



# **ERODE SENGUNTHAR ENGINEERING COLLEGE**

**(An Autonomous Institution)**

Approved by AICTE, New Delhi, Permanently Affiliated to Anna University- Chennai,  
Accredited by National Board of Accreditation (NBA), New Delhi &  
National Assessment and Accreditation Council (NAAC), Bangalore with 'A' Grade  
**PERUNDURAI -638 057, TAMILNADU, INDIA**



## **DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

### **19AD606 - INTERNET OF THINGS LABORATORY**

### **LAB MANUAL**



## INDEX

[illegible]



## **LIST OF EXPERIMENTS**

1. STUDY OF RASPBERRY PI
2. BLINKING LED USING RASPBERRY PI
3. REAL TIME MOTION DETECTION USING SENSOR
4. MEASURING TEMPERATURE AND HUMIDITY USING SENSOR
5. REAL TIME SMOKE DETECTION WITH ALERT SYSTEM
6. IOT BASED INTELLIGENT TRAFFIC MANAGEMENT SYSTEM
7. IOT BASED SMART IRRIGATION SYSTEM
8. IOT BASED SMART WASTE MANAGEMENT SYSTEM FOR SMART CITY
9. IOT BASED SMART WATER MANAGEMENT SYSTEM
10. ANY OPEN ENDED PROJECT



**Ex.No : 1**

## **STUDY OF RASPBERRY PI**

**Date :**

### **Aim:**

To understand the basic functionalities and setup process of Raspberry Pi, focusing on the Raspberry Pi 4 model.

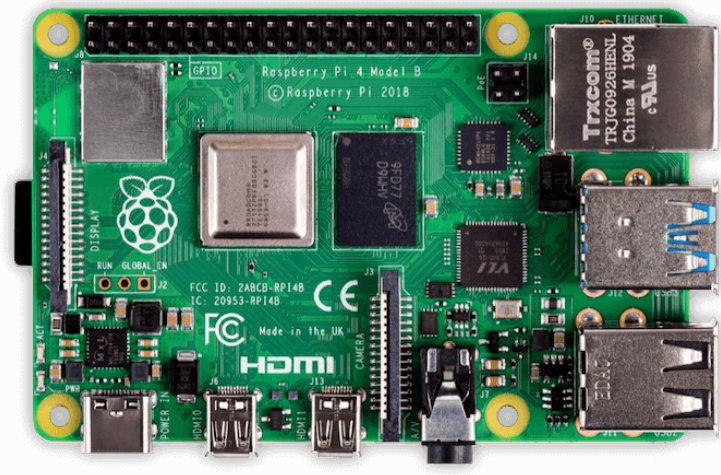
### **Apparatus Required:**

Raspberry Pi, power supply, microSD card, monitor, keyboard, mouse, breadboard, LEDs, resistors, jumper wires, sensors (optional), and networking tools

### **Introduction**

Raspberry Pi is a small, affordable, and powerful single-board computer designed to promote computer science education and innovation. Originally developed by the Raspberry Pi Foundation, it has evolved into a popular platform for projects ranging from DIY electronics to advanced IoT applications. Raspberry Pi supports various operating systems, making it versatile for educational and professional use.

The Raspberry Pi 4, the latest in the series, offers significant improvements in performance, making it suitable for tasks such as programming, media streaming, and AI/ML applications.



### **Flavors**

Raspberry Pi supports multiple operating systems tailored for different purposes. Some popular options include:

**Raspberry Pi OS (formerly Raspbian):** A Debian-based Linux OS optimized for the Raspberry Pi hardware.

**Ubuntu:** A full-featured Linux distribution widely used in development and server applications.

**Kali Linux:** A security-focused OS designed for penetration testing and ethical hacking.

**RetroPie:** A platform for retro gaming, allowing users to emulate classic game consoles.

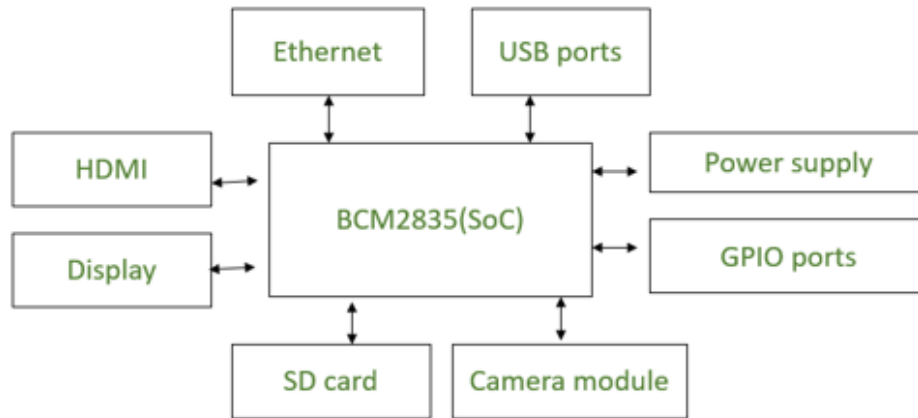
**Windows IoT Core:** A simplified version of Windows for IoT applications.

Each flavor is customizable and suited to specific applications, making Raspberry Pi a

flexible tool for learners and professionals.

### Architecture

Raspberry Pi's architecture is based on the ARM (Advanced RISC Machine) processor family, known for energy efficiency and compact design. Key features of the Raspberry Pi 4 include:



- Processor: Quad-core Cortex-A72 64-bit ARMv8 processor, running at 1.5 GHz.
- RAM: Available in 2GB, 4GB, or 8GB LPDDR4 options.
- Storage: MicroSD card slot for OS and data storage.
- Ports:
  - 2 × USB 3.0 and 2 × USB 2.0 ports.
  - Dual micro-HDMI ports supporting 4K resolution.
  - Ethernet port for high-speed network connectivity.
- Wireless Connectivity: Built-in Wi-Fi (802.11ac) and Bluetooth 5.0.
- GPIO Pins: 40 General-Purpose Input/Output pins for hardware interfacing.

This architecture provides a balance between affordability and performance, suitable for educational and real-world applications.

### Experiment: Using Raspberry Pi 4

The experiment involves setting up the Raspberry Pi 4, installing Raspberry Pi OS, and running basic Python scripts.

#### Materials Required:

- Raspberry Pi 4 (any RAM variant)
- Power supply (USB-C adapter)
- MicroSD card (minimum 16GB)
- HDMI monitor or TV
- Keyboard and mouse
- Internet connection

#### Procedure:

1. Download and flash Raspberry Pi OS onto the microSD card using tools like Balena Etcher.
2. Insert the microSD card into the Raspberry Pi 4 and connect peripherals (monitor,



keyboard, mouse).

3. Power on the Raspberry Pi and complete the initial OS setup.

4. Open the terminal and update the system:

`sudo apt update && sudo apt upgrade -y`

5. Run a basic Python script to test functionality:

`print("Hello, Raspberry Pi!")`

6. Explore GPIO pins by connecting an LED and writing a Python program to control it.

## **Result**

The Raspberry Pi 4 was successfully set up, and a basic Python script was executed. This demonstrated its functionality and provided foundational knowledge for further exploration.

## **Output:**



**Ex.No : 2**

## **BLINKING LED USING RASPBERRY PI**

**Date :**

**Aim:**

To create a smooth fading effect for an LED by continuously adjusting its brightness. The brightness is increased and decreased in a loop to create the fading effect.

**Apparatus Required:**

Raspberry Pi, power supply, microSD card, LED, 330  $\Omega$  resistor, breadboard, jumper wires, HDMI monitor, keyboard, mouse, and HDMI cable.

**Procedure:**

Step 1: Initialize:

- Set initial brightness to 0.
- Set fade amount to 5.

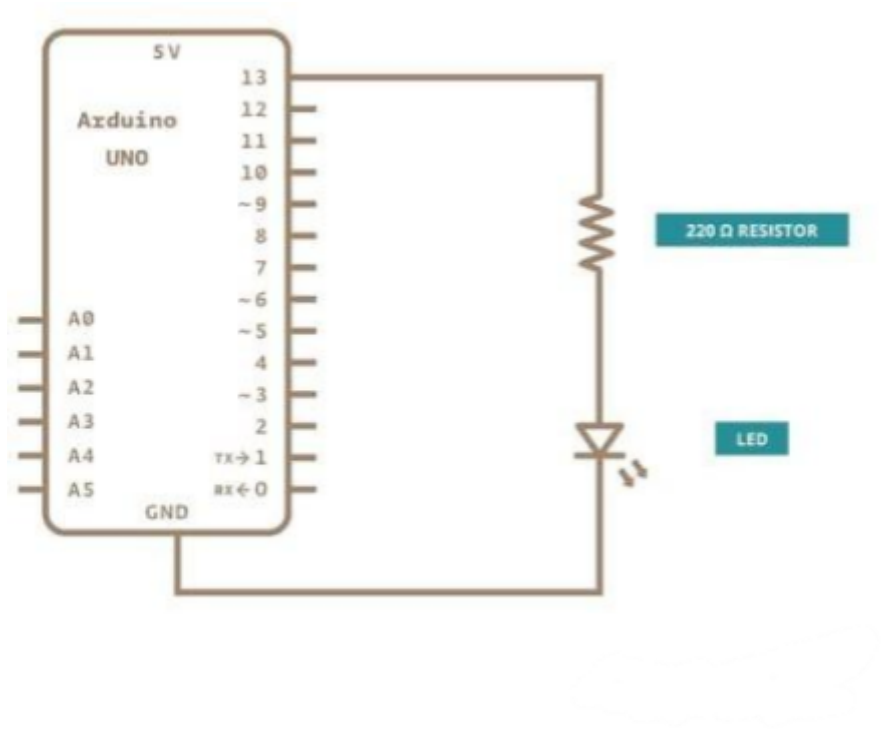
Step 2: Set Up LED Pin:

- Configure the LED pin as an output.

Step 3: Main Loop:

- Write current brightness to the LED.
- Adjust brightness by adding fade amount.
- Reverse fade direction if brightness reaches 0 or 255.
- Delay briefly to show the fade effect.

### Circuit Diagram :



### Output:

INPUT (V)	OUTPUT (LED)
5	LED ON
0	LED OFF

**Program:**

```
import RPi.GPIO as GPIO
import time
```

```
LED_PIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
```

```
try:
```

```
    while True:
        GPIO.output(LED_PIN, GPIO.HIGH)
        time.sleep(1)
        GPIO.output(LED_PIN, GPIO.LOW)
        time.sleep(1)
```

**Result:**

The LED was successfully controlled to blink at regular intervals using Raspberry Pi and Python. This experiment demonstrated the use of GPIO pins for basic hardware interfacing. Thus the program was executed and output was verified successfully.



**Ex.No 3**

## **REAL TIME MOTION DETECTION USING SENSOR**

**Date :**

### **Aim:**

To develop a real-time motion detection system using a motion sensor (PIR sensor) and Raspberry Pi, demonstrating basic hardware interfacing and sensor-based event detection.

### **Apparatus Required:**

Arduino Uno, Arduino IDE Tool, Ultrasonic Sensor (HC-SR04), Breadboard, Jumper wires.

### **Procedure:**

Step 1: Initialize the sensor and pins (Trigger as output, Echo as input).

Step 2: Start loop:

- Send a short pulse on the Trigger pin.
- Measure the time for the Echo pin to receive the reflected signal.
- Calculate distance using the formula:  
$$\text{Distance (cm)} = \frac{\text{Time } (\mu\text{s})}{2} \times 0.0343$$
  
$$\text{Distance (cm)} = \text{Time } (\mu\text{s}) \times 0.0343$$
- Display the distance on the serial monitor or

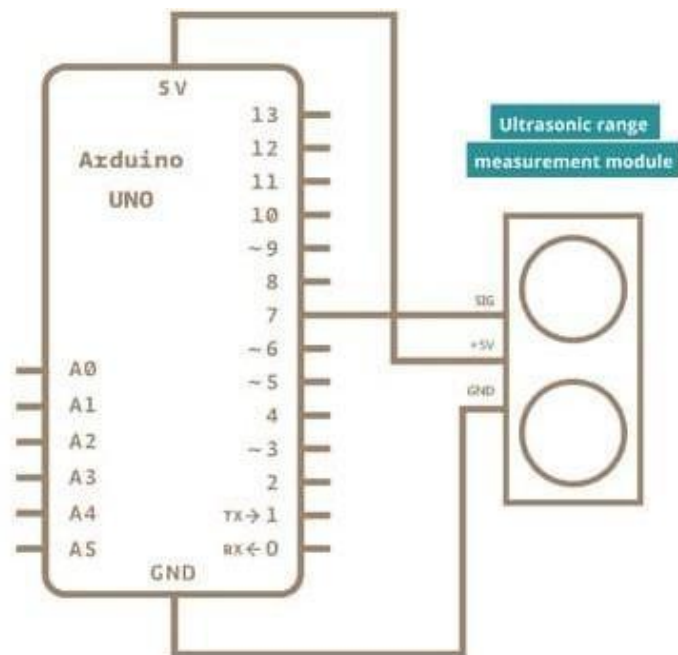
LCD. Step 3: Repeat the loop continuously.

### **Program:**

```
const int trigPin = 9;
const int echoPin = 10;
int distance;
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(500); }
```



## Circuit Diagram:



## Output:

```
ultrasonic_base [Arduino1.8.3]
File Edit Sketch Tools Help
ultrasonic_base
pinMode(trigPin, OUTPUT);
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(5);
digitalWrite(trigPin, LOW);
pinMode(echoPin, INPUT);
duration = pulseIn(echoPin, HIGH);
inches = microsecondsToInches(duration);
cm = microsecondsToCentimeters(duration);
// Serial.println(cm);
// Serial.println(inches);
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;
}

Data speedup
```

COM1

1cm  
2cm  
3cm  
4cm  
5cm  
10cm  
13cm  
18cm  
20cm  
24cm  
26cm  
35cm  
47cm

Autobaud No line ending 9600 baud Clear output

**S**

**Result:**

The Arduino successfully measured and displayed the distance of objects from the ultrasonic sensor continuously, with accurate distance readings shown on the serial monitor or LCD screen.



**Ex.No 4**

## **MEASURING TEMPERATURE AND HUMIDITY USING SENSOR**

**Date :**

### **Aim:**

To design and implement a temperature monitoring system using an Arduino board and a temperature sensor to continuously measure and display temperature in real-time.

### **Apparatus Required:**

Arduino UNO Board, Temperature Sensor (e.g., LM35 or DHT11), Jumper Wires, Breadboard, Resistor, USB Cable, Arduino IDE Tool.

### **Procedure:**

Step 1: Initialize Arduino and temperature sensor.

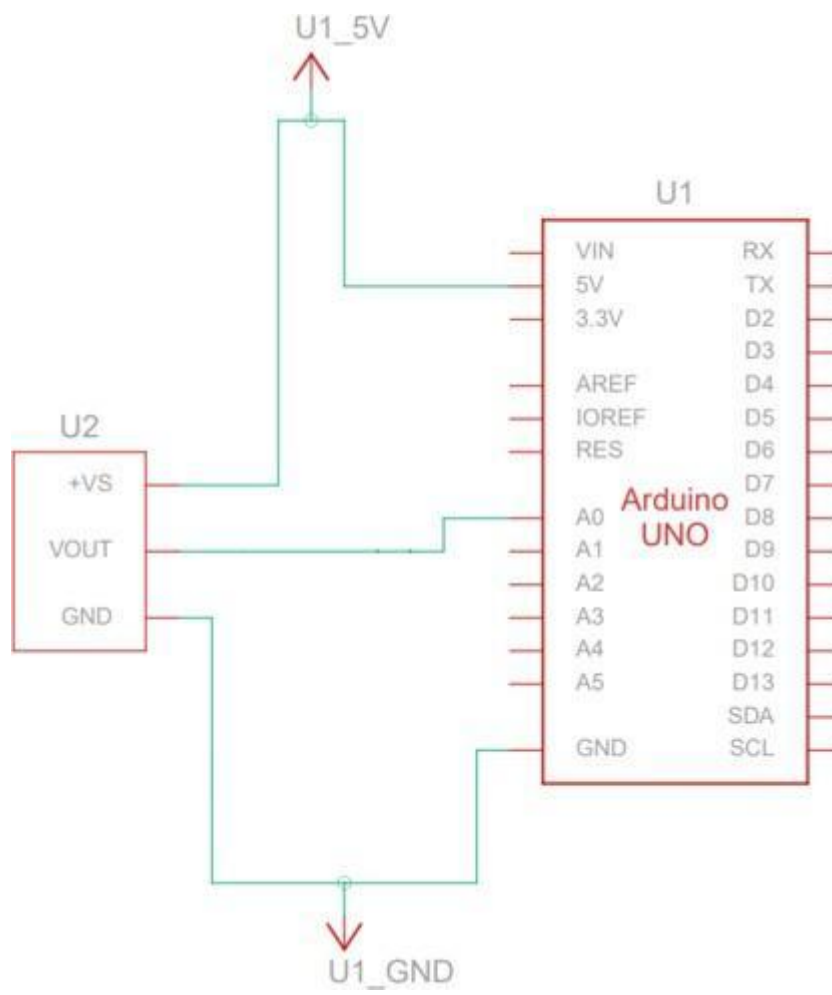
Step 2: Read the temperature data using `analogRead()` for LM35 or `digitalRead()` for DH11

Step 3: Convert the sensor data to temperature (Celsius or Fahrenheit).

Step 4: Display the temperature value on Serial Monitor or LCD.

Step 5: Repeat the process with a delay.

**Circuit diagram:**



**Program:**

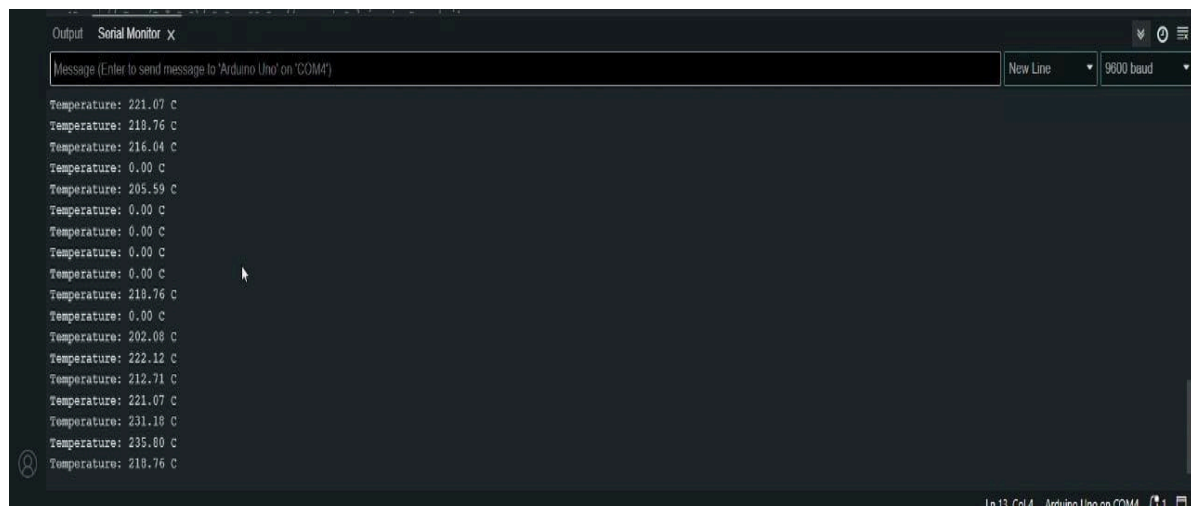
```
#include <dht.h> dht DHT;

#define DHT11_PIN 7 // define DHT pin

void setup(){
  Serial.begin(9600); // Start serial communication
}

void loop(){
  int chk = DHT.read11(DHT11_PIN); // check the data coming from the DHT pin
  Serial.print("Temperature = "); // print temperature on the serial monitor
  Serial.println(DHT.temperature);
  Serial.print("Humidity = "); // Print humidity on the serial monitor
  Serial.println(DHT.humidity);
  delay(1000); // delay of 1 second
}
```

## Output



The screenshot shows the Arduino IDE Serial Monitor window. The title bar reads "Output Serial Monitor x". Below the title bar is a text input field with the placeholder "Message (Enter to send message to 'Arduino Uno' on 'COM4')". To the right of the input field are two dropdown menus: "New Line" and "9600 baud". The main area of the window displays a list of temperature readings in Celsius, each preceded by the label "Temperature:". The readings are: 221.07 C, 218.76 C, 216.04 C, 0.00 C, 205.59 C, 0.00 C, 0.00 C, 0.00 C, 0.00 C, 218.76 C, 0.00 C, 202.08 C, 222.12 C, 212.71 C, 221.07 C, 231.18 C, 235.80 C, and 218.76 C. A mouse cursor is visible over the text "0.00 C". At the bottom right of the window, the status bar shows "In 13, Col 4 - Arduino Uno on COM4" along with some icons.

```
Temperature: 221.07 C
Temperature: 218.76 C
Temperature: 216.04 C
Temperature: 0.00 C
Temperature: 205.59 C
Temperature: 0.00 C
Temperature: 0.00 C
Temperature: 0.00 C
Temperature: 0.00 C
Temperature: 218.76 C
Temperature: 0.00 C
Temperature: 202.08 C
Temperature: 222.12 C
Temperature: 212.71 C
Temperature: 221.07 C
Temperature: 231.18 C
Temperature: 235.80 C
Temperature: 218.76 C
```

**Result:**

The temperature was successfully monitored using the Arduino and displayed in real-time on the Serial Monitor (or LCD). The system accurately measured temperature variations and provided real-time data





**Ex.No 5**

## **REAL TIME SMOKE DETECTION WITH ALERT SYSTEM**

**Date :**

### **Aim:**

To design and implement a system using an MQ sensor and Arduino to detect the presence and concentration of gases in the environment and display the sensor data in real-time.

### **Apparatus Required:**

Arduino UNO Board, MQ Sensor, Jumper Wires, Breadboard, Resistor, USB Cable, Arduino IDE Tool, LCD Display, Power Supply, LED, Buzzer.

### **Procedure:**

Step 1: Initialize Arduino and MQ Sensor:

- Connect the MQ sensor to the Arduino.
- Setup the Arduino with the required power supply.

Step 2: Sensor Calibration :

- Initialize sensor parameters.
- Run the MQ sensor in clean air for a period to establish baseline readings.

Step 3: Read Analog Data from the MQ Sensor:

- Use `analogRead()` function to read the gas concentration from the MQ sensor.
- Store the sensor reading in a variable.

Step 4: Process the Data:

- Convert the raw analog data to a human-readable form (e.g., gas concentration in ppm).
- (Optional) Compare sensor values against predefined threshold levels to detect dangerous gas concentrations.

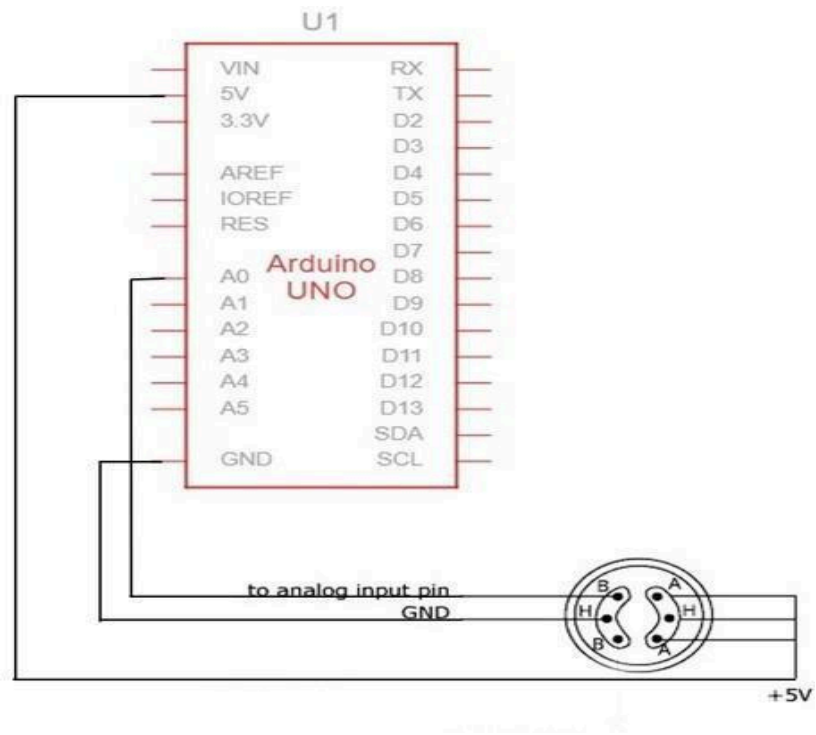
Step 5: Output the Sensor Data:

- Print the sensor readings to the Serial Monitor of the Arduino IDE.
- (Optional) Display the gas concentration on an LCD display or trigger an alert via LED or buzzer if the concentration exceeds a threshold.

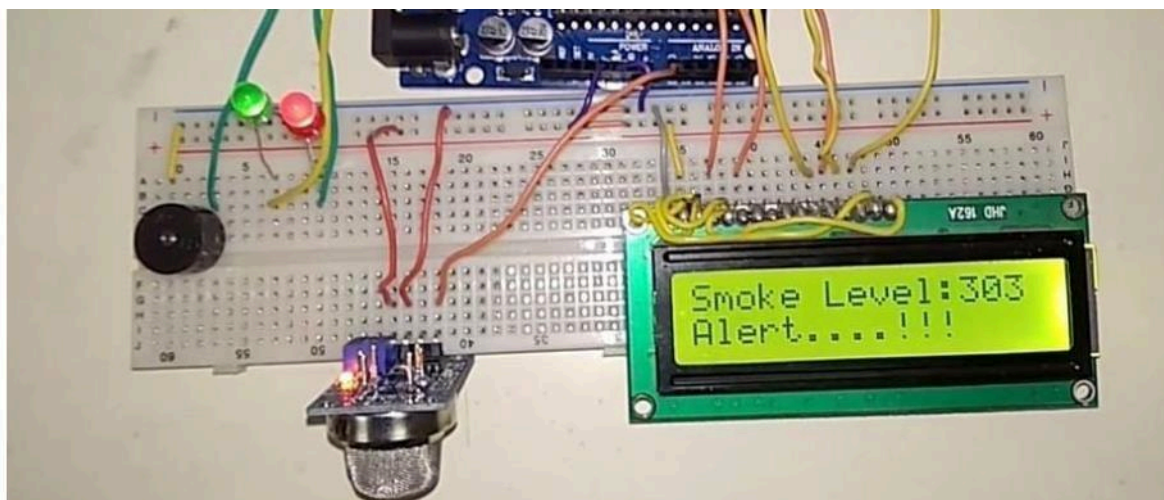
Step 6: Repeat the Reading Process:

- Continuously monitor gas concentration by running the data reading and processing in an infinite loop.
- Add a small delay between readings to avoid flooding the data.

## Circuit Diagram:



## Output:



**Program :**

```
sensorPin = A0;
int sensorValue = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  Serial.print("Gas Sensor Value: ");
  Serial.println(sensorValue);
  delay(1000);
}
```

**Result :**

The gas concentration was successfully detected using the MQ sensor and displayed in real-time on the Arduino Serial Monitor (or LCD). The system accurately measured gas levels, and alerts (LED/buzzer) were triggered when the concentration exceeded the defined

threshold.



**Ex.No 6**

## **IOT BASED INTELLIGENT TRAFFIC MANAGEMENT SYSTEM**

**Date :**

### **Aim:**

To design and implement an IoT-based intelligent traffic management system using Arduino, enabling efficient traffic flow by monitoring and controlling traffic signals based on real-time data.

### **Apparatus Required:**

Arduino Uno or Mega, IR sensors or Ultrasonic sensors, LEDs (Red, Yellow, Green), Breadboard, Jumper wires, Resistors (330 ohm), Wi-Fi module (ESP8266/ESP32), Power supply or USB cable

### **Procedure:**

Step 1:

- Connect the IR/ultrasonic sensors to Arduino for detecting vehicle density.

Step 2:

- In Connect LEDs to Arduino as traffic signals using resistors.

Step 3:

- Attach the Wi-Fi module to enable IoT connectivity.

Step 4:

- Write and upload code using Arduino IDE for traffic control based on sensor data.

Step 5:

- Link the setup to an IoT platform for real-time data monitoring.

Step 6:

- Test and verify lane prioritization and data transmission.

**Result:**

The system successfully detected vehicle density using sensors and managed traffic signals accordingly. Real-time traffic data was transmitted to the IoT platform, and lane prioritization was achieved efficiently.



**Ex.No 7**

## **IOT BASED SMART IRRIGATION SYSTEM**

**Date :**

### **Aim:**

To design and implement an IoT-based smart irrigation system using Arduino to monitor soil moisture and automate water supply for efficient water management.

### **Apparatus Required:**

Arduino Uno or Mega, Soil moisture sensor, Water pump with relay module, Water pipe, Jumper wires, Breadboard, Wi-Fi module (ESP8266/ESP32), Power supply or USB cable, 9V battery (for pump).

### **Procedure:**

Step 1:

- Connect the soil moisture sensor to Arduino to measure soil moisture levels.

Step 2:

- Interface the water pump with the Arduino using a relay module for automated water supply.

Step 3:

- Attach the Wi-Fi module for IoT connectivity to monitor data remotely.

Step 4:

- Write and upload code using Arduino IDE to automate watering based on sensor data.

Step 5:

- Link the setup to an IoT platform for real-time monitoring and control.

Step 6:

- Test the system by varying soil moisture levels and observing pump activation.

**Result:**

The soil moisture levels were successfully monitored using the Arduino, and the water pump was activated automatically based on sensor readings. Real-time data was displayed on the IoT platform, ensuring efficient water management.

**Ex.No 8**

**Date :**

## **IOT BASED SMART WASTE MANAGEMENT SYSTEM FOR SMART CITY**

**Aim:**

To develop an IoT-based smart waste management system using Arduino to monitor waste levels in bins and optimize waste collection in smart cities.

**Apparatus Required:**

Arduino Uno or Mega, Ultrasonic sensor, Wi-Fi module (ESP8266/ESP32), Buzzer or LED indicator, Jumper wires, Breadboard, Power supply or USB cable, 9V battery (optional for independent modules).

**Procedure:**

Step 1:

- Connect ultrasonic sensors to Arduino to measure the waste levels in bins.

Step 2:

- Attach a Wi-Fi module for transmitting data to an IoT platform.

Step 3:

- Add a buzzer or LED indicator to signal when the bin is full.

Step 4:

- Write and upload code using Arduino IDE for monitoring and transmitting waste levels.

Step 5:

- Link the setup to an IoT platform to display bin status in real-time.

Step 6:

- Test the system by filling bins to different levels and observing data transmission and alerts.

**Result:**

The waste levels in bins were accurately detected using ultrasonic sensors and displayed in real-time on the IoT platform. Alerts were generated when the bins were full, facilitating timely waste collection.

**Ex.No 9**

## **IOT BASED SMART WATER MANAGEMENT SYSTEM**

**Date :**

### **Aim:**

To design and implement an IoT-based smart water management system using Arduino to monitor water levels, detect leakages, and optimize water usage.

### **Apparatus Required:**

Arduino Uno or Mega, Ultrasonic sensor, Water flow sensor, Wi-Fi module (ESP8266/ESP32), Buzzer or LED indicator, Relay module, Jumper wires, Breadboard, Power supply or USB cable.

### **Procedure:**

Step 1:

- Connect the ultrasonic sensor to monitor water tank levels.

Step 2:

- Attach a water flow sensor to measure water usage and detect leakages.

Step 3:

- Interface a Wi-Fi module for IoT connectivity to transmit data.

Step 4:

- Write and upload Arduino IDE code for real-time water monitoring and alerts.

Step 5:

- Link the system to an IoT platform for data visualization and control.

Step 6:

- Test the system for water level changes and flow rates to ensure proper functioning.

**Result:**

Water levels and usage were successfully monitored using sensors, and real-time data was transmitted to the IoT platform. The system detected leakages, optimized water usage, and provided alerts for abnormal conditions effectively.

