRCUIT DIAGRAM



SCHEMATIC DIAGRAM



CODE

```
#include<Adafruit_LiquidCrystal.h>

Adafruit_LiquidCrystal lcd_1(0);
void setup(){
        lcd_1.begin(16,2);
        lcd_1.print("Akshat");
}
void loop(){
        lcd_1.setCursor(0,1);
        lcd_1.setBacklight(1);
        delay(500);
        lcd_1.setBacklight(0);
        delay(500);
 }
```

KEY COMPONENTS

Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	MCP23008-based, 32 LCD 16 x 2 (I2C)

STEPS OF WORKING

1. Assemble the Circuit: Connect the components required for the circuit, including the Arduino board, an LCD (Liquid Crystal Display), and a potentiometer (for contrast adjustment). Wire the LCD to the Arduino according to the pinout provided in your LCD datasheet or tutorial.

2. Set up the Arduino IDE: Install the Arduino IDE on your computer if you haven't already. Open the IDE and ensure you have selected the correct board and port from the Tools menu.

3. Write the Code: In the Arduino IDE, create a new sketch (File > New). Write the code to initialize the LCD, set up the necessary pins, and print your name on the screen. You'll need to include the LiquidCrystal library and use functions such as lcd.begin() and lcd.print() to control the display.

4. Upload the Code: Connect your Arduino board to your computer using a USB cable. Click the upload button (arrow icon) in the Arduino IDE to compile and upload the code to the Arduino board.

5. Observe the LCD: Once the code is uploaded successfully, your name should appear on the LCD screen. Adjust the potentiometer to adjust the contrast of the display if necessary.

6. Experiment: Modify the code to print different messages or add additional features such as scrolling text or displaying sensor readings. Explore the capabilities of the LCD and Arduino to create customized displays.

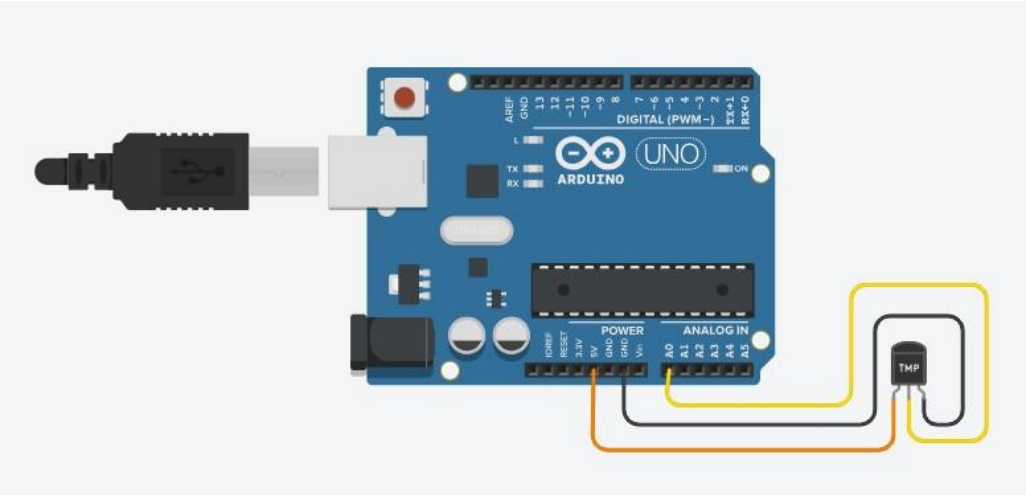
PRACTICAL 6

AIM: Make a circuit to measure and display temperature using a TMP36 temperature sensor.

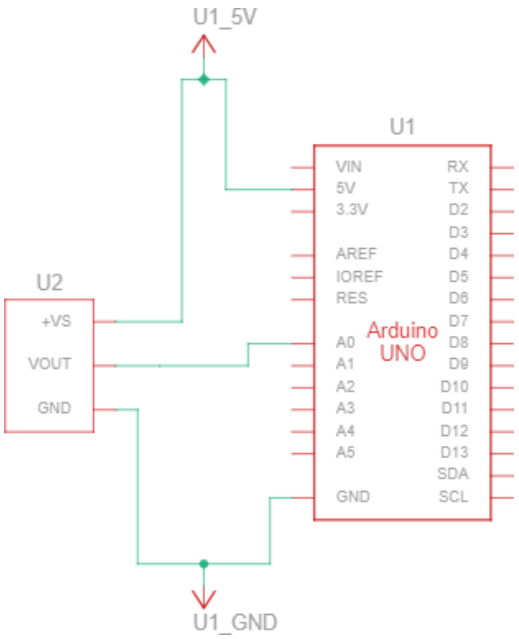
INTRODUCTION

This Arduino project involves interfacing a TMP36 temperature sensor with an Arduino board to measure temperature. The TMP36 sensor provides analog voltage output corresponding to temperature, which is then converted to Celsius and Fahrenheit readings. These temperature values are then serially printed to the Arduino IDE serial monitor, providing real-time temperature data.

CIRCUIT DIAGRAM



SCHEMATIC DIAGRAM



CODE

```
#define sensorPin A0

void setup() {
  // Begin serial communication at 9600 baud rate
  Serial.begin(9600);
}

void loop() {
  // Get the voltage reading from the TMP36
  int reading = analogRead(sensorPin);

  // Convert that reading into voltage
  // Replace 5.0 with 3.3, if you are using a 3.3V Arduino
  float voltage = reading * (5.0 / 1024.0);

  // Convert the voltage into the temperature in Celsius
  float temperatureC = (voltage - 0.5) * 100;

  // Print the temperature in Celsius
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.print("\xC2\xB0"); // shows degree symbol
  Serial.print("2C |  ");

  // Print the temperature in Fahrenheit
  float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
  Serial.print(temperatureF);
  Serial.print("\xC2\xB0"); // shows degree symbol
  Serial.println("F");

  delay(1000); // wait a second between readings
```

KEY COMPONENTS

Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	Temperature Sensor [TMP36]

STEPS OF WORKING

1. **Setup**: Connect the TMP36 temperature sensor to the Arduino board. The Vout pin of the TMP36 should be connected to an analog pin on the Arduino (e.g., A0).

2. **Initialization**: Begin serial communication at a baud rate of 9600 in the setup() function to enable communication between the Arduino and the computer.

3. **Reading Temperature**: In the loop() function, read the analog voltage output from the TMP36 sensor using analogRead(). Store the reading in a variable.

4. **Voltage Conversion**: Convert the analog reading to voltage using the formula: voltage = reading * (5.0 / 1024.0). Replace 5.0 with 3.3 if using a 3.3V Arduino

5. **Temperature Calculation**: Convert the voltage to temperature in Celsius using the formula: temperatureC = (voltage - 0.5) * 100.

6. **Display Temperature**: Print the temperature in Celsius to the serial monitor using Serial.print() function. Optionally, print the temperature in Fahrenheit by converting Celsius to Fahrenheit using the formula: temperatureF = (temperatureC * 9.0 / 5.0) + 32.0.

7. **Delay**: Add a delay of 1000 milliseconds (1 second) between readings using the delay() function to prevent rapid updates and ensure stability

8. **Observation**: Open the serial monitor in the Arduino IDE to observe the temperature readings displayed in Celsius and Fahrenheit.

9. **Analysis** and **Modification**: Analyze the temperature readings and make adjustments as necessary. Modify the code to add additional functionality or customize the output format..

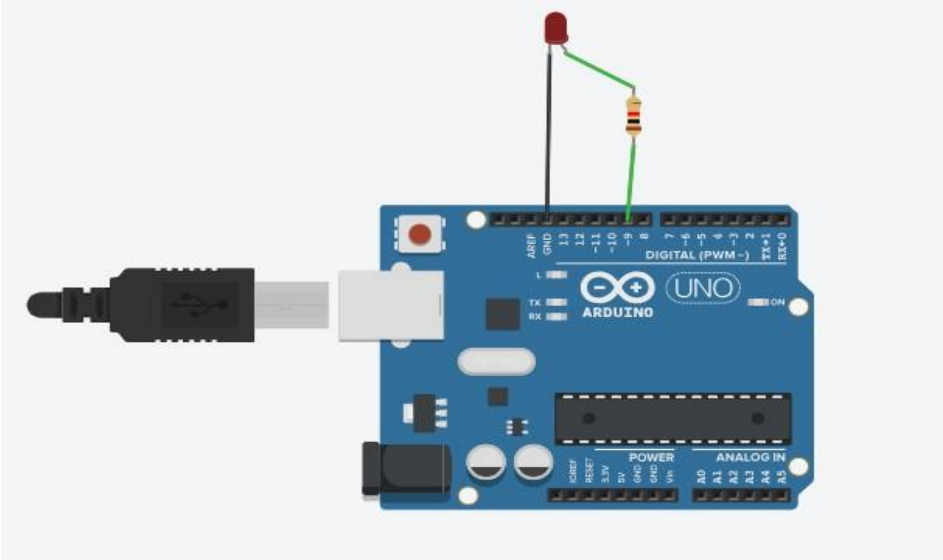
PRACTICAL 7

AIM: Write a program to show how to fade a LED using analog write function.

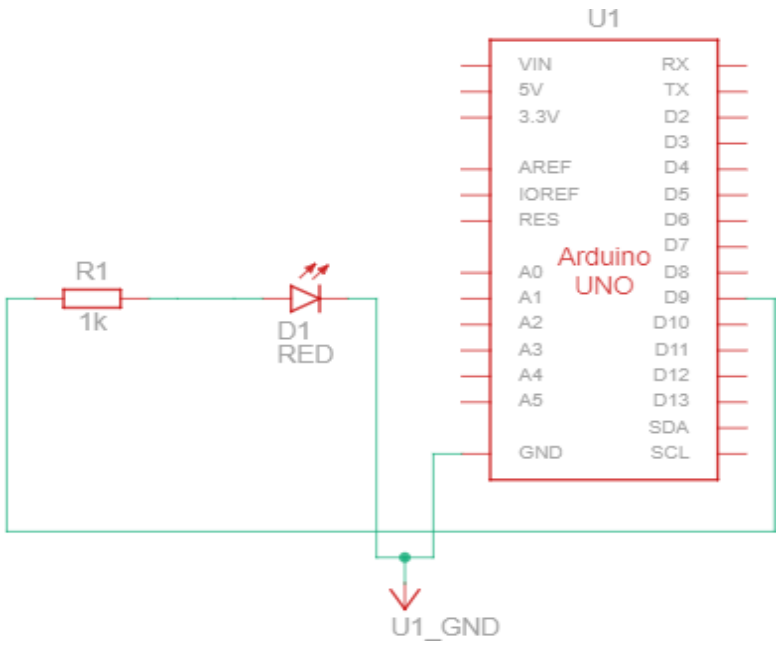
INTRODUCTION

Using Arduino platform, we can easily create programs to control various electronic components, including LEDs. In this program we aim to create a circuit that fade a LED using an Arduino board.

CIRCUIT DIAGRAM



SCHEMATIC DIAGRAM



CODE

```
int brightness = 0;

void setup()
{
  pinMode(9, OUTPUT);
}

void loop()
{
  for (brightness = 0; brightness <= 255; brightness += 5) {
    analogWrite(9, brightness);
    delay(30); // Wait for 30 millisecond(s)
  }
  for (brightness = 255; brightness >= 0; brightness -= 5) {
    analogWrite(9, brightness);
    delay(30); // Wait for 30 millisecond(s)
  }
}
```

KEY COMPONENTS

Name	Quantity	Component
U1	1	Arduino Uno R3
D1	1	Red LED
R1	1	1 kΩ Resistor

STEPS OF WORKING:

1. Build the circuit: Connect the led to the Arduino board, with the longer leg (anode) to pin 9 and the shorter leg (cathode) to ground GND.
2. Set up the Arduino IDE: Install and open the Arduino IDE. Ensure the correct board and port are selected.
3. Write the code: Create a new sketch and write code to fade the LED using analogWrite () and delay () function.
4. Upload the code: connect the Arduino to your computer and click the upload button on the IDE to compile and upload the code.
5. Observe the LED: Once uploaded, the LED will fade after 30 milliseconds.

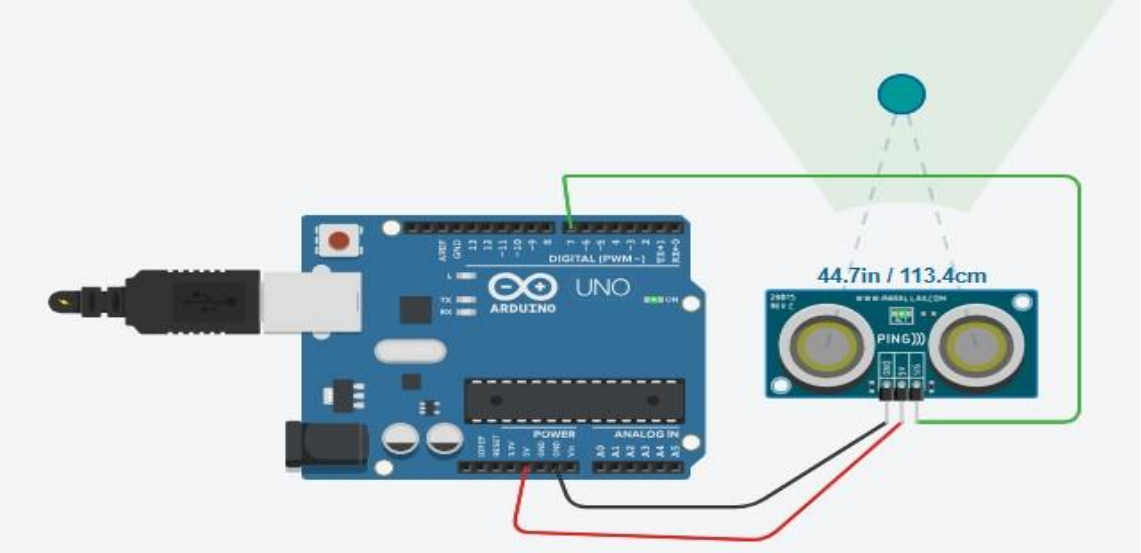
PRACTICAL 8

AIM: Write a program to for Arduino by using ultrasonic sensor for distance measurement

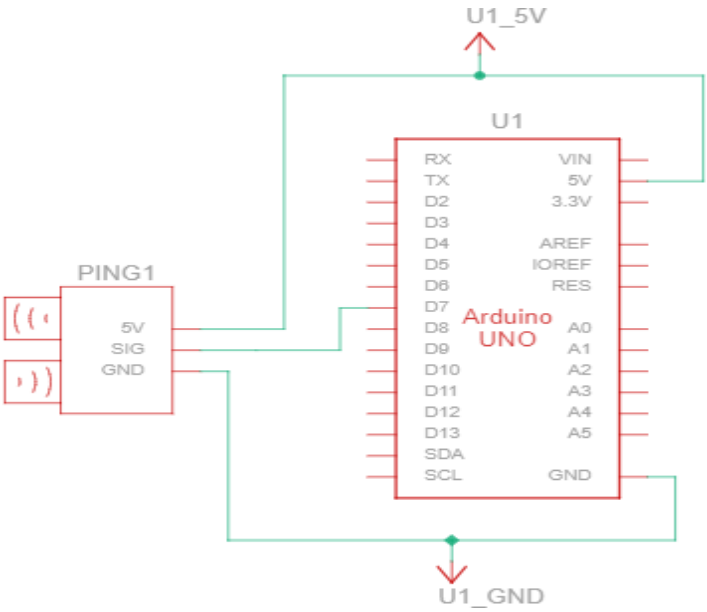
INTODUCTION

Using Arduino platform, we can easily create programs to control various electronic components, including Ultrasonic Distance Sensor. An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. In this experiment we aim to learn how ultrasonic distance senor measures and senses the distance.

CIRCUIT DIAGRAM



SCHEMATIC DIAGRAM



CODE

```
int inches = 0;
int cm = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT); // Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);

  // Sets the trigger pin to HIGH state for 10 microseconds

 digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds
  return pulseIn(echoPin, HIGH);
}

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

}

void loop()
{
  // measure the ping time in cm

 cm = 0.01723 * readUltrasonicDistance(7, 7);

  // convert to inches by dividing by 2.54
  inches = (cm / 2.54);

  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.println("cm");

  delay(100); // Wait for 100 millisecond(s)
}
```

KEY COMPONENTS

Name	Quantity	Component
U1	1	Arduino Uno R3
PING1	1	Ultrasonic Distance Sensor

STEPS OF WORKING

1. Build the circuit: Connect the Ultrasonic Distance Sensor pins to the Arduino board. Connect Ground pin to GND pin of anode, Power to the 5V pin of anode and Signal to the 7V pin.
2. Set up the Arduino IDE: Install and open the Arduino IDE. Ensure the correct board and port are selected.
3. Write the code: Create a new sketch and write code to measure the distance using ultrasonic sensor.
4. Upload the code: Connect the Arduino to your computer and click the upload button in the IDE to compile and upload the code.
5. Observe the ultrasonic sensor, it will start measuring the distance and will sense accordingly. It will stop sensing once the distance is out of range.

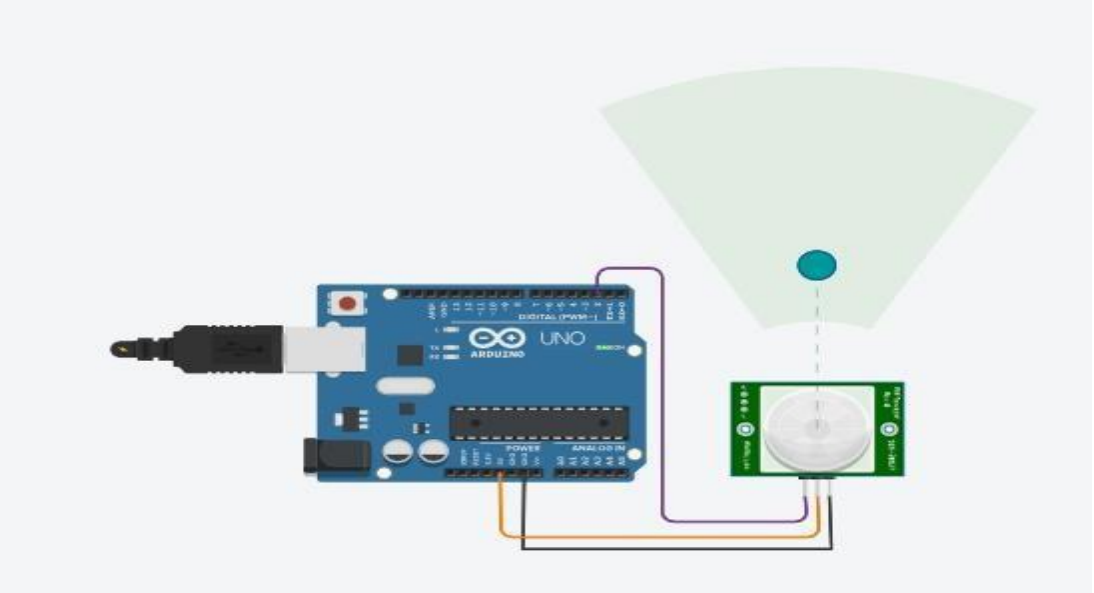
PRACTICAL 9

AIM : Write a program using for detecting objects using PIR sensor in Arduino.

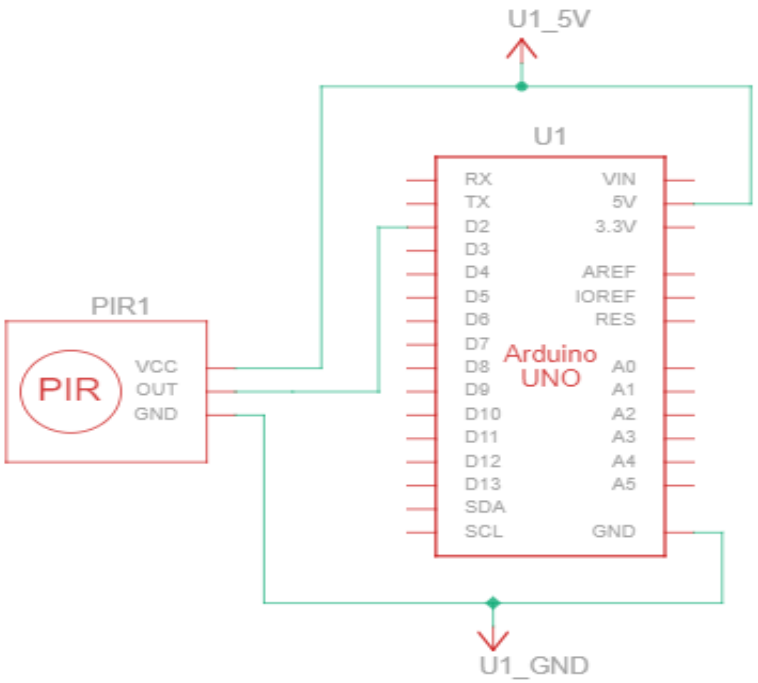
INTRODUCTION:

Utilizing the Arduino platform offers a straightforward approach to developing programs for controlling a myriad of electronic components, among which is the Ultrasonic Distance Sensor. An ultrasonic sensor operates as a tool to gauge the distance to an object by leveraging ultrasonic sound waves. In this experiment, our objective is to delve into the methodology through which an ultrasonic distance sensor measures and senses distances, thus enhancing our understanding of its functionality.

CIRCUIT DIAGRAM:



SCHEMATIC DIAGRAM:



CODE:

```
const int PIR_SENSOR_OUTPUT_PIN = 2;
int warm_up;
void setup() {
  pinMode(PIR_SENSOR_OUTPUT_PIN, INPUT);
// Serial.begin(9600);
  delay(20000);
}
void loop() {
  int sensor_output;
  sensor_output = digitalRead(PIR_SENSOR_OUTPUT_PIN);
  if( sensor_output == LOW )
  {
    if( warm_up == 1 )
     {
      Serial.print("Warming Up\n\n");
      warm_up = 0;
      delay(2000);
    }
    Serial.print("No object in sight\n\n");
    delay(1000);
  }
  else
  {
    Serial.print("Object detected\n\n");
    warm_up = 1;
    delay(1000);
  }
}
```

KEY COMPONENTS:

Name	Quantity	Component
U1	1	Arduino Uno R3
PIR1	1	PIR Sensor

STEPS OF WORKING:

1. Build the Circuit: Connect the Ultrasonic Distance Sensor to the Arduino board: GND to GND, VCC to 5V, and OUT to a digital pin (e.g., pin 7).
2. Set up the Arduino IDE: Install and open the Arduino IDE, select the correct board and port.
3. Write the Code: Create a new sketch, and write code to measure distance using the ultrasonic sensor.
4. Upload the Code: Connect the Arduino to your computer and upload the code using the Arduino IDE.
5. Observation: The ultrasonic sensor will start measuring distance and display the results on the Serial Monitor. It stops sensing when the distance is out of range.

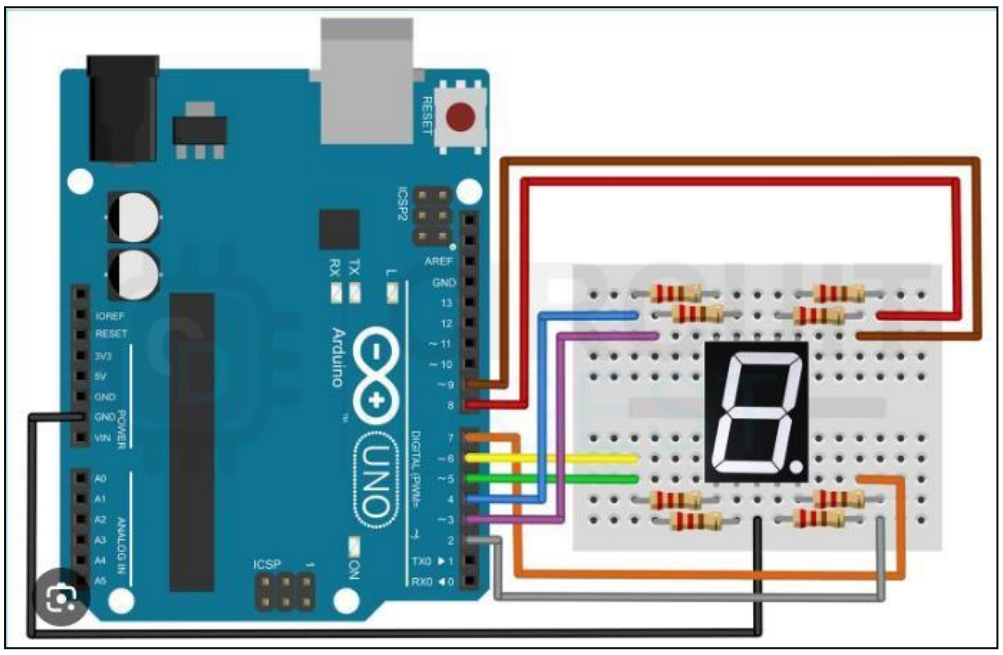
PRACTICAL 10

AIM : Write a program to interface 1 digit 7 segment display with Arduino.

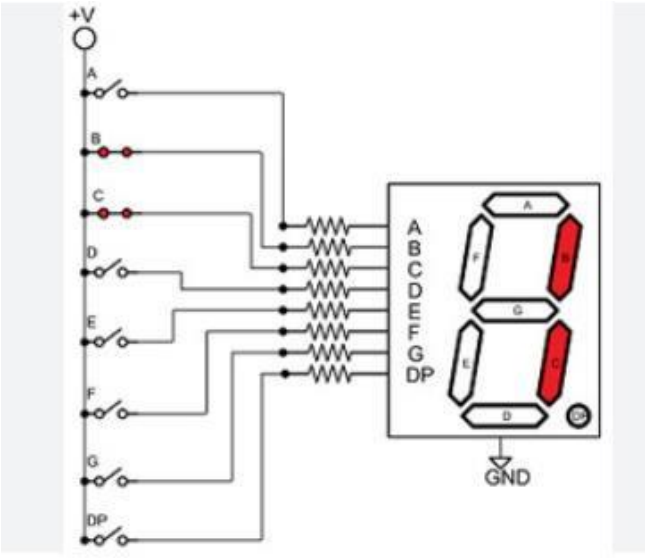
INTRODUCTION:

In this practical, we aim to interface a 1-digit 7-segment display with Arduino. 7-segment displays are commonly used to display numeric digits in various electronic projects. By interfacing the 7-segment display with Arduino, we can control and display numbers on the display using Arduino's digital output pins.

CIRCUIT DIAGRAM



SCHEMATIC DIAGRAM



CODE

```
// Define Arduino pins for 7-segment display segments
#define SEG_A 2
#define SEG_B 3
#define SEG_C 4
#define SEG_D 5
#define SEG_E 6
#define SEG_F 7
#define SEG_G 8
#define SEG_DP 9

void setup() {
  // Set Arduino pins as outputs
  pinMode(SEG_A, OUTPUT);
  pinMode(SEG_B, OUTPUT);
  pinMode(SEG_C, OUTPUT);
  pinMode(SEG_D, OUTPUT);
  pinMode(SEG_E, OUTPUT);
  pinMode(SEG_F, OUTPUT);
  pinMode(SEG_G, OUTPUT);
  pinMode(SEG_DP, OUTPUT);
}

void loop() {
  // Display number 0
  displayNumber(0);
  delay(1000); // Delay for 1 second

  // Display number 1
  displayNumber(1);
  delay(1000); // Delay for 1 second

  // Display number 2
  displayNumber(2);
  delay(1000); // Delay for 1 second

  // Display number 3
  displayNumber(3);
  delay(1000); // Delay for 1 second

  // Display number 4
  displayNumber(4);
  delay(1000); // Delay for 1 second

  // Display number 5
  displayNumber(5);
```

```
  delay(1000); // Delay for 1 second

  // Display number 6
  displayNumber(6);
  delay(1000); // Delay for 1 second

  // Display number 7
  displayNumber(7);
  delay(1000); // Delay for 1 second

  // Display number 8
  displayNumber(8);
  delay(1000); // Delay for 1 second

  // Display number 9
  displayNumber(9);
  delay(1000); // Delay for 1 second
}

// Function to display a digit (0-9) on the 7-segment display
void displayNumber(int num) {
  switch (num) {
    case 0:
      digitalWrite(SEG_A, HIGH);
      digitalWrite(SEG_B, HIGH);
      digitalWrite(SEG_C, HIGH);
      digitalWrite(SEG_D, HIGH);
      digitalWrite(SEG_E, HIGH);
      digitalWrite(SEG_F, HIGH);
      digitalWrite(SEG_G, LOW);
      digitalWrite(SEG_DP, LOW);
      break;
    case 1:
      digitalWrite(SEG_A, LOW);
      digitalWrite(SEG_B, HIGH);
      digitalWrite(SEG_C, HIGH);
      digitalWrite(SEG_D, LOW);
      digitalWrite(SEG_E, LOW);
      digitalWrite(SEG_F, LOW);
      digitalWrite(SEG_G, LOW);
      digitalWrite(SEG_DP, LOW);
      break;
    case 2:
      // Add code to display digit 2
      break;
    // Add cases for digits 3-9
    default:
      // Turn off all segments if num is out of range
```

```
    digitalWrite(SEG_A, LOW);
    digitalWrite(SEG_B, LOW);
    digitalWrite(SEG_C, LOW);
    digitalWrite(SEG_D, LOW);
    digitalWrite(SEG_E, LOW);
    digitalWrite(SEG_F, LOW);
    digitalWrite(SEG_G, LOW);
    digitalWrite(SEG_DP, LOW);
    break;
  }
}
```

STEPS OF WORKING

1. Open Arduino IDE: Launch the Arduino IDE software on your computer.
2. Create New Sketch: Click on "File" in the menu bar, then select "New" to create a new sketch.
3. Define Pins: Define the Arduino pins for each segment of the 7-segment display by using the #define directive.
4. Setup Function: Write the setup() function to set the defined pins as outputs using the pinMode() function.
5. Loop Function: Write the loop() function to control the display of numbers on the 7-segment display. Use digitalWrite() functions to control each segment based on the number to be displayed.
6. Upload Code: Connect your Arduino board to the computer via USB. Click on the upload button (right arrow icon) in the Arduino IDE to compile and upload the code to the Arduino board.
7. Test Display: Power on the Arduino board and observe the 7-segment display to verify that the desired numbers are correctly displayed based on the code.

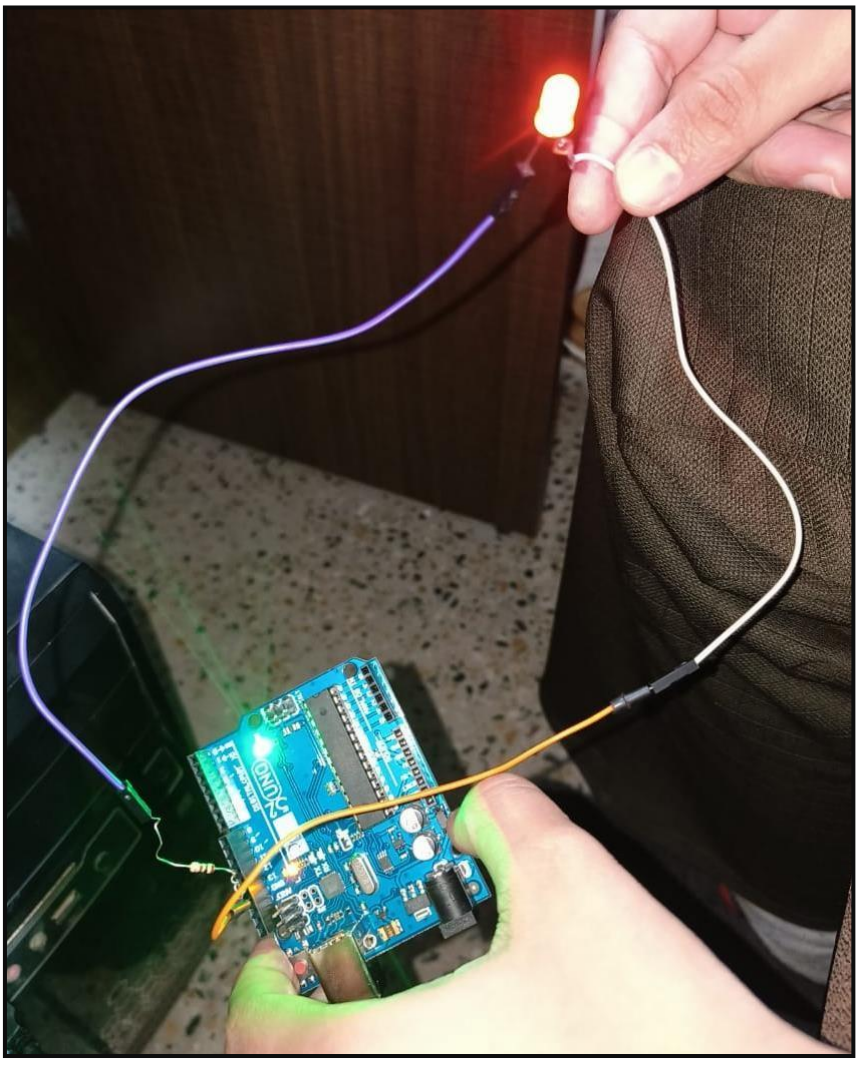
PRACTICAL 11

AIM : Hardware implementation of IOT Experiments

SWITCH ON LED BULB

INTRODUCTION:

The LED light experiment serves as a fundamental project in understanding hardware implementations using Arduino. LEDs (Light Emitting Diodes) are semiconductor light sources that emit light when an electric current passes through them. In this experiment, we aim to explore the hardware setup and programming necessary to control LEDs using an Arduino board. By grasping the basics of circuit connections and code execution, participants can gain insight into the principles of electronic control systems.



CODE

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

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

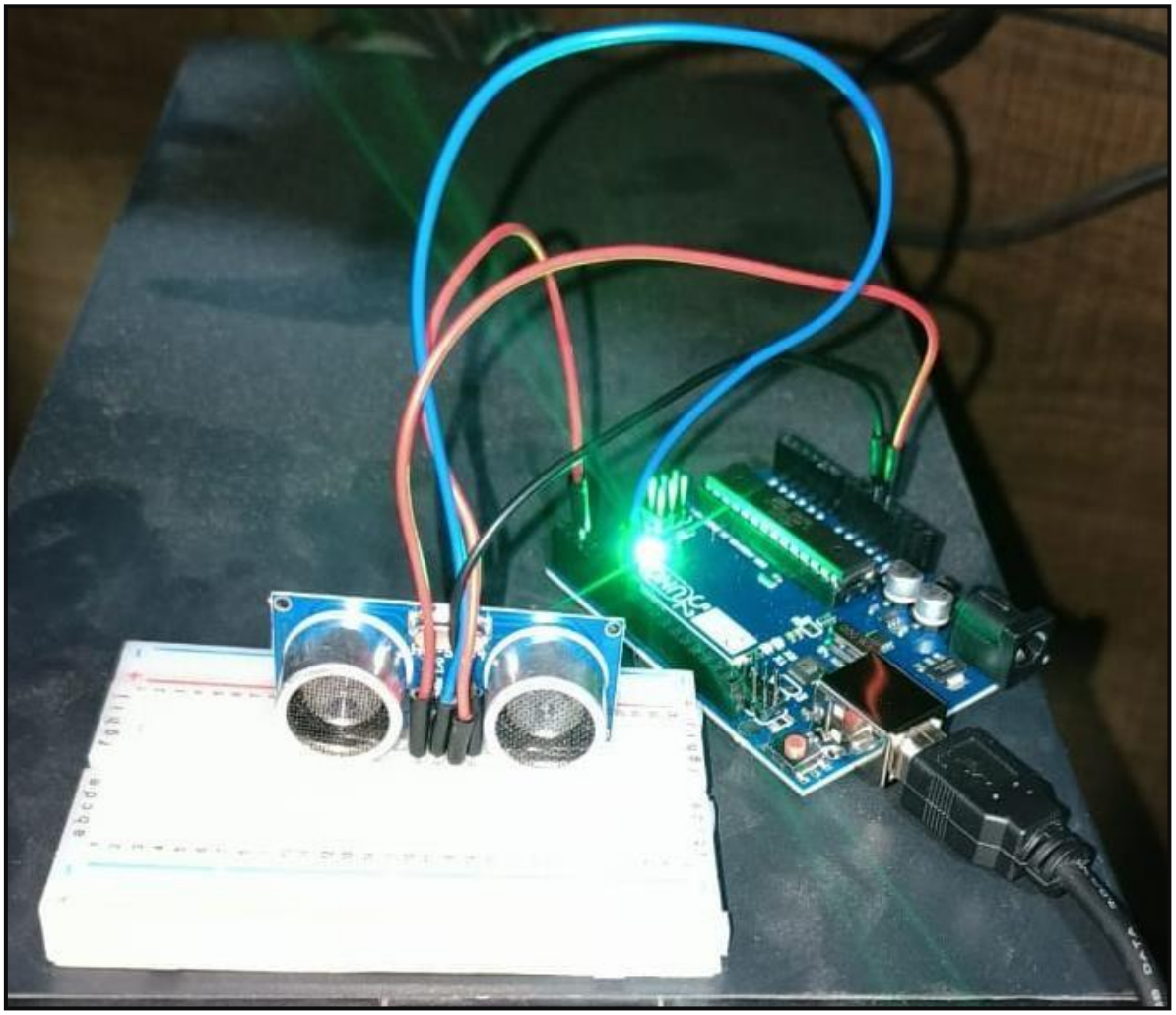
STEPS OF WORKING

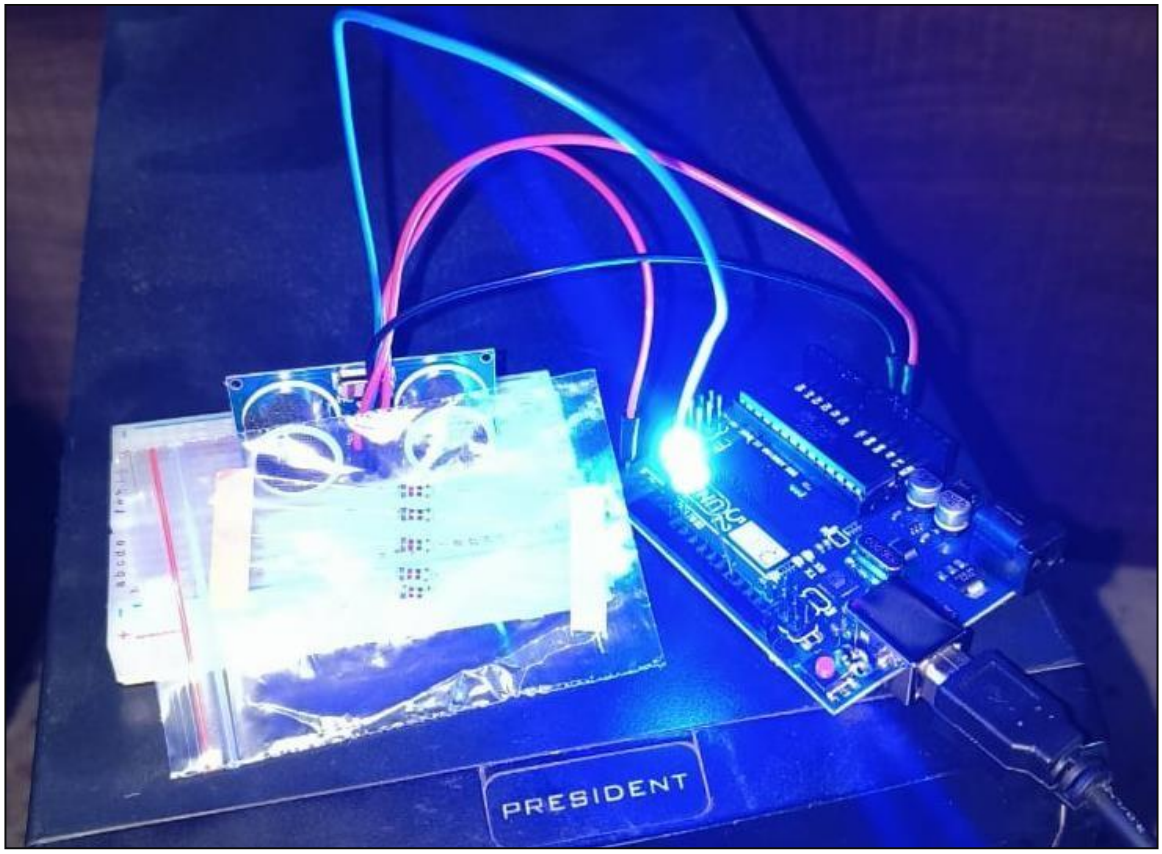
1. Gather Components: Assemble an Arduino board, LEDs, resistors, jumper wires, and a breadboard.
2. Build the Circuit: Connect LEDs to digital pins on the Arduino with resistors in series, ensuring each LED is connected to a separate pin.
3. Set Up Arduino IDE: Install the Arduino IDE, connect the Arduino board to the computer, and select the appropriate settings.
4. Write the Code: Develop a sketch to control LED behavior using functions like pinMode(), digitalWrite(), and delay().
5. Upload and Run: Upload the code to the Arduino board, observing LED responses as programmed.
6. Experiment and Modify: Explore different code variations for LED effects, and adjust circuit connections accordingly.
7. Conclusion: Reflect on outcomes and consider implications for future hardware projects with Arduino.

ULTRASONIC SENSOR

INTRODUCTION:

The Ultrasonic Sensor experiment delves into hardware implementation using Arduino, specifically focusing on measuring distance. Ultrasonic sensors utilize sound waves to gauge the distance to an object, making them ideal for applications such as proximity sensing and obstacle detection. In this experiment, we aim to understand the principles behind ultrasonic distance measurement and explore how to interface an ultrasonic sensor with an Arduino board.






```
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds
  return pulseIn(echoPin, HIGH);
}

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

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on 'COM4')

```
32in, 83cm
26in, 69cm
26in, 73cm
30in, 77cm
29in, 74cm
26in, 68cm
32in, 83cm
14in, 37cm
```

STEPS OF WORKING

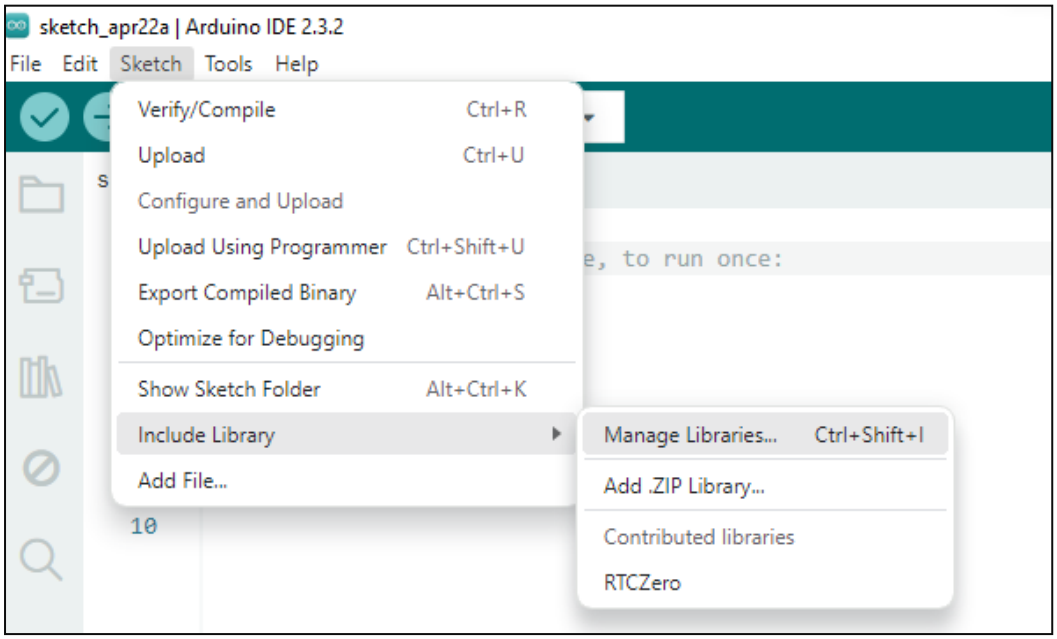
1. Gather Components: Assemble an Arduino board, an ultrasonic sensor, jumper wires, and a breadboard.
2. Build the Circuit: Connect the ultrasonic sensor to the Arduino, ensuring power and signal connections.
3. Set Up Arduino IDE: Install and configure the Arduino IDE for the board.
4. Write the Code: Develop a sketch to read distance data from the sensor using functions like digitalWrite(), pulseIn(), and Serial.println().
5. Upload and Run: Upload the code to the Arduino and observe distance readings on the Serial Monitor.
6. Experiment and Modify: Explore variations in code to customize sensor behavior or integrate additional functionality.
7. Conclusion: Reflect on experiment outcomes and consider applications of ultrasonic sensors in various projects.

PRACTICAL 12

AIM : To install libraries in Arduino IDE

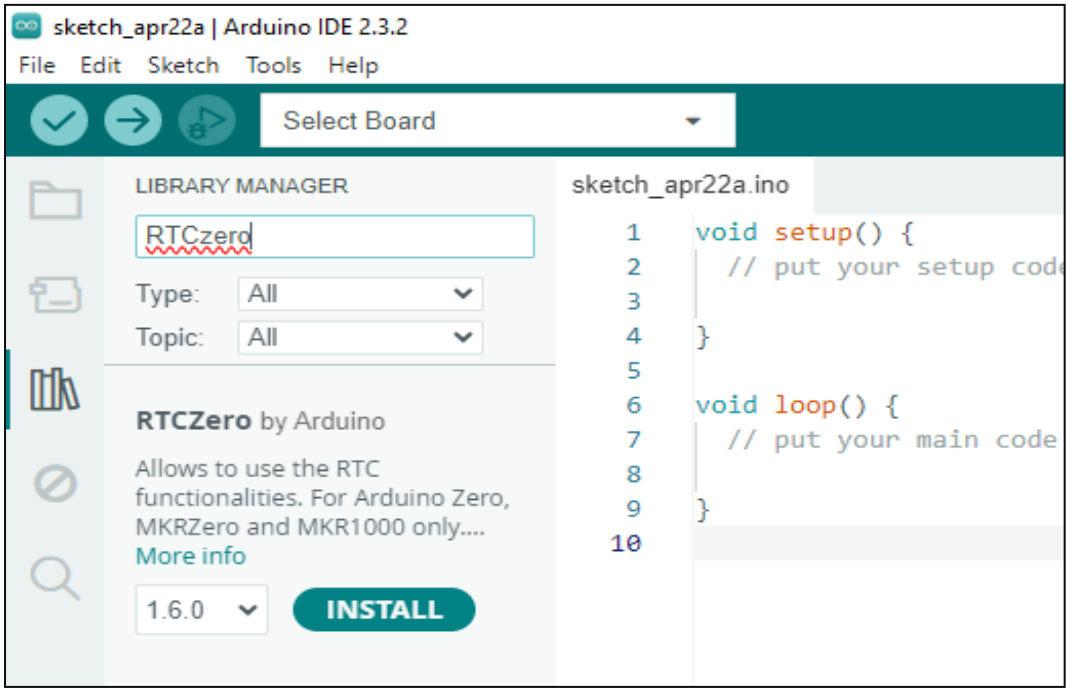
Step -1 Open Arduino IDE: Launch the Arduino IDE on your computer.

Step -2 Open Library Manager: Go to the "Sketch" menu at the top of the Arduino IDE window. From the dropdown menu, select "Include Library" and then "Manage Libraries...".

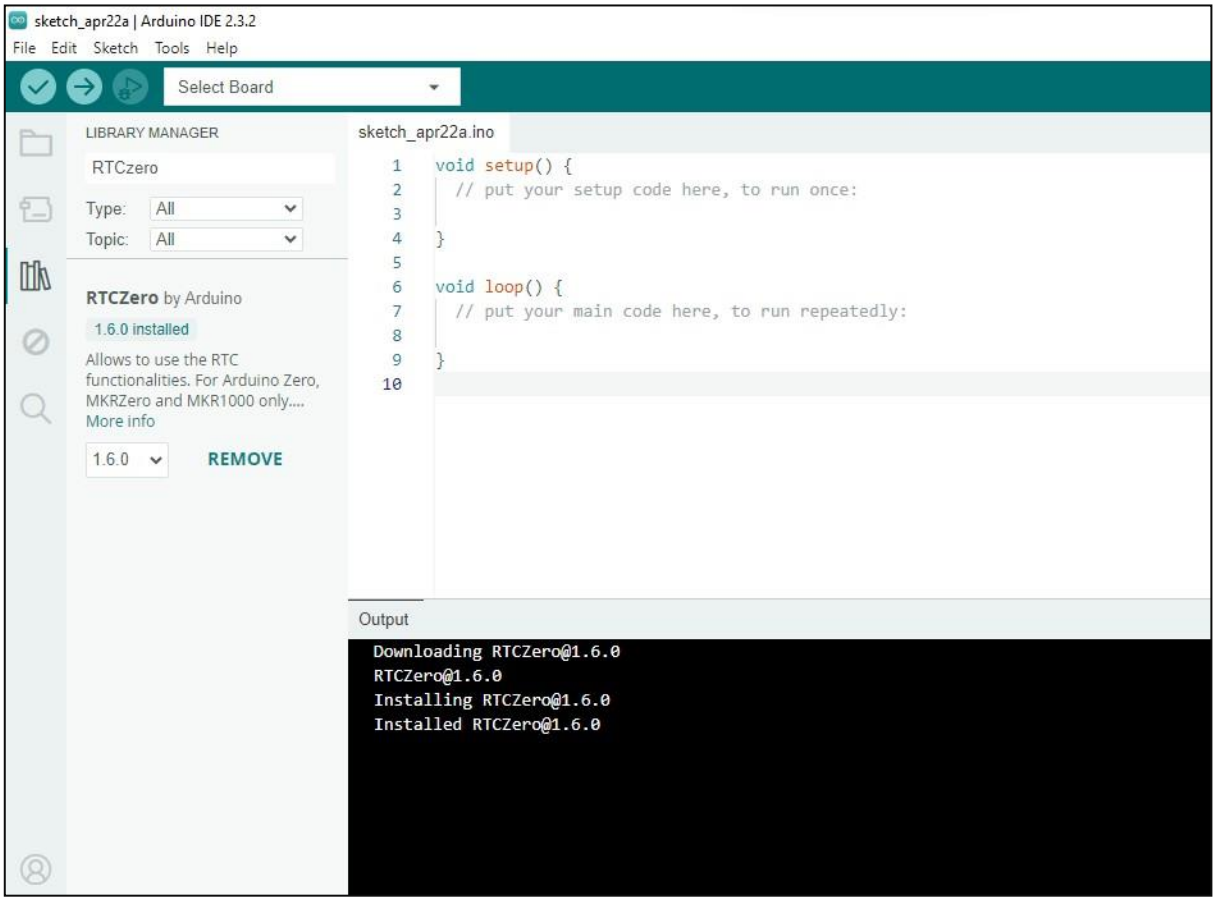


Step -3 Browse Libraries: The Library Manager window will open, displaying a list of available libraries. You can browse through the list or use the search bar to find a specific library.

Step -4 Select Library: Once you've found the library you want to install, click on it to select it.



Step – 5 Install Library: To install the library, click on the "Install" button next to the library name. The Arduino IDE will download and install the library files automatically**.**



After the installation is complete, you can close the Library Manager window. The newly installed library should now be available for use in your Arduino sketches.

PRACTICAL 13

AIM : Perform an Experiment in Raspberry Pi

INTRODUCTION:

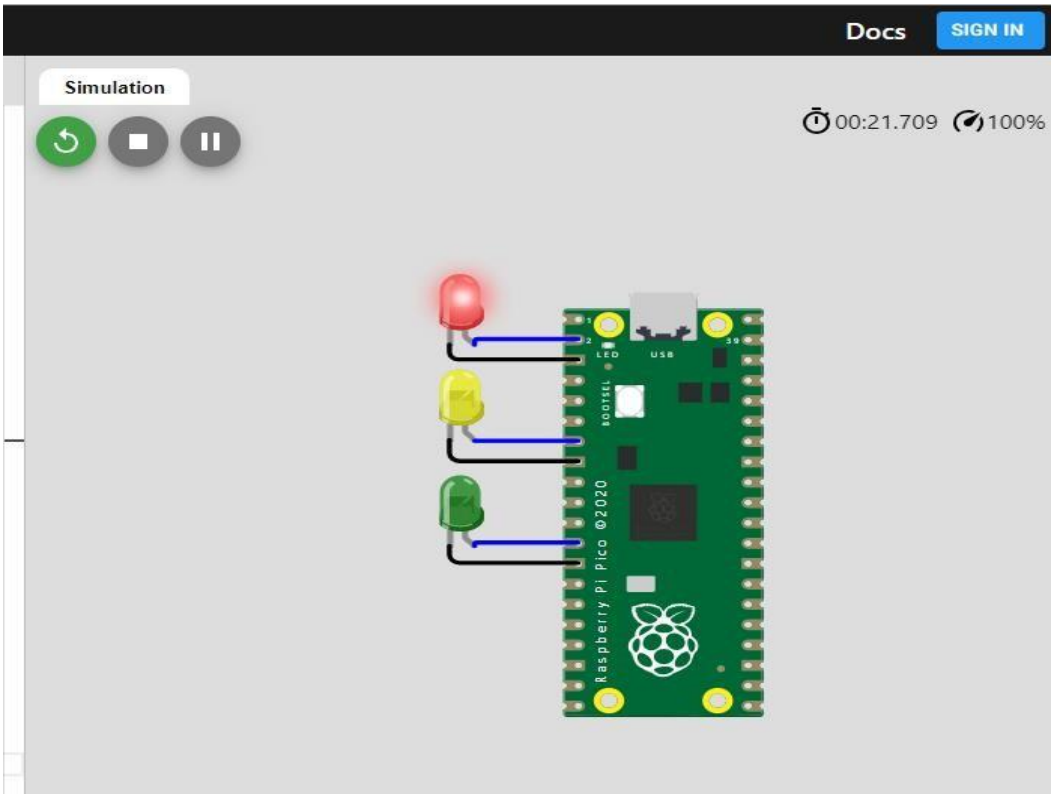
Raspberry Pi, a popular single-board computer, offers a diverse range of software options, making it a versatile platform for various projects and applications. From operating systems to programming environments and software libraries, Raspberry Pi supports a vast ecosystem of software tools tailored to different user needs.

TOPIC**:** Traffic Light Program in IDE

CODE :

```
#define RED 1
#define YELLOW 5
#define GREEN 9

void setup() {
  pinMode(RED, OUTPUT);
  pinMode(YELLOW, OUTPUT);
  pinMode(GREEN, OUTPUT);
}

void loop() {
  digitalWrite(GREEN, HIGH);
  delay(3000);

  digitalWrite(GREEN, LOW);
  digitalWrite(YELLOW, HIGH);
  delay(500);

  digitalWrite(YELLOW, LOW);
  digitalWrite(RED, HIGH);
  delay(2000);

  digitalWrite(YELLOW, HIGH);
  delay(500);
  digitalWrite(YELLOW, LOW);
  digitalWrite(RED, LOW);
}
```

CIRCUIT DIAGRAM



STEPS OF WORKING:

1. Access Raspberry Pi: Connect to your Raspberry Pi remotely using SSH or a remote desktop application.
2. Open IDE: Open your preferred IDE on the Raspberry Pi.
3. Copy Code: Copy the provided Traffic Light program code into a new file in the IDE.
4. Compile and Upload (If using Arduino IDE): Verify and upload the code to the Arduino board if using Arduino IDE.
5. Run Code: The Traffic Light program will start running, displaying the traffic light sequence.

PRACTICAL 14

AIM: Perform an Experiment in Raspberry Pi

INTRODUCTION:

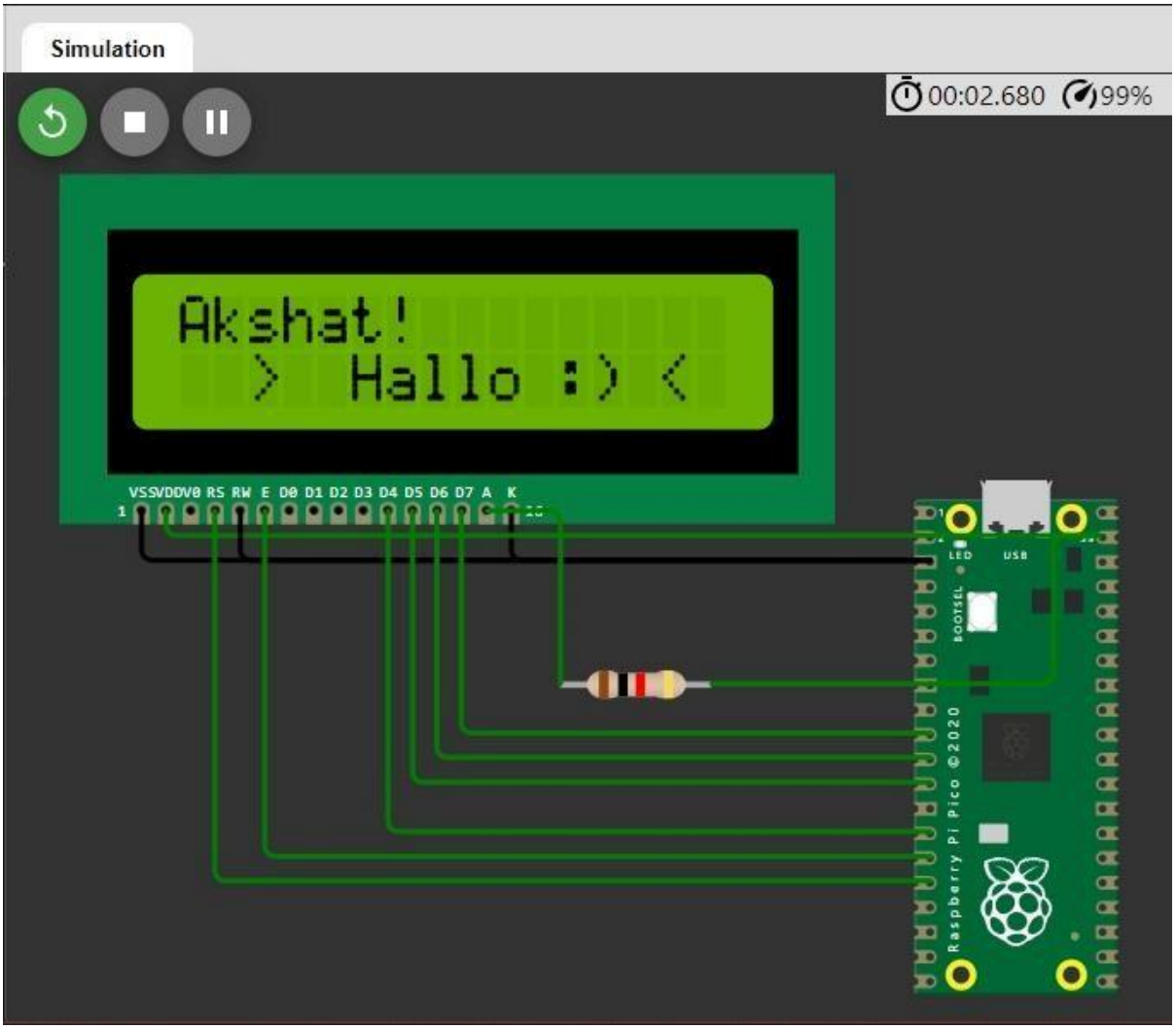
Raspberry Pi, a popular single-board computer, offers a diverse range of software options, making it a versatile platform for various projects and applications. From operating systems to programming environments and software libraries, Raspberry Pi supports a vast ecosystem of software tools tailored to different user needs.

TOPIC**:** Print your name on LCD

CODE

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("Akshat");

  lcd.setCursor(2, 1);
  lcd.print("> Hallo :) <");
}

void loop() {
  delay(1); // Adding a delay() here speeds up the simulation
}
```

CIRCUIT DIAGRAM



STEPS OF WORKING

1. Connect LCD: Wire the LCD to the GPIO pins on the Raspberry Pi.
2. Install Packages: Ensure Python and RPi.GPIO are installed on the Raspberry Pi.
3. Write Script: Create a Python script and copy the provided code into it.
4. Run Script: Execute the Python script on the Raspberry Pi.
5. Monitor Output: Verify that “Akshat” and "> Hallo :D <" are displayed on the LCD.

PRACTICAL 15

AIM: Application Based Experiment

TOPIC: Smart Irrigation Assistant

"Smart Irrigation Assistant" is a modern solution designed to optimize water usage in agriculture and gardening. By integrating soil moisture sensors with IOT technology, this system efficiency monitors the moisture levels in the soil. It automates the watering process, ensuring plants receive the right amount of water at the right time, minimizing water wastage and maximizing plant health. Through data analysis and remote access capabilities, users can conveniently monitor and adjust irrigation schedules from anywhere, leading to improved crop yields, reduced water consumption, and sustainable agricultural practices.

Project Overview

The objective of the Smart Irrigation Assistant project is to develop an efficient and automated system for optimizing water usage in agriculture and gardening through the integration of soil moisture sensors and IoT technology.

Components

1. Soil Moisture Sensors: These sensors detect the moisture levels in the soil.
2. Microcontroller (e.g., Arduino or Raspberry Pi): Controls the system and processes sensor data.
3. Watering Mechanism (e.g., water pump or solenoid valve): Delivers water to plants based on soil moisture readings.
4. IoT Module (e.g., Wi-Fi or GSM module): Enables remote monitoring and control of the system.
5. User Interface: Provides a graphical interface for users to monitor soil moisture levels and adjust watering settings.

Functionality

1. Soil Moisture Monitoring: The system continuously monitors soil moisture levels using the soil moisture sensors.
2. Automated Watering: When the soil moisture falls below a certain threshold, the system activates the watering mechanism to irrigate the plants.
3. Data Logging: The system logs soil moisture data over time, allowing users to track moisture trends and patterns.
4. Remote Access: Users can remotely access the system through a web interface or mobile app to monitor soil moisture levels and adjust watering schedules.
5. Alerts and Notifications: The system sends alerts and notifications to users when soil moisture levels are critically low or when watering events occur.

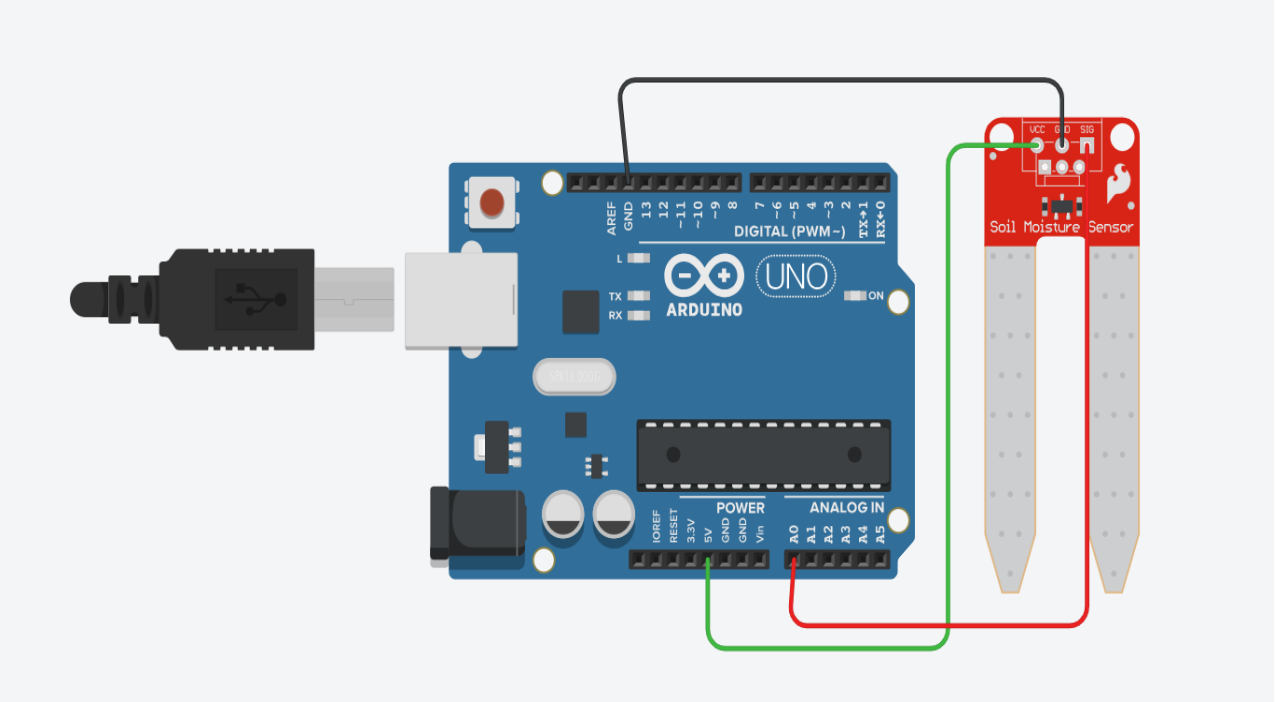
Benefits

1. Water Conservation: By watering plants only, when necessary, the system reduces water wastage and promotes efficient water usage.
2. Improved Plant Health: Consistent monitoring and optimized watering schedules lead to healthier plants with improved growth and yield.
3. Convenience: Users can remotely monitor and control the irrigation system, providing convenience and flexibility in managing their gardens or crops.
4. Sustainability: The Smart Irrigation Assistant promotes sustainable agricultural practices by minimizing water usage and environmental impact.

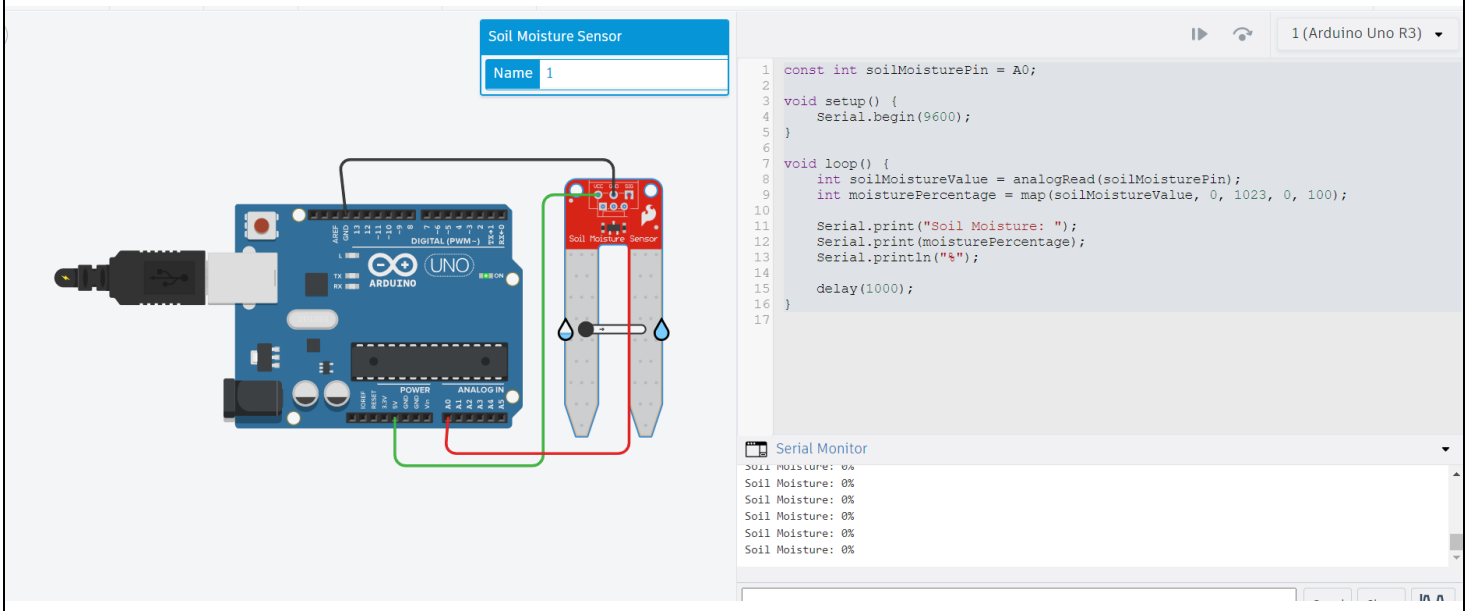
CODE

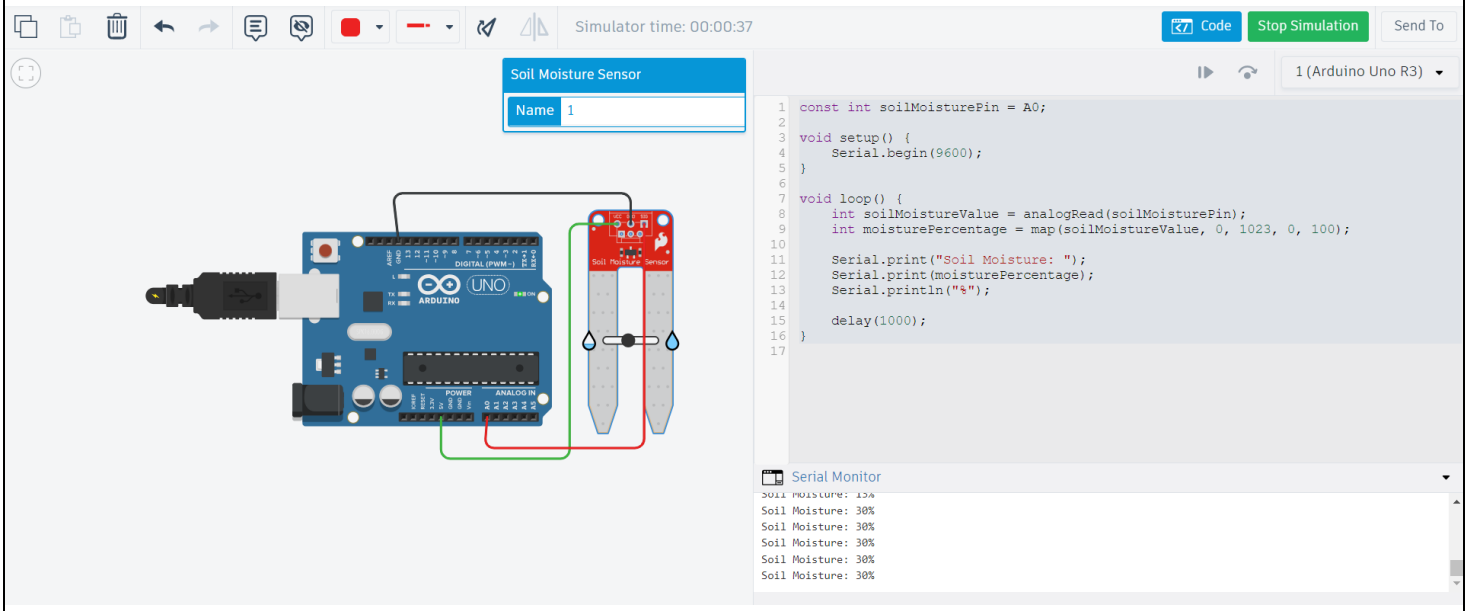
```
const int soilMoisturePin = A0;

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

void loop () {
    int soilMoistureValue = analogRead(soilMoisturePin);
    int moisturePercentage = map(soilMoistureValue, 0, 1023, 0, 100);

    Serial.print("Soil Moisture: ");
    Serial.print(moisturePercentage);
    Serial.println("%");

    delay (1000);
}
```

CIRCUIT DIAGRAM



WORKING OF CIRCUIT





STEPS OF WORKING

1. Connect Soil Moisture Sensor: Wire the soil moisture sensor to the analog pin on the microcontroller.
2. Install Libraries: Ensure necessary libraries for soil moisture sensor and IoT communication are installed.
3. Write Firmware: Develop firmware to read sensor data and control watering mechanism.
4. Upload Firmware: Upload the firmware to the microcontroller.
5. Power On: Power on the system and initialize sensor readings.
6. Monitor Moisture Levels: Continuously monitor soil moisture levels using sensor data.
7. Activate Watering: If moisture falls below threshold, activate watering mechanism.
8. Provide Feedback: Display soil moisture readings or send notifications to user interface.
9. Log Data: Log soil moisture data for analysis and historical tracking.
10. Test and Calibrate: Verify system functionality and calibrate thresholds if needed.
11. Deploy: Deploy the system in the desired agricultural or gardening environment.