

Exp.No: 1

IoT-BASED SOIL HEALTH MONITORING SYSTEM

AIM:

To develop and evaluate an IoT-based soil health monitoring system that utilizes sensors, connectivity, and data analytics to provide real-time information about soil parameters.

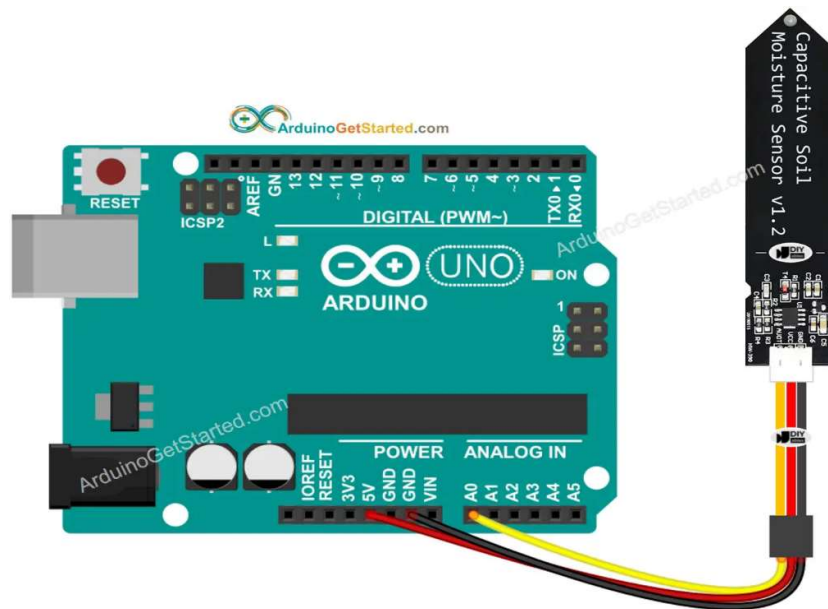
COMPONENTS REQUIRED:

- Soil Sensors
- IoT Gateway
- Connectivity (Wi-Fi, cellular networks, LPWAN, etc.)
- Cloud Platform
- Data Analytics
- Mobile/Desktop Applications
- Power Supply

Theory:

IoT-based soil health monitoring systems have immense potential to transform agriculture by providing farmers with accurate, real-time information about their soil conditions. By leveraging sensors, connectivity, and cloud-based analytics, these systems enable farmers to optimize their farming practices, conserve resources, and increase crop yields. As technology continues to advance, we can expect further advancements in IoT-based soil health monitoring systems, leading to more sustainable and efficient agriculture practices.

CIRCUIT DIAGRAM:



PROCEDURE:

1. Install Soil Sensors: Place the soil sensors at desired locations in the agricultural field, ensuring they are inserted into the soil at an appropriate depth.
2. Connect Sensors to IoT Gateway: Connect the soil sensors to the IoT gateway device using the designated wiring or wireless communication protocols specified by the manufacturer.
3. Establish Network Connectivity: Set up a reliable network connection, such as Wi-Fi or cellular, for the IoT gateway to transmit data. Configure the gateway to connect to the network.
4. Configure Cloud Platform: Set up an account on a cloud platform that supports IoT data storage and processing. Configure the platform to receive and store the data transmitted by the sensors. Set up necessary security measures.
5. Develop Data Visualization: Implement a simple data visualization tool to display the collected data in real time. This can be a basic dashboard or graph that shows soil parameters such as moisture, temperature, or pH level.
6. Test and Monitor the System: Conduct a series of tests to ensure the system's proper functioning. Verify the accuracy of the data collected by comparing it

with manual soil testing. Monitor the system regularly for data consistency and address any issues that may arise.

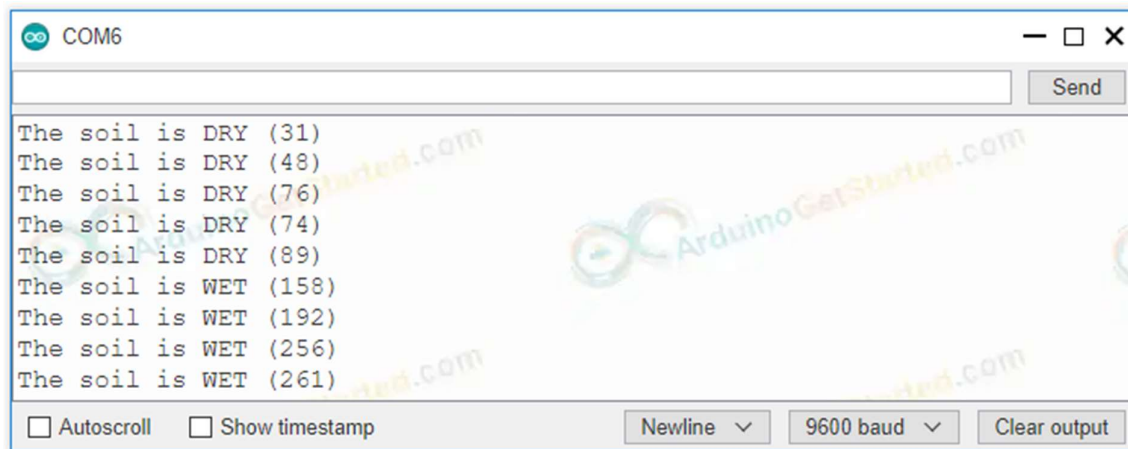
CODE:

```
#define AOUT_PIN A0 // Arduino pin that connects to AOUT pin of moisture sensor
#define THRESHOLD 100 // CHANGE YOUR THRESHOLD HERE

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

void loop() {
    int value = analogRead(AOUT_PIN); // read the analog value from sensor
    if (value < THRESHOLD)
        Serial.print("The soil is DRY (");
    else
        Serial.print("The soil is WET (");
    Serial.print(value);
    Serial.println(")");
    delay(500);
}
```

OUTPUT:



```
COM6
The soil is DRY (31)
The soil is DRY (48)
The soil is DRY (76)
The soil is DRY (74)
The soil is DRY (89)
The soil is WET (158)
The soil is WET (192)
The soil is WET (256)
The soil is WET (261)
```

INFERENCE:

The IoT-based soil health monitoring system provides real-time information about soil parameters, enabling farmers to make informed decisions and optimize farming practices. It allows for remote accessibility and timely interventions based on alerts and notifications. By leveraging data-driven insights, farmers can improve resource management, increase crop yields, and promote sustainable agriculture practices.

RESULT:

The IoT-based soil health monitoring system provides real-time data on soil parameters, enabling informed decision-making for farmers. By optimizing irrigation, fertilization, and crop management practices based on the collected data, it can lead to improved resource efficiency and increased crop yields.

Exp.No: 2

AIR QUALITY MONITORING SYSTEM

AIM:

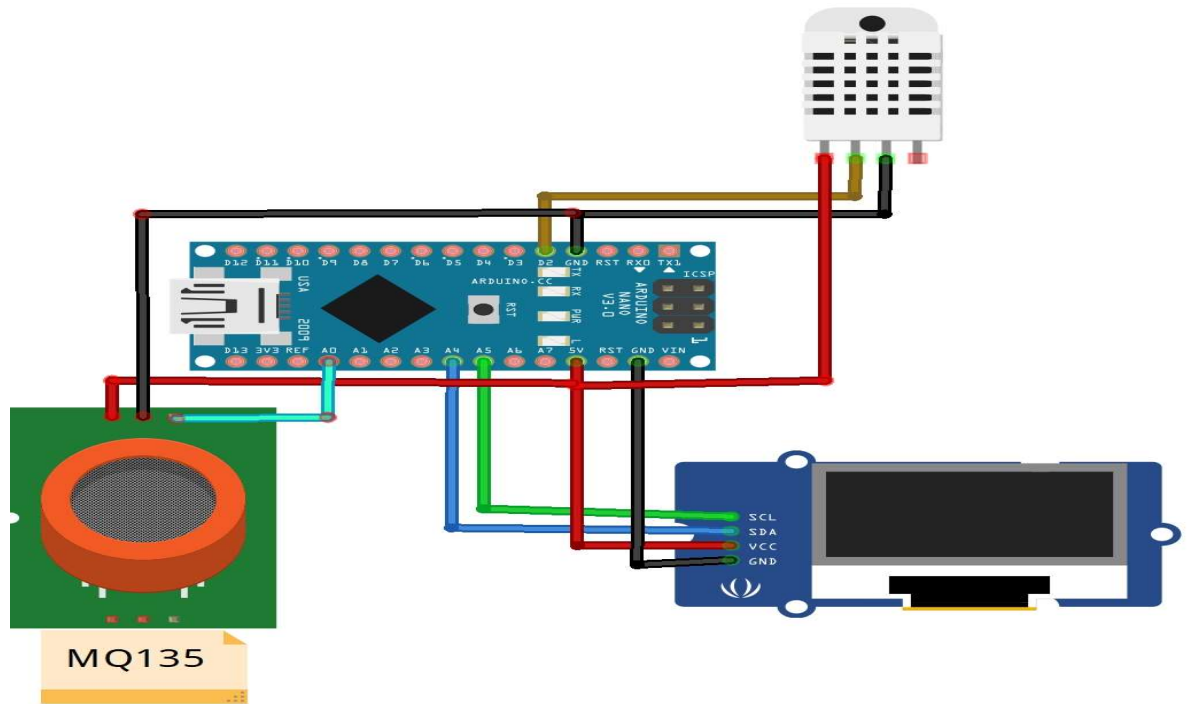
Developing an IOT-based air quality monitoring system to ensure healthier environments by accurately measuring and analyzing air pollutants.

COMPONENTS REQUIRED:

- Arduino Nano
- MQ135 Air Quality Gas Sensor
- Jumper wires
- LCD(I2C) display
- DHT11

THEORY:

The objective of this project is to design and develop an air quality monitoring system using Arduino. The system will continuously measure the concentration of pollutants in the surrounding air and display the results in a user-friendly manner.



fritzing

PROCEDURE:

- Take the Arduino board and connect the MQ135 sensor to the appropriate pins on the Arduino board.
- Connect the DHT11 sensor and LCD display to the Arduino, ensuring the correct wiring connections.
- Open the Arduino IDE application on your computer, write the code for your project, and install any necessary libraries required for the sensors and display.
- Connect the Arduino board to your device (e.g., computer) using a USB cable.
- Compile the code in the Arduino IDE and upload it to the Arduino board.
- Check the output of air quality, humidity, and temperature readings in both the Serial Monitor (accessible in the Arduino IDE) and the LCD display.

CODE:

```
#include <Wire.h>
#include <dht.h>
```

```

dht DHT;
#define DHT11_PIN D2

void setup()
{
  pinMode(A0,INPUT);
  Serial.begin(9600);
}
void loop() {

  int mq_value=analogRead(A0);
  Serial.println("Air quality monitoring system");
  Serial.println("Air Quality is:");
  Serial.println(mq_value);
  Serial.println();
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(1000);
}

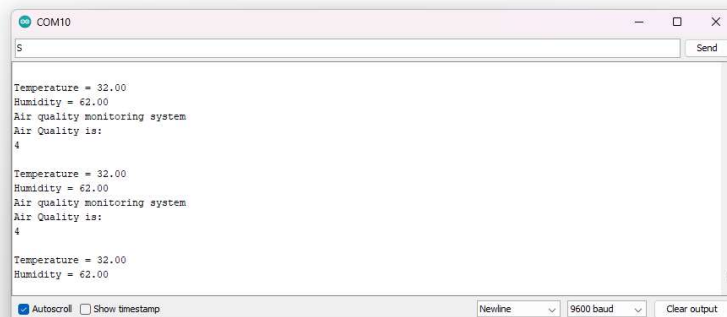
```

OUTPUT:

```

1
2
3 #include <Wire.h>
4 #include <dht.h>
5
6 dht DHT;
7
8 #define DHT11_PIN 7
9
10
11 void setup()
12 {
13   pinMode(A0,INPUT);
14   Serial.begin(9600);
15 }
16 void loop() {
17
18   int mq_value=analogRead(A0);
19   Serial.println("Air quality monitoring system");
20   Serial.println("Air Quality is:");
21   Serial.println(mq_value);
22   Serial.println();
23   int chk = DHT.read11(DHT11_PIN);
24   Serial.print("Temperature = ");
25   Serial.println(DHT.temperature);
26   Serial.print("Humidity = ");
27   Serial.println(DHT.humidity);
28   delay(1000);
29 }
30
31
32
33

```



INFERENCE:

BECE351E- Internet of Things Lab E-Record

Deploying an IoT-based air quality monitoring system allows real-time tracking of pollutants, enabling informed decisions for better environmental health and well-being.

RESULT:

Hence IOT based air quality monitoring system is implemented using ARDUINO, MQ135, DHT11 AND LCD.

Exp.No: 3

Smart Parking System

Aim:

To implement a smart parking system that displays available parking slots

Software used:

Arduino IDE

Components required:

Arduino UNO (1)

IR Proximity Sensors (2)

Servo Motor (1)

16x2 LCD i2c Display (1)

Jumper wires

Theory:

Real-time:

- **Parking Space Detection:** Smart parking systems employ various types of sensors such as ultrasonic, infrared, or magnetic sensors to detect the presence of vehicles in parking spaces. These sensors can be embedded in the ground, mounted on walls, or integrated into parking meters.
- **Data Collection and Analysis:** The sensors collect data about the availability or occupancy status of parking spaces, which is then transmitted to a centralized control system or cloud-based platform. The system analyzes this data to determine the availability of parking spaces in real-time.
- **Real-time Parking Information:** The collected data is made accessible to drivers through mobile applications, dynamic signage, or websites. This enables drivers to locate and navigate to available parking spaces more efficiently, reducing the time spent searching for parking.

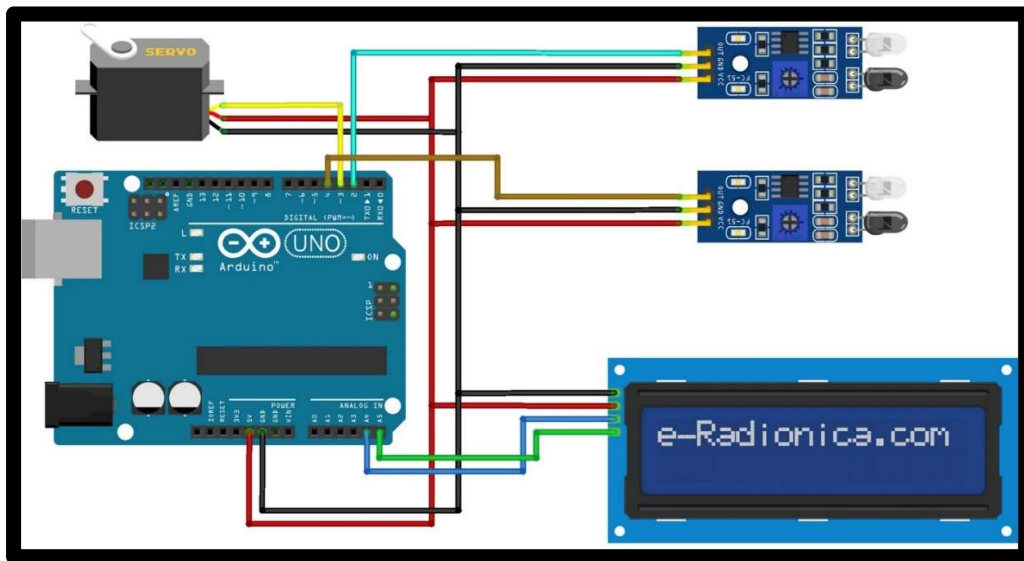
Experiment:

- **Parking Space Detection:** This smart parking system employ infrared, or sensors to detect the entry and exit of vehicles in parking spaces. These sensors are embedded in the ground.
- **Data Collection and Analysis:** The sensors collect data about the availability or occupancy status of parking spaces. The Arduino IDE analyzes this data by a piece of code to determine the availability of parking spaces.
- **Real-time Parking Information:** This collected data can be made accessible to drivers. This enables drivers to locate and navigate to available parking spaces more efficiently, reducing the time spent searching for parking. Though this feature is not implemented in this experiment.

Procedure:

- The ground and 5V pins are connected on the breadboard. The ground and VCC pins of all the devices (IR sensors, servo motor and LCD) are connected to these ground and power pins of the Arduino.
- The OUT pins of the IR sensors are connected to the Arduino board directly.
 - Arduino pin 2 – IR sensor 1
 - Arduino pin 4 – IR sensor 2
 - Arduino pin ~3 – To the servo motor

Circuit Diagram:



Codes:

For 16x2 LCD i2c display:

```
#include <Wire.h>
void setup(){
  Wire.begin();
  Serial.begin(9600);
  Serial.println("\nI2C Scanner");
}
void loop()
{
  byte error, address;
  int Devices;
  Serial.println("Scanning...");
  Devices = 0;
  for(address = 1; address < 127; address++ )
  {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
    }
  }
}
```

```

        Serial.print(address,HEX);
        Serial.println("  !");
        Devices++;
    }
    else if (error==4)
    {
        Serial.print("Unknown error at address 0x");
        if (address<16)
            Serial.print("0");
        Serial.println(address,HEX);
    }
}
if (Devices == 0)
    Serial.println("No I2C devices found\n");
else
    Serial.println("done\n");
delay(5000);
}

```

For Car Parking System:

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,16,2); //Change the HEX address
#include <Servo.h>
Servo myservo1;
int IR1 = 2;
int IR2 = 4;
int Slot = 4;           //Enter Total number of parking Slots
int flag1 = 0;
int flag2 = 0;
void setup()
{
    lcd.begin();
    lcd.backlight();
    pinMode(IR1, INPUT);
    pinMode(IR2, INPUT);

    myservo1.attach(3);
    myservo1.write(100);
}

```

```

    lcd.setCursor (0,0);
    lcd.print("    ARDUINO    ");
    lcd.setCursor (0,1);
    lcd.print(" PARKING SYSTEM ");
    delay (2000);
    lcd.clear();
}
void loop()
{
    if(digitalRead (IR1) == LOW && flag1==0)
    {
        if(Slot>0)
        {
            flag1=1;
            if(flag2==0)
            {
                myservo1.write(0); Slot = Slot-1;
            }
        }
        else
        {
            lcd.setCursor (0,0);
            lcd.print("    SORRY :(    ");
            lcd.setCursor (0,1);
            lcd.print(" Parking Full ");
            delay (3000);
            lcd.clear();
        }
    }
    if(digitalRead (IR2) == LOW && flag2==0)
    {
        flag2=1;
        if(flag1==0)
        {
            myservo1.write(0);
            Slot = Slot+1;
        }
    }
    if(flag1==1 && flag2==1)
    {
        delay (1000);
        myservo1.write(100);
        flag1=0, flag2=0;
    }
    lcd.setCursor (0,0);

```

```

    lcd.print("    WELCOME!    ");
    lcd.setCursor (0,1);
    lcd.print("Slot Left: ");
    lcd.print(Slot);
}

```

For Car Parking System without any LCD:

```

#include <Servo.h>
Servo myservo1;
int IR1 = 2;
int IR2 = 4;

int Slot = 4;
    //Enter Total number of parking Slots
int flag1 = 0;
int flag2 = 0;
void setup(){
    pinMode(IR1, INPUT);
    pinMode(IR2, INPUT);

    myservo1.attach(3);
    myservo1.write(100);
    Serial.begin(9600);
    Serial.println("  ARDUINO  ");
    Serial.println(" PARKING SYSTEM ");
    delay (2000);
}
void loop()
{
    if(digitalRead(IR1) == LOW && flag1 == 0)
    {
        if(Slot > 0)
        {
            flag1 = 1;
            if(flag2 == 0)
            {
                myservo1.write(0);
                Slot = Slot - 1;
                Serial.println("Open gate to get in");
                //    Serial.print(Slot);
            }
        }
    }
}

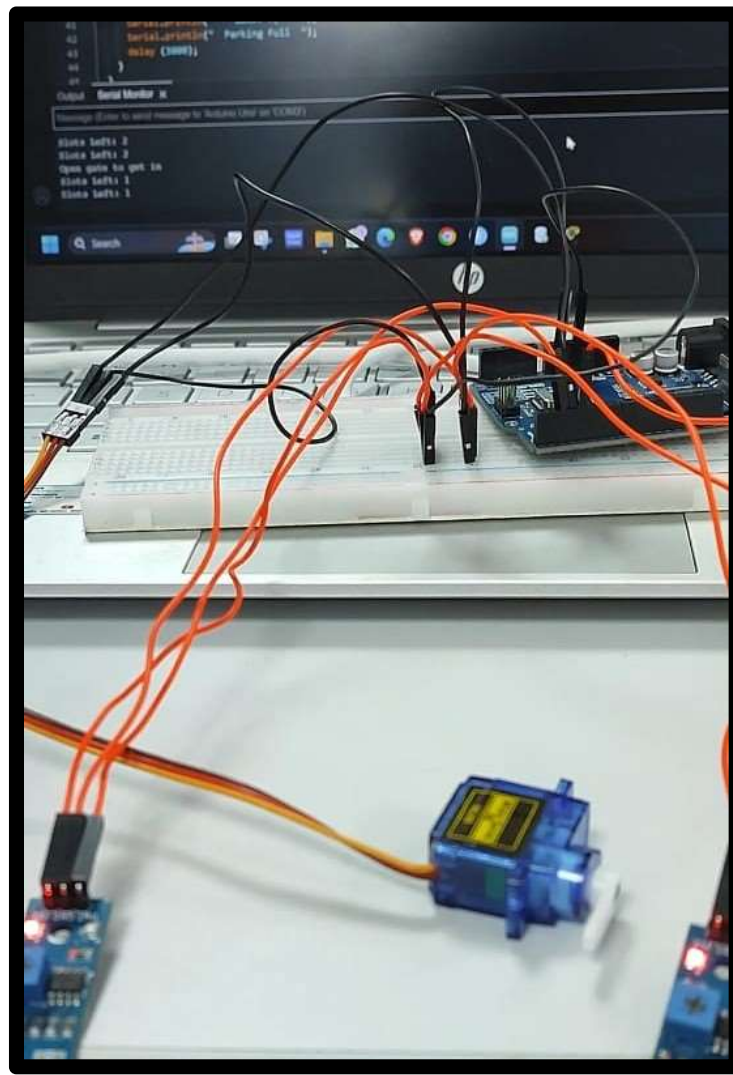
```

```

    }
  }
  else
  {
    Serial.println("  SORRY :(  ");
    Serial.println(" Parking Full ");
    delay (3000);
  }
}
if(digitalRead(IR2) == LOW && flag2 == 0)
{
  flag2 = 1;
  if(flag1 == 0)
  {
    myservo1.write(0);
    Slot = Slot + 1;
    Serial.println("Open gate to get out");
  }
}
if(flag1 == 1 && flag2 == 1)
{
  delay (1000);
  myservo1.write(100);
  flag1 = 0;
  flag2 = 0;
}
Serial.print("Slots Left: ");
Serial.println(Slot);
delay(3000);
}

```

Output:



```
Output Serial Monitor x
Not connected. Select a board and a port to connect automatically. New Line 9600 baud

ARDUINO
PARKING SYSTEM
Slots Left: 4
Slots Left: 4
Slots Left: 4
Slots Left: 4
Slots Left: 4
Slots Left: 4
Slots Left: 4
Slots Left: 4
Slots Left: 4
Open gate to get in
Slots Left: 3
Slots Left: 3
Slots Left: 3
Slots Left: 3
Slots Left: 3
Slots Left: 3
Slots Left: 3
Open gate to get in
Slots Left: 2
Slots Left: 2
Open gate to get in
Slots Left: 1
```



```
Output Serial Monitor x
Not connected. Select a board and a port to connect automatically.
New Line 9600 baud

Slots Left: 2
Slots Left: 2
Slots Left: 2
Open gate to get in
Slots Left: 1
Slots Left: 1
Open gate to get in
Slots Left: 0
Slots Left: 0
Slots Left: 0
SORRY :(
Parking Full
Slots Left: 0
Slots Left: 0
Slots Left: 0
Open gate to get out
Slots Left: 1
Slots Left: 1
Slots Left: 1
Slots Left: 1
Slots Left: 1
Slots Left: 1
Slots Left: 1
Slots Left: 1
Slots Left: 1
Slots Left: 1
1

Ln 66, Col 2 Arduino Uno (not connected)
```

Result:

Smart Parking System was successfully implemented using Arduino.

Exp.No: 4

IoT-BASED HEALTH CARE AND FITNESS MONITORING SYSTEM

AIM:

To develop and evaluate an IoT-based healthcare and fitness monitoring system is to enable real-time monitoring, analysis, and personalized care, ultimately improving healthcare outcomes.

COMPONENTS REQUIRED:

- Sensors (heart rate sensors, temperature sensors)
- Medical devices
- Connectivity
- Cloud-based platforms for data storage and analysis
- Mobile applications or web interfaces for user interaction

Theory:

IoT-based health care and fitness monitoring systems utilize interconnected devices and sensors to collect and transmit real-time data on vital signs, activity levels, and other health parameters. These systems enable remote monitoring, analysis, and personalized care, enhancing healthcare decision-making. Wearable devices and environmental sensors track data, which is wirelessly transmitted to a central hub or cloud platform for analysis. The connectivity and communication between devices facilitate seamless data exchange, empowering healthcare providers to make informed decisions and improve patient outcomes.

PROCEDURE:

1. Identify and select suitable wearable devices, sensors, and medical equipment based on the specific health and fitness parameters to be monitored.
2. Set up the required connectivity infrastructure, such as Wi-Fi or Bluetooth, to enable seamless communication between devices.

3. Integrate the selected devices and sensors into a centralized system, ensuring they can collect and transmit data to a cloud-based platform.
4. Develop or deploy a cloud-based platform for data storage, analysis, and visualization, enabling healthcare providers to access and interpret real-time data.
5. Develop or utilize mobile applications or web interfaces for users to interact with the system, view their health data, and receive personalized recommendations.
6. Continuously monitor, evaluate, and update the system to ensure data accuracy, security, and compatibility with emerging technologies and medical advancements.

CODE:

```
#include <Wire.h>

#include "MAX30105.h"

#include "heartRate.h"

MAX30105 particleSensor;

const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is good.

byte rates[RATE_SIZE]; //Array of heart rates

byte rateSpot = 0;

long lastBeat = 0; //Time at which the last beat occurred

float beatsPerMinute;

int beatAvg;

void setup() {

  Serial.begin(115200);

  Serial.println("Initializing...");

  // Initialize sensor

  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {

    Serial.println("MAX30102 was not found. Please check wiring/power. ");

    while (1);

  }
```

```

Serial.println("Place your index finger on the sensor with steady pressure.");

particleSensor.setup(); //Configure sensor with default settings

particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate
sensor is running

particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
}

void loop() {

  long irValue = particleSensor.getIR();
  if (checkForBeat(irValue) == true) {
    //We sensed a beat!

    long delta = millis() - lastBeat;

    lastBeat = millis();

    beatsPerMinute = 60 / (delta / 1000.0);

    if (beatsPerMinute < 255 && beatsPerMinute > 20) {
      rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in the array
      rateSpot %= RATE_SIZE; //Wrap variable

      //Take average of readings

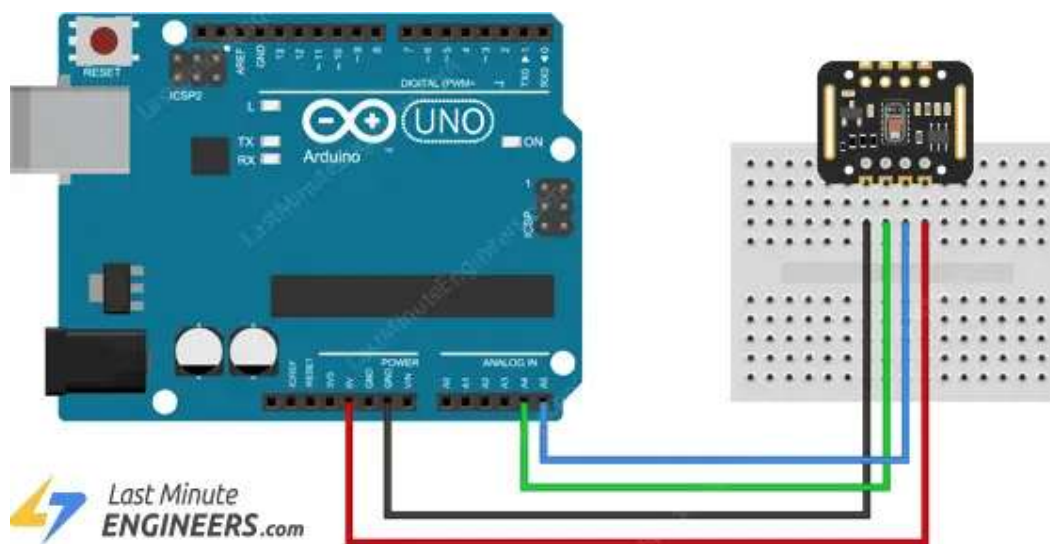
      beatAvg = 0;
      for (byte x = 0 ; x < RATE_SIZE ; x++)
        beatAvg += rates[x];
      beatAvg /= RATE_SIZE;
    }
  }

  Serial.print("IR=");
  Serial.print(irValue);
  Serial.print(", BPM=");

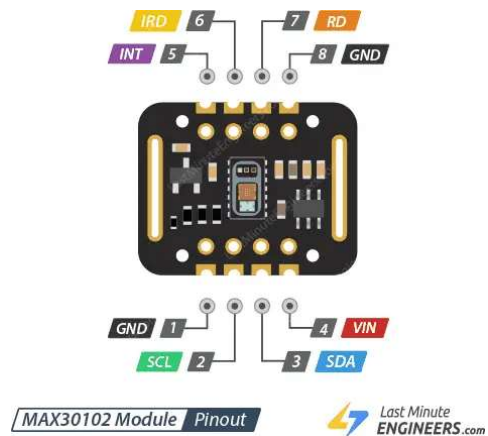
```

```
Serial.print(beatsPerMinute);  
Serial.print(", Avg BPM=");  
Serial.print(beatAvg);  
if (irValue < 50000)  
    Serial.print(" No finger?");  
Serial.println();}
```

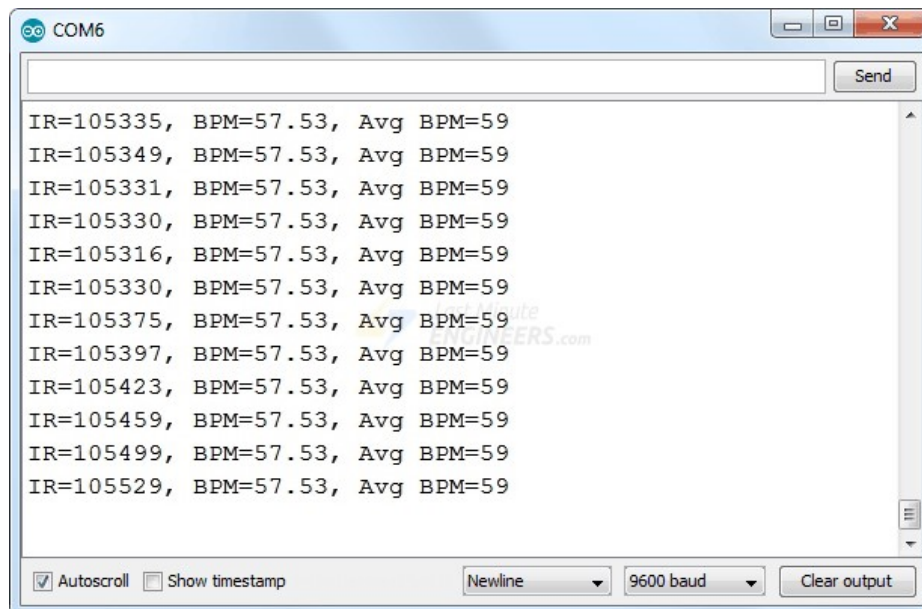
CIRCUIT DIAGRAM:



PIN DIAGRAM OF MAX30102:



OUTPUT:



INFERENCE:

IoT-based health care and fitness monitoring systems revolutionize healthcare by enabling real-time data collection, analysis, and personalized care. These systems leverage wearable devices, sensors, and connectivity to provide continuous monitoring of vital signs and activity levels. The seamless communication between devices and cloud-based platforms empowers healthcare providers to make informed decisions and improve patient outcomes. By integrating IoT technology, these systems pave the way for proactive interventions, early detection of health issues, and better individual health management.

RESULT:

IoT-based health care and fitness monitoring systems offer real-time data collection, analysis, and personalized care. These systems enhance healthcare decision-making, enable proactive interventions, and improve patient outcomes. By leveraging wearable devices, sensors, and connectivity, IoT technology revolutionizes the way we monitor and manage health and fitness.

Exp.No: 5

Environmental monitoring Arduino Uno with DHT 11 Sensor using Node-Red

Aim: Environmental monitoring ArduinoUno with DHT 11 Sensor and using Node-Red Software.

Apparatus/Components Required-

BECE351E- Internet of Things Lab E-Record

- Arduino Uno R3
- Breadboard
- DHT 11 Sensor
- Jumper wires
- Node-RED

THEORY:

Environmental monitoring Arduino Uno with DHT 11 Sensor

WHAT IS RELATIVE HUMIDITY?

The **DHT11** measures *relative humidity*. Relative humidity **is the amount of water vapor in air vs. the saturation point of water vapor in air**. At the **saturation point, water vapor starts to condense and accumulate on surfaces forming dew**. The **saturation point changes with air temperature**. Cold air can hold less water vapor before it becomes saturated, and hot air can hold more water vapor before it becomes saturated.

The formula to calculate relative humidity is:

$$RH = \left(\frac{\rho_w}{\rho_s} \right) \times 100\%$$

RH : Relative Humidity

ρ_w : Density of water vapor

ρ_s : Density of water vapor at saturation

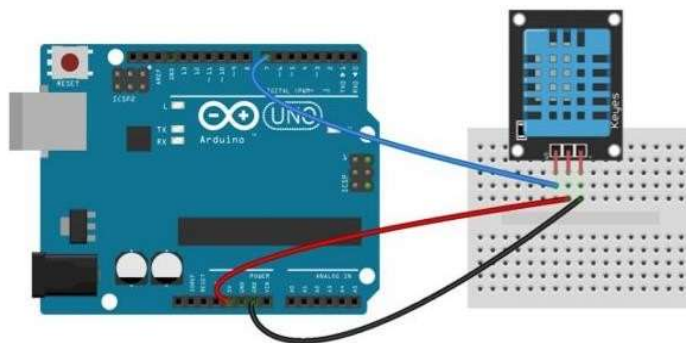
Ranges and accuracy of the **DHT11**:

- Humidity Range: 20-90% RH
- Humidity Accuracy: $\pm 5\%$ RH
- Temperature Range: 0-50 °C
- Temperature Accuracy: $\pm 2\%$ °C
- Operating Voltage: 3V to 5.5V

Relative humidity is expressed as a percentage.

At 100% RH, condensation occurs, and at 0% RH, the air is completely dry.

Circuit schematic-



Procedure:

Hardware Setup:

- Connect the DHT11 sensor to the Arduino Uno.
- Connect the VCC pin of the DHT11 to the 5V pin on the Arduino.
- Connect the GND pin of the DHT11 to the GND pin on the Arduino.
- Connect the data pin of the DHT11 to any digital pin on the Arduino (e.g., pin 2).
- Connect the Arduino Uno to your computer via USB.

Software Setup:

- Install the Arduino IDE on your computer if you haven't already.
- Open the Arduino IDE and go to "File" -> "Examples" -> "DHT" -> "DHTtester".
- Upload the DHTtester sketch to the Arduino Uno.

Node-RED Setup:

- Install Node-RED on your computer or a Raspberry Pi if you haven't already.
- Launch Node-RED and open the Node-RED editor in your web browser.
- Install the necessary Node-RED nodes:
 - Click on the menu icon in the top-right corner of the editor.
 - Select "Manage palette" from the dropdown menu.
- In the "Palette" tab, search for and install the following nodes:
 - node-red-contrib-arduino
 - node-red-dashboard

Node-RED Flow:

- Drag and drop an "Arduino" node from the "Input" section onto the flow canvas.
- Double-click the Arduino node to configure it.
- Select the serial port that corresponds to your Arduino Uno.
- Set the baud rate to match the one specified in your Arduino code (usually 9600).
- Drag and drop a "Function" node from the "Function" section onto the flow canvas.
- Double-click the Function node to open its editor
- Connect the Arduino node's output to the Function node's input.
- Drag and drop a "Dashboard" node from the "Dashboard" section onto the flow canvas.
- Connect the Function node's output to the Dashboard node's input.

Dashboard Configuration:

- Double-click the Dashboard node to configure it.
- In the "Group" field, enter a name for your dashboard group (e.g., "Environmental Monitoring").

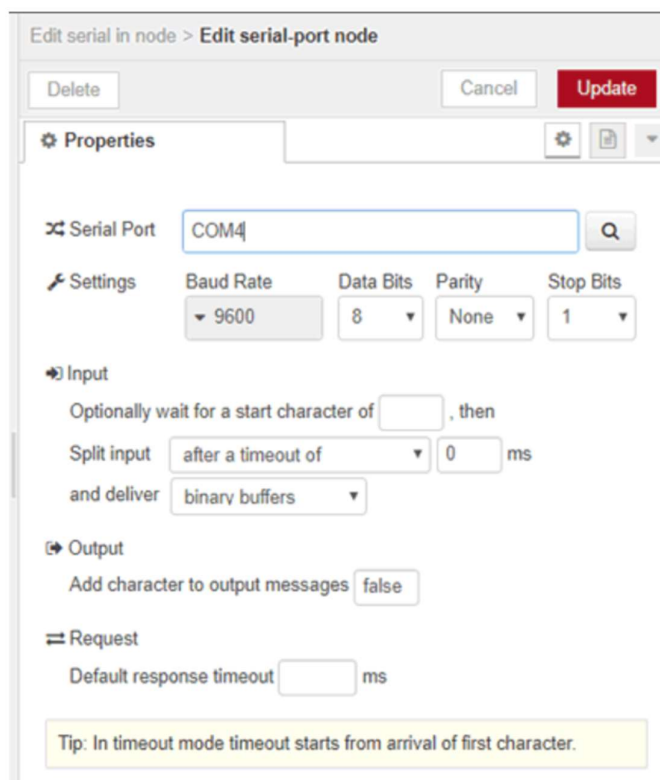
- Click "Done" to close the configuration dialog.
- Deploy the Node-RED flow by clicking the "Deploy" button in the top-right corner of the editor.

Accessing the Dashboard:

- Open a web browser and enter the URL: <http://<your-node-red-ip-address>:1880/ui>
- Replace "<your-node-red-ip-address>" with the IP address or hostname of your Node-RED server.
- You should now see the dashboard with the temperature and humidity readings from the DHT11 sensor.

PROPERTIES SET FOR EACH NODE WITH EXPLANATION:

Editing serial node:



Edit serial in node > Edit serial-port node

Delete Cancel Update

⚙ Properties

Serial Port COM4

Settings Baud Rate 9600 Data Bits 8 Parity None Stop Bits 1

Input

Optionally wait for a start character of , then

Split input after a timeout of 0 ms

and deliver binary buffers

Output

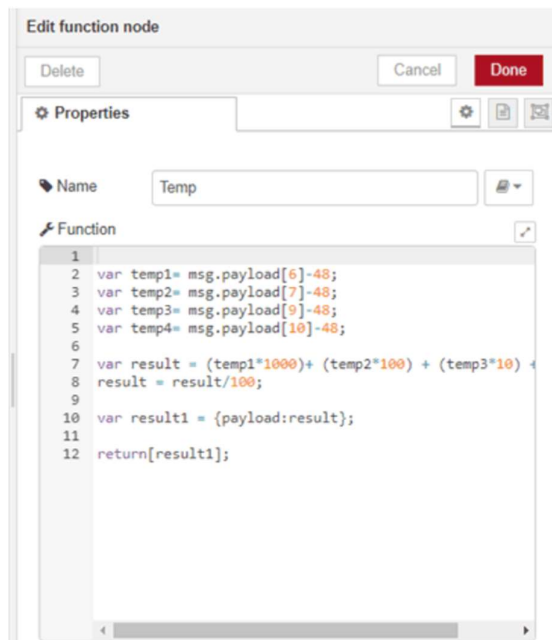
Add character to output messages false

Request

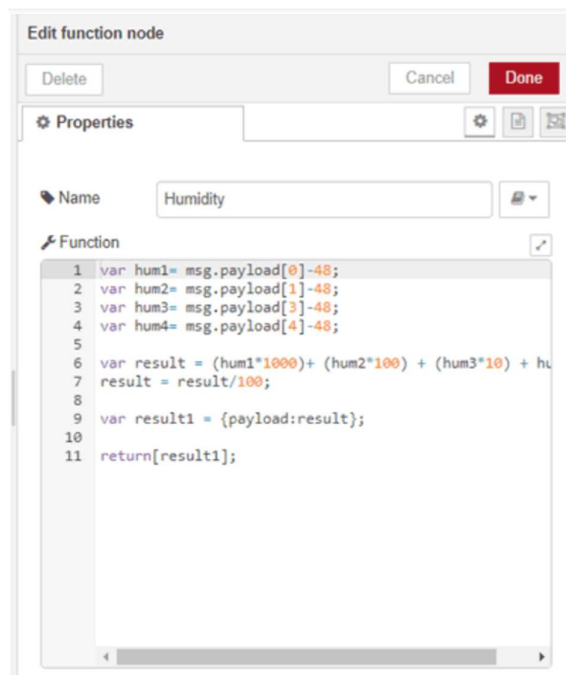
Default response timeout ms

Tip: In timeout mode timeout starts from arrival of first character.

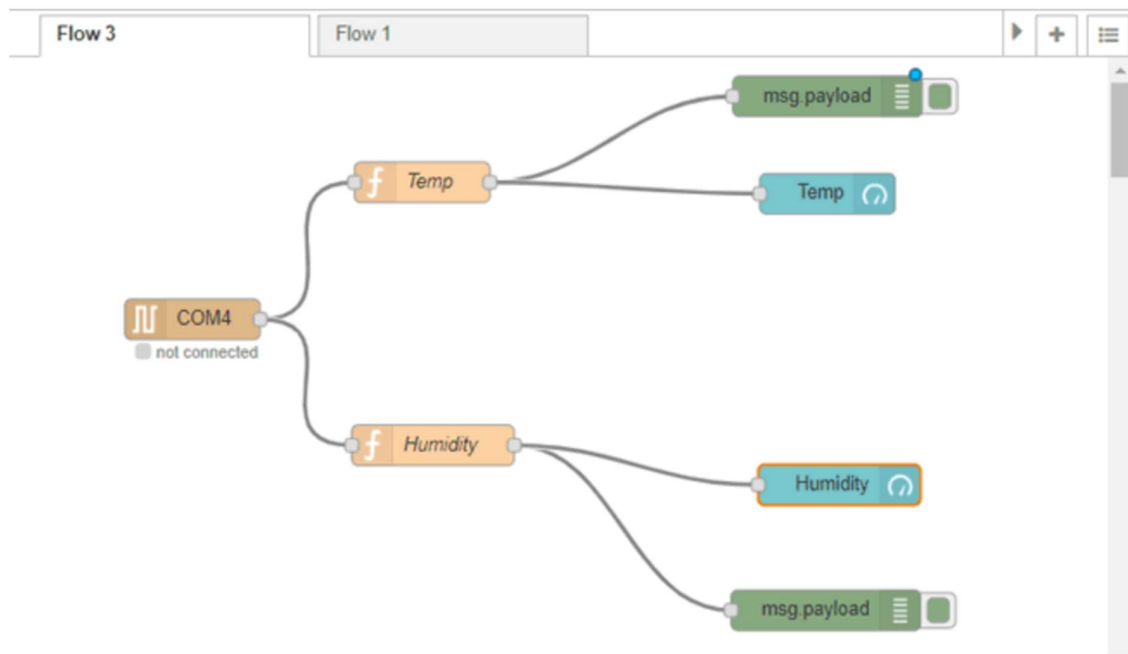
Editing Temperature node:



Editing Humidity node:



Node diagram:



Program:

```
// Read DHT11 sensor and send serially to PC
#include<DHT.h>

// Include AdafruitDHT11 Sensors Library
#define DHTPIN 7

// DHT11 Output Pin connection
#define DHTTYPE DHT11

// DHT Type is DHT11
DHT dht (DHTPIN, DHTTYPE); // Initialize DHT sensor

void setup() {
  dht.begin();
```

```

Serial.begin(9600); // To see data on serial monitor
}

void loop(){
  float H = dht.readHumidity(); //Read Humidity
  float T = dht.readTemperature(); // Read temperature
  if(isnan(H) || isnan(T)){
    Serial.println("Failed to read from DHT sensor!");
    return;
  } // Combine Humidity and Temperature into single string
  String dhtData= "Humidity "+String(H) + ","+ "Temperature "+String(T);
  Serial.println(dhtData);
  delay(2000); // Wait two seconds between measurements
}

```

Node red temperature code:

```

var temp1= msg.payload[6]-48;
var temp2= msg.payload[7]-48;
var temp3= msg.payload[9]-48;
var temp4= msg.payload[10]-48;
var result = (temp1*1000)+ (temp2*100) + (temp3*10) + temp4;
result = result/100;
var result1 = {payload:result};
return[result1];

```

Nod red Humidity code:

```

var hum1= msg.payload[0]-48;
var hum2= msg.payload[1]-48;
var hum3= msg.payload[3]-48;
var hum4= msg.payload[4]-48;

```

```
var result = (hum1*1000)+ (hum2*100) + (hum3*10) + hum4;
```

```
result = result/100;
```

```
var result1 = {payload:result};
```

```
return[result1];
```

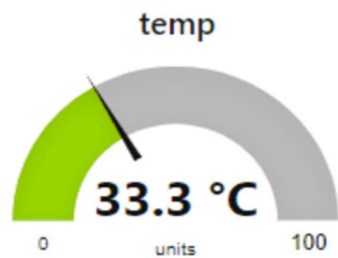
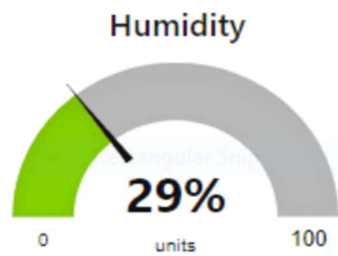
Node red code:

```
[{"id":"74e0880a.c777f8","type":"tab","label":"Flow 3","disabled":false,"info":"","{
  "id":"e26c53af.cca2b","type":"serial in","z":"74e0880a.c777f8","name":"","serial":"58430f06.f23e4","x":137.5173568725586,"y":193.7500286102295,"wires":[["e45d22bf.57606","e06252b0.156d2"]]},{"id":"e45d22bf.57606","type":"function","z":"74e0880a.c777f8","name":"Temp","func":"\nvar temp1= msg.payload[6]-48;\nvar temp2= msg.payload[7]-48;\nvar temp3= msg.payload[9]-48;\nvar temp4= msg.payload[10]-48;\n\nvar result = (temp1*1000)+ (temp2*100) + (temp3*10) + temp4; \nresult = result/100;\n\nvar result1 = {payload:result};\n\nreturn[result1];","outputs":1,"noerr":0,"x":307.07645416259766,"y":92.56599521636963,"wires":[["d914ac8d.cd404","9f2a37c9.0d8f88"]]},{"id":"d914ac8d.cd404","type":"debug","z":"74e0880a.c777f8","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":581.3498992919922,"y":37.79081916809082,"wires":[]},{"id":"e06252b0.156d2","type":"function","z":"74e0880a.c777f8","name":"Humidity","func":"var hum1= msg.payload[0]-48;\nvar hum2= msg.payload[1]-48;\nvar hum3= msg.payload[3]-48;\nvar hum4= msg.payload[4]-48;\n\nvar result = (hum1*1000)+ (hum2*100) + (hum3*10) + hum4; \nresult = result/100;\n\nvar result1 = {payload:result};\n\nreturn[result1];","outputs":1,"noerr":0,"x":315.0911521911621,"y":286.6883945465088,"wires":[["f35a3030.d7841","f7b0f890.f22e18"]]},{"id":"f35a3030.d7841","type":"debug","z":"74e0880a.c777f8","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":582.6041069030762,"y":361.12417221069336,"wires":[]},{"id":"9f2a37c9.0d8f88","type":"ui_gauge","z":"74e0880a.c777f8","name":"Temp","group":"d5da3e9b.46abf","order":0,"width":0,"height":0,"gtype":"gage","title":"temp","label":"units","format":"{{value}} °C","min":0,"max":100,"colors":["#00b500","#e6e600","#ca3838"],"seg1":"","seg2":"","x":570.0998001098633,"y":99.71789360046387,"wires":[]},{"id":"f7b0f890.f22e18","type":"ui_gauge","z":"74e0880a.c777f8","name":"Humidity","group":"d5da3e9b.46abf","order":1,"width":0,"height":0,"gtype":"gage","title":"Humidity","label":"units","format":"{{value}} %","min":0,"max":100,"colors":["#00b500","#e6e600","#ca3838"],"seg1":"","seg2":"","x":583.8411865234375,"y":288.4635429382324,"wires":[]},{"id":"58430f06.f23e4","type":"serial-port","z":"","serialport":"COM4","serialbaud":"9600","databits":"8","parity":"none","stopbits":"1","waitfor":"","newline":"0","bin":"bin","out":"time","addchar":"false","responsetimeout":"","{
  "id":"d5da3e9b.46abf","type":"ui_group","z":"","name":"DHT11","tab":"784aac14.5c2404","order":2,"disp":true,"width":"6","collapse":false},{"id":"784aac14.5c2404","type":"ui_tab","z":"","name":"Station","icon":"dashboard","order":1,"disabled":false,"hidden":false}]
```

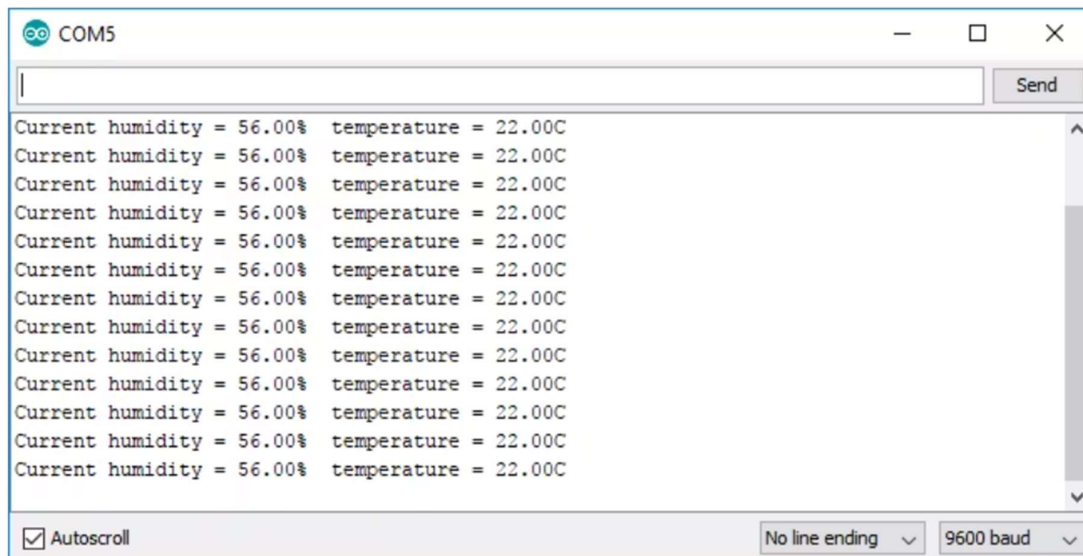
OUTPUT:

node red output

DHT11



Arduino output:



INFERENCE-

The project you are referring to involves interfacing an Arduino Uno with a DHT11 sensor and using Node-Red to monitor the temperature and humidity. The DHT11 sensor is a popular 3-pin sensor that can measure temperature and humidity. The sensor works with a one-wire protocol and is easy to use with development boards like Arduino. The Vcc pin of DHT11 is connected with Arduino's 3.3v pin, and the GND pin is connected with Arduino's GND pin1. Programming Arduino UNO for this project doesn't require much effort as it only uses one library for the DHT sensor1. The Arduino board has to measure the temperature and humidity value from the DHT11 sensor and send it out serially using a COM port. This serial information will then be processed by the Node-RED

RESULT-

we have implemented the weather monitoring system using DHT11 sensor in Arduino IDE and as well as NODE-RED software.

Exp.No: 6

IoT enabled accident prevention and detection system

AIM-

To make an accident detection and prevention device using the concepts of IoT

APPARATUS-

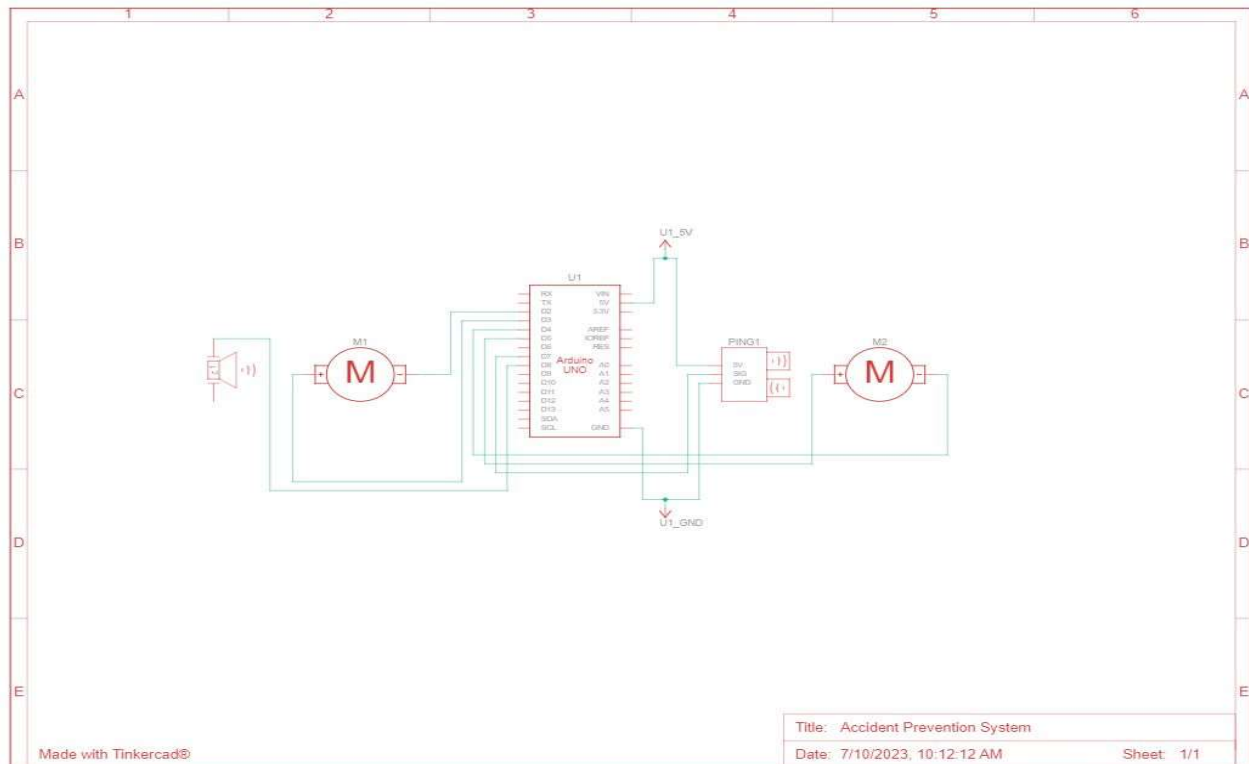
- 1.Arduino UNO
- 2.Breadboard
- 3.Ultrasonic sensor
- 4.Buzzer

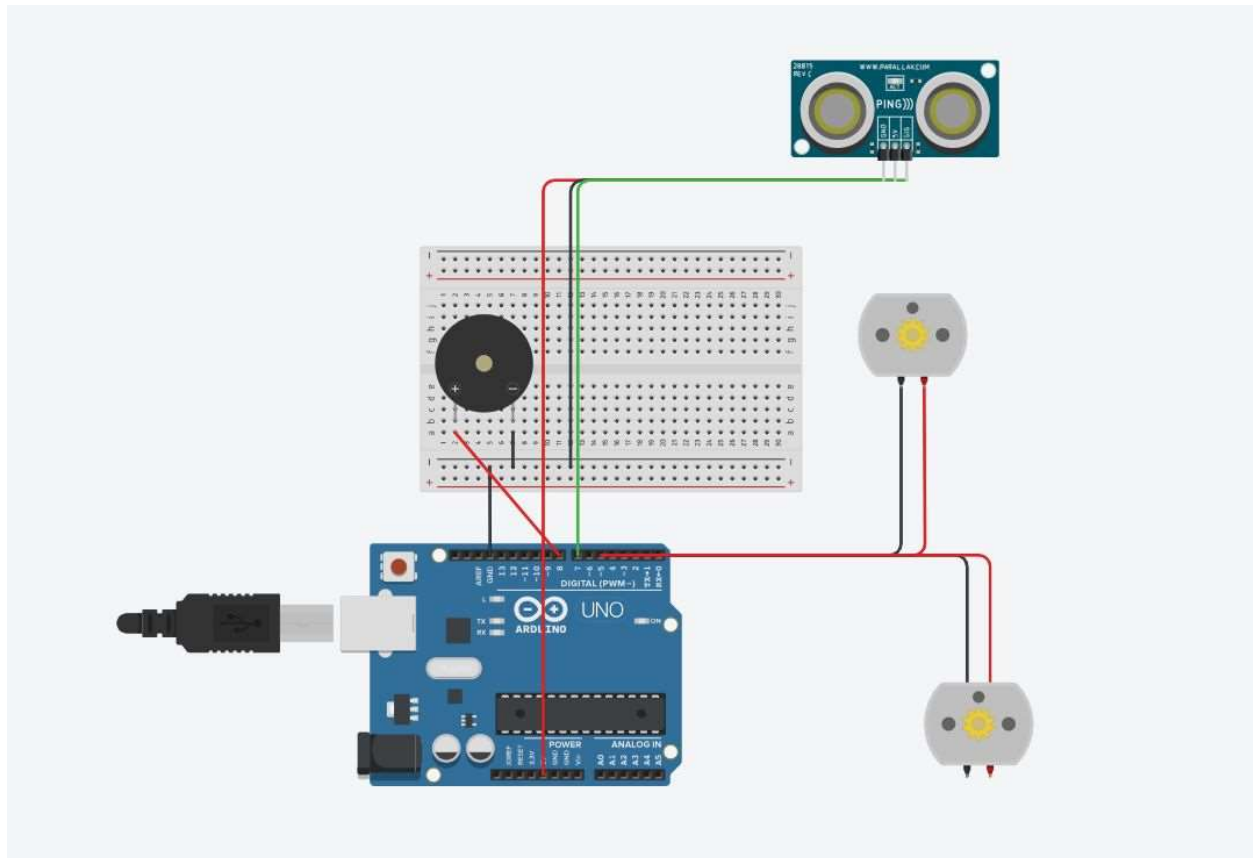
Theory -

Accident detection and prevention devices using IoT are innovative systems designed to enhance safety and mitigate the impact of accidents in various domains such as transportation, industries, and personal safety. These devices leverage the power of the Internet of Things (IoT) to gather real-time data, analyze it, and trigger appropriate actions to prevent accidents or minimize their consequences.

The primary goal of such devices is to detect potential accidents or hazardous situations promptly and provide timely alerts or interventions to prevent or mitigate the damage.

CIRCUIT DIAGRAM-





CODE-

```
int buzzer = 8;  
int motor1Pin1 = 2;  
int motor1Pin2 = 3;  
int motor2Pin1 = 4;  
int motor2Pin2 = 5;  
int trigPin = 9;  
int echoPin = 10 ;
```

```
void setup()  
{
```

```
  pinMode (trigPin, OUTPUT);  
  pinMode(echoPin, INPUT);  
  pinMode (motor1Pin1, OUTPUT);  
  pinMode (motor1Pin2, OUTPUT);  
  pinMode (motor2Pin1, OUTPUT);
```

```

pinMode (motor2Pin2, OUTPUT);
pinMode (buzzer, OUTPUT);
digitalWrite(motor1Pin1, HIGH);
    digitalWrite(motor2Pin1, HIGH);
digitalWrite(motor1Pin2, LOW);
    digitalWrite(motor2Pin2, LOW);
Serial.begin(9600);
}

void loop()
{

    long duration, inches, cm;
    digitalWrite(trigPin,LOW);
    delay(2);
    digitalWrite(trigPin, HIGH);
    delay(2);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH);
    cm = microsecondsToCentimeters(duration);

    if(cm < 5 && cm > 0){

        Serial.print("Collision Detected: ");
        Serial.println(cm);
        digitalWrite(motor1Pin1, LOW);
        digitalWrite(motor2Pin1, LOW);
        digitalWrite(buzzer, HIGH);
        delay(5);
        digitalWrite(buzzer,LOW);
    }
    delay(100);
    digitalWrite(motor1Pin1, HIGH);
        digitalWrite(motor2Pin1, HIGH);

}

```

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

INFERENCE -

Hence we can see that if the object comes under the threshold value set in the code, it sends a signal of some sort and stops the vehicle immediately.

RESULT- Therefore the working of an IoT based accident prevention and detection device is shown

Exp.No: 7

IoT-BASED Smart Street Light System

AIM:

To develop and evaluate an IoT-based Smart Street Light using ARDUINO UNO

COMPONENTS REQUIRED:

- Two IR sensors
- Two 220Ω resistors
- IR sensor module
- Light depending resistor LDR
- One 1 KΩ resistor
- Arduino Uno
- Two LEDs

Theory :

Street lights are an essential part of all cities and the highways, that's helpful to prevent accidents and unwanted thefts or robbery. Thousands of Street lights are installed beside the highways and the main roads. But the main problem is these street lights consume about 25-30% of the total energy spent in the city. The normal street lights always glow with high intensity, who consumes high electricity. But in the case of a smart street lights system, it will glow with high intensity if there are vehicles or human movement on the road otherwise the lights will remain dim. Another advantage of this system is, street lights will automatically turn on in the evening, and turn on automatically at day time (presence of sunlight). By using this system we can save enough amount of electricity and off-course money, this saved electricity can be used to lighten few more homes in the rural areas.

PROCEDURE:

First of all we will interface the IR sensor with the Arduino such that:

- Connect the VCC of both sensors with the 5V of the Arduino
- Connect the ground of both sensors with the ground of the Arduino

- Connect the output of the first IR sensor to the digital pin number 2 of the Arduino
- Connect the output of the second IR sensor to the digital pin number 3 of the Arduino. You can use any digital pin according to the programming.

Now we will connect the LEDs

- Connect the first led with the digital pin number 5 of the arduino
- Connect the second led with the digital pin number 6 of the Arduino

CODE:

```
int IR1 = 2;
int IR2 = 3;
int LED1 = 5;
int LED2 = 6;
int LDR = A3;
void setup()
{
  Serial.begin(9600);
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
  pinMode(LDR, INPUT_PULLUP);
}
void loop()
{
  int LDRValue = analogRead(LDR);
  Serial.print("sensor = ");
  Serial.print(LDRValue);
  delay (500);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  Serial.println("It's Bright Outside; Lights status: OFF");
  if (LDRValue > 100 && digitalRead(IR1) == LOW)
  {
    digitalWrite(LED1, HIGH);
    Serial.println("It's Dark Outside; LED1 Lights status: ON");
  }

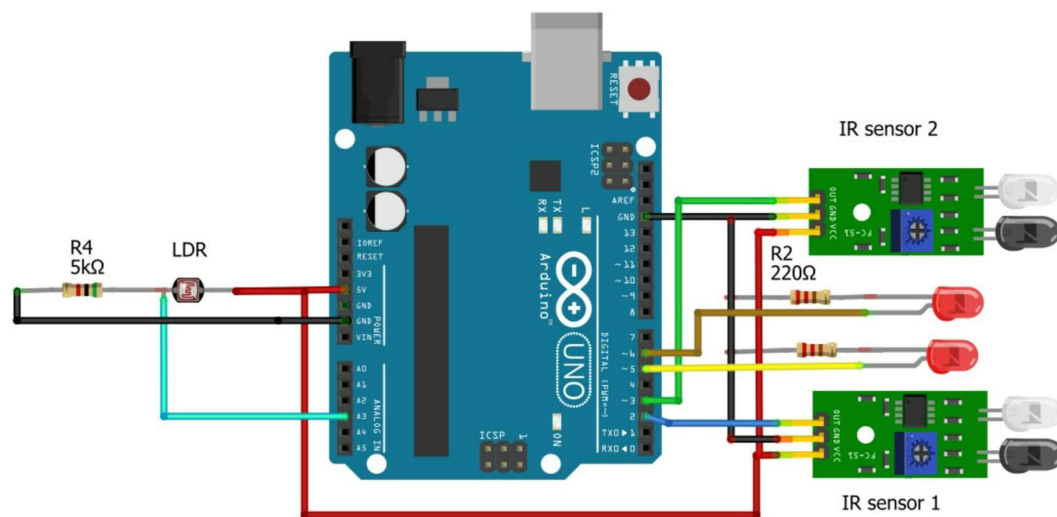
  if (LDRValue > 100 && digitalRead(IR2) == LOW)
```

```

{
  digitalWrite(LED2, HIGH);
  Serial.println("It's Dark Outside; LED2 Lights status: ON");
}
}

```

CIRCUIT DIAGRAM:



OUTPUT:


```
COM5
sensor = 43It's Bright Outside; Lights status: OFF
sensor = 43It's Bright Outside; Lights status: OFF
sensor = 43It's Bright Outside; Lights status: OFF
sensor = 179It's Bright Outside; Lights status: OFF
It's Dark Outside; LED2 Lights status: ON
sensor = 204It's Bright Outside; Lights status: OFF
It's Dark Outside; LED2 Lights status: ON
sensor = 205It's Bright Outside; Lights status: OFF
It's Dark Outside; LED2 Lights status: ON
sensor = 209It's Bright Outside; Lights status: OFF
It's Dark Outside; LED1 Lights status: ON
It's Dark Outside; LED2 Lights status: ON
sensor = 197It's Bright Outside; Lights status: OFF
It's Dark Outside; LED1 Lights status: ON
It's Dark Outside; LED2 Lights status: ON
sensor = 193It's Bright Outside; Lights status: OFF
sensor = 200It's Bright Outside; Lights status: OFF
It's Dark Outside; LED2 Lights status: ON
sensor = 204It's Bright Outside; Lights status: OFF
It's Dark Outside; LED1 Lights status: ON
It's Dark Outside; LED2 Lights status: ON
sensor = 197It's Bright Outside; Lights status: OFF
sensor = 203It's Bright Outside; Lights status: OFF
```

INFERENCE:

1. Smart street light systems are an innovative technology that uses sensors and advanced lighting controls to improve energy efficiency and reduce costs.
2. These systems can automatically adjust the brightness of street lights based on the presence of pedestrians, vehicles, or weather conditions.
3. They can also provide real-time data on energy consumption and performance, allowing for better management and maintenance.
4. Smart street lights can enhance safety by illuminating dark areas and improving visibility for drivers and pedestrians.
5. Additionally, the use of LED lights in these systems can reduce carbon emissions and support sustainability efforts.
6. As cities continue to grow and urbanization increases, smart street light systems have the potential to become an integral part of smart city infrastructure.

RESULT:

The IoT-based smart street light system can improve energy efficiency, enhanced safety, real-time monitoring and control, and scalability and customizability.

Exp No.: 8

Plant Health Monitoring System

AIM:

To design a plant health monitoring using a suitable IoT platform and services

Apparatus used:

- a. Arduino UNO
- b. LRD Sensor
- c. Temperature and Humidity Sensor
- d. Soil Moisture Sensor
- e. Jumper Wires
- f. Relay 5v

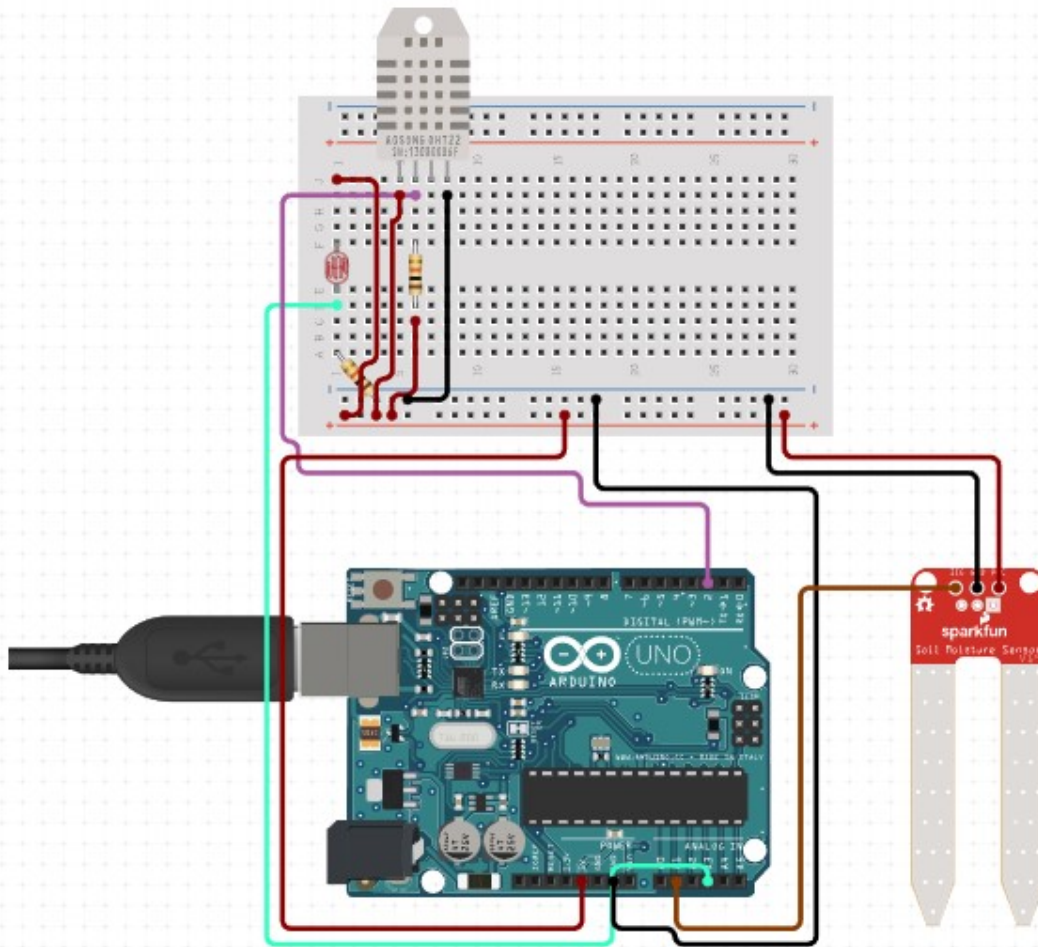
THEORY:

By combining an Arduino UNO with various sensors, such as the LRD sensor, temperature and humidity sensor, and soil moisture sensor, we can create an efficient plant health monitoring system. This system allows us to continuously monitor and control crucial environmental factors affecting the plant's growth. With the ability to automate tasks such as providing additional light or watering the plant, we can ensure optimal conditions for its well-being and enhance its overall health and productivity.

1. The Arduino UNO reads analog input values from the LRD sensor and converts them to a corresponding light intensity level.
2. The temperature and humidity sensor provide analog readings to the Arduino, which are then converted into temperature and humidity values.
3. The soil moisture sensor provides a digital signal to the Arduino indicating the moisture level in the soil.
4. The Arduino analyzes the sensor data and determines if any action is required based on predefined thresholds.
5. If the light intensity is insufficient, the Arduino can activate the relay module to control a lighting system to provide additional light to the plant.
6. If the temperature or humidity levels are outside the desired range, the Arduino can trigger alerts or activate a cooling or heating system accordingly.

7. If the soil moisture level is below a certain threshold, the Arduino can activate a water pump through the relay module to water the plant.

CIRCUIT DIAGRAM:



PROCEDURE:

- i. Gather the required components: Arduino UNO, LRD sensor, temperature and humidity sensor, soil moisture sensor, jumper wires, and a relay module.
- ii. Set up the Arduino IDE software on your computer and connect the Arduino UNO to the computer using a USB cable.
- iii. Connect the LRD sensor to the Arduino UNO using jumper wires.
- iv. Connect the temperature and humidity sensor to the Arduino UNO using jumper wires.
- v. Connect the soil moisture sensor to the Arduino UNO using jumper wires.
- vi. Connect the relay module to the Arduino UNO using jumper wires.
- vii. Write the code for the Arduino UNO in the Arduino IDE that reads sensor data and controls the relay module based on predefined thresholds.
- viii. Compile the code and upload it to the Arduino UNO.
- ix. Power the sensors and relay module using a suitable power supply.
- x. Position the sensors near the plant: Place the LRD sensor to measure light intensity, position the temperature and humidity sensor to monitor environmental conditions, and insert the soil moisture sensor into the soil to measure moisture level.
- xi. Monitor the plant health: Observe the sensor readings displayed on the Arduino IDE or an output display connected to the Arduino UNO.
- xii. Make adjustments if needed: Modify the code to adjust thresholds and actions based on your plant's requirements, such as changing lighting conditions or water pump activation levels.

CODING:

```
#include<DHT.h>
#define DHTPIN 7

// DHT11 Output Pin connection
#define DHTTYPE DHT11

// DHT Type is DHT11
const int sensor_pin = A1;

DHT dht (DHTPIN, DHTTYPE); // Initialize DHT sensor
```

```

void setup() {
  dht.begin();
  Serial.begin(9600); // To see data on serial monitor
}

void loop(){
  float moisture_percentage;
  int sensor_analog;
  float H = dht.readHumidity(); //Read Humidity
  float T = dht.readTemperature(); // Read temperature as Celsius// Check if any
reads failed and if exit
  int analogValue = analogRead(A0); //Reads analog pin

  if(isnan(H) || isnan(T)){
    Serial.println("Failed to read from DHT sensor!");
    return;
  } // Combine Humidity and Temperature into single string

  sensor_analog=analogRead(sensor_pin);
  moisture_percentage = ( 100 - ( sensor_analog/1023.00) * 100 ) );

  Serial.print("Humidity : ");
  Serial.print(String(H));
  Serial.print("\nTemperature : ");
  Serial.print(String(T));
  Serial.print("\nMoisture Percentage = ");
  Serial.print(moisture_percentage);

  Serial.print("\nLight Level");
  if (analogValue < 10) {
    Serial.println(" - Dark");
  } else if (analogValue < 200) {
    Serial.println(" - Dim");
  } else if (analogValue < 500) {
    Serial.println(" - Light");
  } else if (analogValue < 800) {
    Serial.println(" - Bright");
  }
}

```

```

    } else {
        Serial.println(" - Very bright");
    }

    Serial.print("\n");

    if(18<=T && T<=24)
        Serial.println("Optimal Temperature");
    else if(T<18)
        Serial.println("Increase Temperature");
    else
        Serial.println("Decrease Temperature");

    if(20<=moisture_percentage && moisture_percentage<=60)
        Serial.println("Irrigation Not Required");
    else if(moisture_percentage <20)
        Serial.println("Irrigation Required");
    else
        Serial.println("Oversaturated");

    if(20<=moisture_percentage && moisture_percentage<=60 && 18<=T && T<=24)
        Serial.println("OPTIMAL PLANT HEALTH CONDITION");

    Serial.print("\n\n\n");
    delay(4000); // Wait Four seconds between measurements
}

```

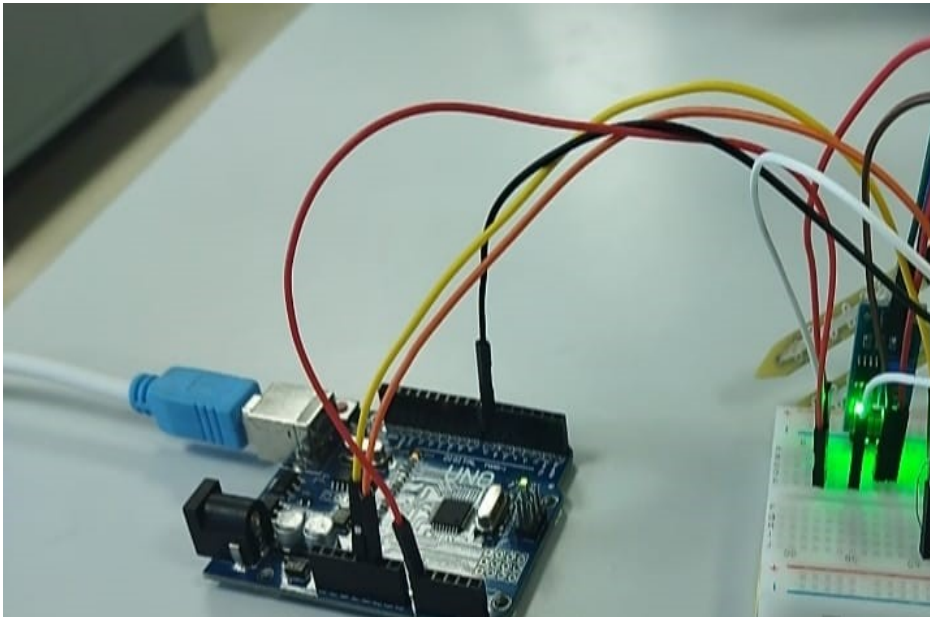
OUTPUT:

```
Humidity : 72.00  
Temperature : 32.50  
Moisture Percentage = 50.64  
Light Level - Light
```

```
Decrease Temperature  
Irrigation Not Required
```

```
Humidity : 72.00  
Temperature : 32.50  
Moisture Percentage = 52.30  
Light Level - Light
```

```
Decrease Temperature  
Irrigation Not Required
```



INFERENCE:

By analyzing the data collected from the LRD sensor, temperature and humidity sensor, and soil moisture sensor, we can draw the following inferences:

1. **Light Intensity:** The LRD sensor enables us to assess the light intensity around the plant. By monitoring the readings, we can determine if the plant is receiving adequate sunlight. Insufficient light intensity may indicate the need for additional lighting systems to promote photosynthesis and healthy growth.
2. **Environmental Conditions:** The temperature and humidity sensor provides information about the ambient conditions surrounding the plant. By analyzing the readings, we can infer whether the temperature and humidity levels are within the desired range for the specific plant species. Deviations from the optimal range may require adjustments to maintain favorable conditions for the plant's well-being.
3. **Soil Moisture Level:** The soil moisture sensor helps us determine the moisture content in the soil. By monitoring the readings, we can infer whether the plant requires watering or if the soil moisture level is sufficient. This information is crucial for avoiding under or overwatering, which can affect the plant's health and growth.
4. **Automated Control:** The integration of the Arduino UNO and relay module enables automated control of external devices. Based on the sensor data and predefined thresholds, the system can activate lighting systems, water pumps, or other devices to provide the necessary environmental conditions for the plant's optimal growth. This inference highlights the potential for automation to streamline plant care processes.

RESULT:

The Experiment was successfully executed and verified in lab.

Exp.No: 9

Unlocking a doorlock using facial recognition

Aim:

To create a basic face recognition program that can identify and authorize a person and unlock a door lock for the person to open the door

Apparatus decision:

Face recognition is a computationally intensive task. This cannot be done on a basic microcontroller such as Arduino. Therefore, for the board, we use the creditcard sized mini computer **Raspberry Pi 4** (8 GB RAM) which can deploy the machine learning algorithms required for the task. Raspberry pi is a full-fledged computer with a linux OS which can perform most basic operations of any modern computer. An alternative is to use your own personal computer with an Arduino board just to handle the activation of door lock.

A strong door lock that is electrically operated is also required to lock and unlock the door when required. Therefore we make use of a **12 V powered solenoid doorlock** for the application. Since Raspberry pi can only provide 5V output at max, a use of **12V relay** is required to power the door lock when signal is given from the raspberry pi board.

The input is obtained as a real-time video either from the raspberry pi camera or any camera that is connected to the board.

For the machine learning program coding, we will make use of the opencv library, which has ready to implement computer vision libraries that are required for the tasks of face detection and face recognition. Raspberry pi takes several hours of compilation to install this library to please plan accordingly

Working:

The raspberry pi board runs the face recognition program. Initial training with the face to be authorized is done using the help of an opencv classifier. Then when an authorized face is detected, the solenoid doorlock is unlocked, which locks itself after the duration.

The solenoid when powered by 12V acts as an electromagnet and sucks in the lock against the spring that pushes it out. After no face is detected, the electromagnet is disabled and the spring pushes out the lock once again.

The flow is as follows

- 1) The camera acts as the sensor to obtain real-time input for the raspberry pi board
- 2) Using a machine learning algorithm, the board checks if there are any authorized faces on the screen. If yes, a HIGH output is given to one of the pins that are connected with the 12 V relay IN.
- 3) This activates the relay, and the 12V solenoid is powered, which sucks in the lock.
- 4) After there is no face, the relay is deactivated, and this also disables the solenoid electromagnet, locking the doorlock again.

Procedure

To setup the face recognition program

- 1) Turn on the raspberry pi 4 board by connecting it to the appropriate power source using the C-type cable. Connect the micro hdmi cable to a monitor to be able to see the computer screen. Connect the keyboard and mouse devices as well.
- 2) It is a good practice to connect the camera module to the board prior to powering it up. Few of the boards will automatically restart when the camera is connected while it is already on. Handle the black bus holder with caution and do not damage the board while connecting the camera module.
- 3) Install the required libraries on your raspberry pi 4 board if they dont exist already.
 - I. Dlib - a toolkit for real-world Machine Learning and data analysis applications
 - II. Pillow - Python Imaging Library
 - III. Face_recognition - to coach and recognize faces

IV. Opencv (use pip install opencv-contrib-python)

4) From the git repository [GrandezXaferdian/Exp9-IOT \(github.com\)](https://github.com/GrandezXaferdian/Exp9-IOT) , install the zip file and then extract it

5) In the code, make necessary changes for the file paths

6) Take several photos of yourself in a well lit environment. Ideally please use around 10 images. Place these under a folder named in your name inside the 'Face_Images' folder

7) Run the 'Face_Trainer.py' program to train your model.

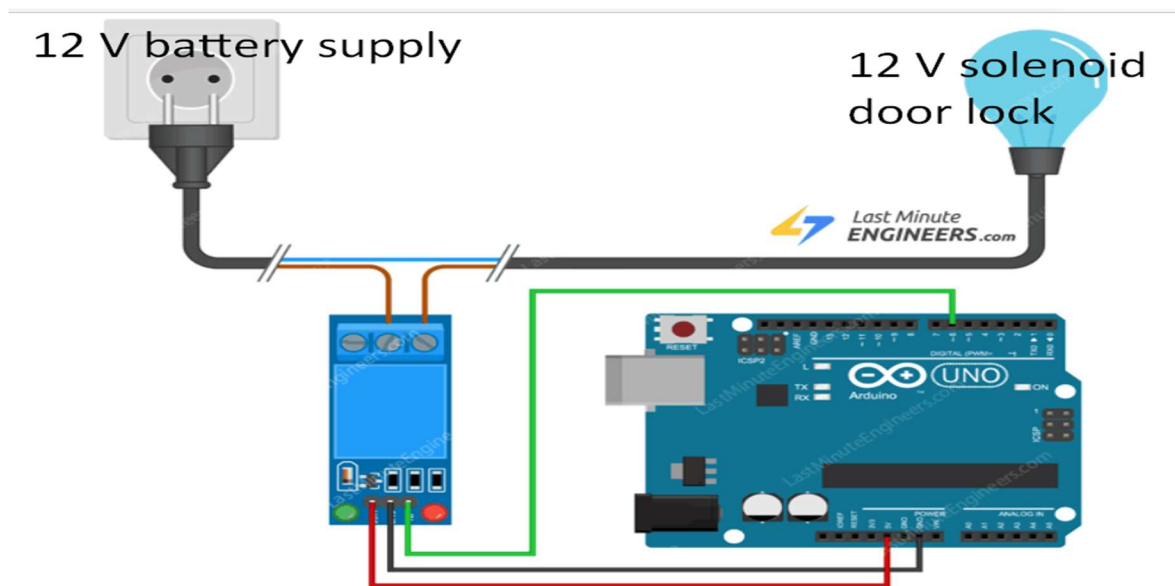
What happens in this program is each of your pictures are converted to greyscale, then to a numpy array and stored in face-trainer.xml as raw data. This is the data that will be used by the ML algorithm to compare and check your face's confidence level.

Note: If you are using your own PC instead of a raspberry pi board, the steps do not change much except for connecting your own camera/ using your inbuilt webcam instead of using raspberry pi webcam

To power the 12V solenoid door lock

1) We will use BCD 8 pin on the raspberry pi board to provide signal for the solenoid door lock since it is the closest to the ground pin (BCD 6). Since the solenoid is 12 V powered and normal 5V provided by raspberry pi cant be used to power it, we will be using a relay to give a 5V signal in order to complete a 12 V battery connected with solenoid door lock circuit.

Please refer to the connection diagram given below



The diagram is given for an arduino board, but we can also understand how we can connect with raspberry pi easily as well. Lets remind ourselves that in Raspberry pi, BCD pin 4 is 5V power, pin 6 is Ground and pin 8 is used for signal in our application. As such, the red wire is connected to pin 4, black to pin 6 and green to pin 8. (Note that BCD and GPIO are different pin configurations)



Alternate Function						Alternate Function
	3.3V PWR	1		2	5V PWR	
I2C1 SDA	GPIO 2	3		4	5V PWR	
I2C1 SCL	GPIO 3	5		6	GND	
	GPIO 4	7		8	UART0 TX	
	GND	9		10	UART0 RX	
	GPIO 17	11		12	GPIO 18	
	GPIO 27	13		14	GND	
	GPIO 22	15		16	GPIO 23	
	3.3V PWR	17		18	GPIO 24	
SPI0 MOSI	GPIO 10	19		20	GND	
SPI0 MISO	GPIO 9	21		22	GPIO 25	
SPI0 SCLK	GPIO 11	23		24	GPIO 8	SPI0 CS0
	GND	25		26	GPIO 7	SPI0 CS1
	Reserved	27		28	Reserved	
	GPIO 5	29		30	GND	
	GPIO 6	31		32	GPIO 12	
	GPIO 13	33		34	GND	
SPI1 MISO	GPIO 19	35		36	GPIO 16	SPI1 CS0
	GPIO 26	37		38	GPIO 20	SPI1 MOSI
	GND	39		40	GPIO 21	SPI1 SCLK

For more information on how relays work, please refer this link: [In-Depth: Interface One Channel Relay Module with Arduino \(lastminuteengineers.com\)](https://www.lastminuteengineers.com/in-depth-interface-one-channel-relay-module-with-arduino/)

2) Make the connections as guided. Check if the relay's red LED is glowing or not.

Note: If you are using the PC & Arduino way of doing this experiment, follow the connections as given in the diagram, and load this piece of code into your Arduino board. Connect the green wire to pin 13

```

void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    char data = Serial.read();
    if (data == '1') {
      digitalWrite(13, HIGH);
    } else if (data == '0') {
      digitalWrite(13, LOW);
    }
  }
}

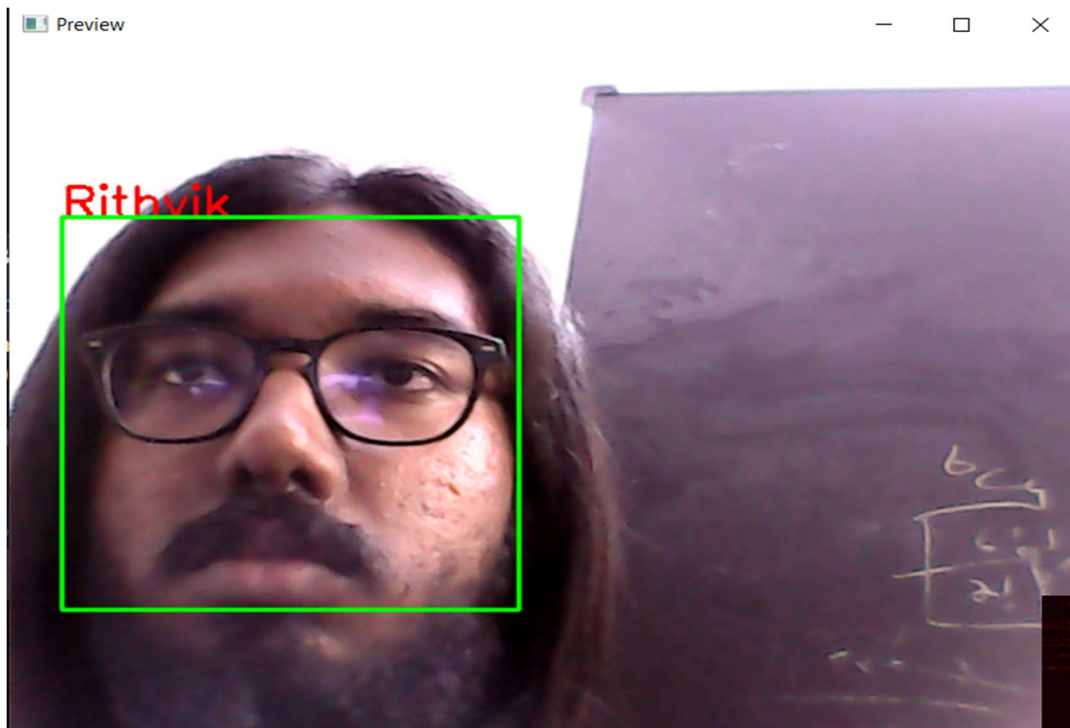
```

Note: Using pin 13 is easy for debugging purposes as there is an inbuilt LED to check if the signal is obtained or not

With the ML algorithm trained with our face and the door lock apparatus ready, we can now test the working of our application

Final step: Run the 'Face_Recog.py' program to start the application to detect your face.

Face Recognition application OUTPUT



Codes

Face_Recog.py

```
import cv2 # For Image processing
import numpy as np # For converting Images to Numerical array
import os # To handle directories
from PIL import Image # Pillow lib for handling images
import serial

# Create a serial connection object
ser = serial.Serial('COM3', 9600) # Replace 'COM3' with the appropriate serial port

labels = ["Person 1","Person 2"]

face_cascade = cv2.CascadeClassifier(r'C:\Users\rithv\OneDrive\Desktop\Face-
Detection-on-Raspberry-Pi-main\haarcascade_frontalface_default.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("face-trainer.yml")

cap = cv2.VideoCapture(0) # Get video feed from the Camera

while True:
    ret, img = cap.read() # Break video into frames
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # Convert Video frame to
    Greyscale
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5) #
    Recognize faces
    for (x, y, w, h) in faces:
        roi_gray = gray[y:y+h, x:x+w] # Convert Face to greyscale

        id_, conf = recognizer.predict(roi_gray) # Recognize the Face
        print(labels[id_],conf)
        if 100>=conf >= 85:
            font = cv2.FONT_HERSHEY_SIMPLEX # Font style for the name
            name = labels[id_] # Get the name from the List using ID number
```



```
cv2.putText(img, name, (x, y), font, 1, (0, 0, 255), 2)

# Send data to Arduino
ser.write(b'1') # Write '1' to turn on doorlock
#else:
    ser.write(b'0')

cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('Preview', img) # Display the Video
if cv2.waitKey(20) & 0xFF == ord('q'):
    break

# When everything is done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Face_Trainer.py

```
import cv2 # For Image processing
import numpy as np # For converting Images to Numerical array
import os # To handle directories
from PIL import Image # Pillow lib for handling images

face_cascade = cv2.CascadeClassifier(r'path_to_classifier.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()

Face_ID = -1
prev_person_name = ""
y_ID = []
x_train = []

Face_Images = os.path.join(os.getcwd(), "Face_Images") # Tell the program where we
have saved the face images
print(Face_Images)

for root, dirs, files in os.walk(r"path_to_face_image"): # go to the face image directory
    for file in files: # check every directory in it
        if True: # for image files ending with jpeg, jpg or png
            path = os.path.join(root, file)
            person_name = os.path.basename(root)
            print(path, person_name)

            if prev_person_name != person_name: # Check if the name of person has
changed
                Face_ID = Face_ID + 1 # If yes, increment the ID count
                prev_person_name = person_name

                Gery_Image = Image.open(path).convert("L") # convert the image to grayscale
using Pillow
                Crop_Image = Gery_Image.resize((800, 800), Image.ANTIALIAS) # Crop the
Grey Image to 550*550 (Make sure your face is in the center in all image)
```

```
Final_Image = np.array(Crop_Image, "uint8")
# print(Numpy_Image)
faces = face_cascade.detectMultiScale(Final_Image, scaleFactor=1.5,
minNeighbors=5) # Detect The face in all sample image
print(Face_ID, faces)

for (x, y, w, h) in faces:
    roi = Final_Image[y:y + h, x:x + w] # crop the Region of Interest (ROI)
    x_train.append(roi)
    y_ID.append(Face_ID)

recognizer.train(x_train, np.array(y_ID)) # Create a Matrix of Training data
recognizer.save("face-trainer.yml") # Save the matrix as YML file
```

Sources

Original Repository: [Face-Detection-on-Raspberry-Pi/Face Trainer.py at main · apurva-v8/Face-Detection-on-Raspberry-Pi · GitHub](#)

Blog Post (Original): [Real-time Face Detection using Raspberry Pi - Connections and Code - Robu.in | Indian Online Store | RC Hobby | Robotics](#)

NOTE: The code used in the blog post is outdated. Several changes have been made in the provided repository in the document which involve overcoming these shortcomings and also integrating the code specific to the required application

Result:

Hence the face recognition door unlocking system was implemented.

Exp.No: 10

SMART TRAFFIC LIGHT SYSTEM

AIM: intelligent traffic light control system using arduino.

COMPONENTS REQUIRED:

- LED lights (red, yellow, and green)
- Arduino board(Arduino Uno)
- jumper wires
- Breadboard(optional)
- (220 ohm) resistors
- Power source (such as a 9V battery or a USB cable)

Theory:

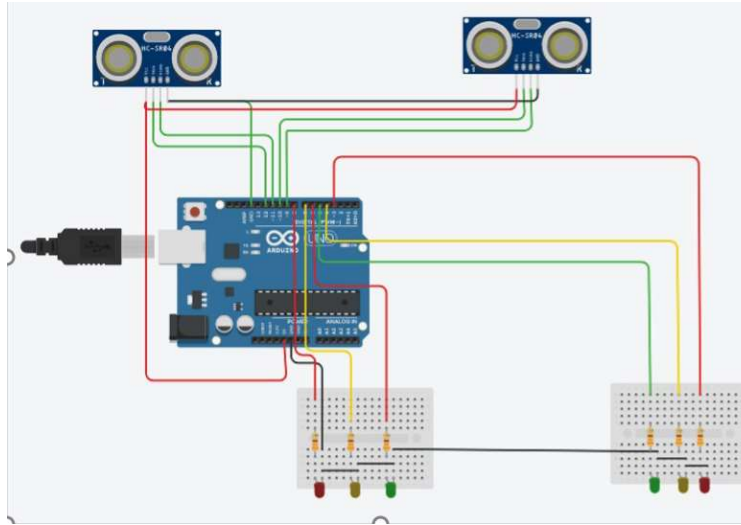
Traffic congestion is a persistent problem in urban areas, leading to numerous negative consequences such as increased pollution, longer commute times, and decreased productivity. Traditional traffic light systems with fixed timers often fail to adapt to real-time traffic conditions, exacerbating congestion issues. However, advancements in sensor technology, particularly ultrasonic sensors, have paved the way for the development of smart traffic light systems that can dynamically adjust signal timings based on actual traffic demand.

The objective of this experiment is to investigate the efficacy of a smart traffic light system using ultrasonic sensors in optimizing traffic flow and reducing congestion at intersections. By accurately detecting the presence of vehicles through ultrasonic sensors and employing adaptive algorithms for signal timing adjustments, the system aims to improve traffic efficiency, reduce waiting times, and enhance overall road safety.

PROCEDURE:

1. Use a USB cable or a 9V battery to connect the Arduino board to your computer or power source.

2. Link the Arduino board to the LED lights. Each LED's cathode (short leg) should be connected to a 220-ohm resistor, and the resistor's other end should be connected to the Arduino's ground (GND) pin.
3. Each LED's anode (long leg) should be connected to one of the Arduino's digital pins, such as pin 2 for red, pin 3 for yellow, or pin 4 for green.
4. Launch your computer's Arduino IDE (Integrated Development Environment).
5. Create the code necessary to operate the traffic light system.
6. Select the correct Arduino board and port in the Arduino IDE.
7. Click on the "Upload" button to upload the code to the Arduino board.



8. Wait for the upload process to complete.
9. Test the smart traffic light system.
10. Once the code is uploaded successfully, Observe the behavior of the LED lights based on the code

you've written.

CIRCUIT DIAGRAM:

CODE:

```
int red1=8;
int green1=6;
int yellow1=7;
int red2=3;
int green2=5;
int yellow2=4;

const int pingPin = 10; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 9; // Echo Pin of Ultrasonic Sensor
const int pingPin2 = 12; // Trigger Pin of Ultrasonic Sensor
const int echoPin2 = 11; // Echo Pin of Ultrasonic Sensor

void setup() {
    // put your setup code here, to run once:
    pinMode(red1,OUTPUT);
    pinMode(red2,OUTPUT);
    pinMode(yellow1,OUTPUT);
    pinMode(yellow2,OUTPUT);
    pinMode(green1,OUTPUT);
    pinMode(green2,OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    int distance1,distance2;
    distance1=calculatedistance(pingPin , echoPin);
    distance2=calculatedistance(pingPin2 , echoPin2);
    if(distance1>=distance2){
        digitalWrite(red1,LOW);
```

```
digitalWrite(green2,LOW);
digitalWrite(red2,LOW);
digitalWrite(green1,LOW);
digitalWrite(yellow1,HIGH);
digitalWrite(yellow2,HIGH);
delay(200);
    while(distance1>distance2){
        distance1=calculatedistance(pingPin , echoPin);
        distance2=calculatedistance(pingPin2 , echoPin2);
digitalWrite(red1,HIGH);
digitalWrite(green2,HIGH);
digitalWrite(red2,LOW);
digitalWrite(green1,LOW);
digitalWrite(yellow1,LOW);
digitalWrite(yellow2,LOW);
    }
}
if(distance2>distance1){
    digitalWrite(red1,LOW);
digitalWrite(green2,LOW);
digitalWrite(red2,LOW);
digitalWrite(green1,LOW);
digitalWrite(yellow1,HIGH);
digitalWrite(yellow2,HIGH);
delay(200);
    while(distance2>distance1){
        distance1=calculatedistance(pingPin , echoPin);
        distance2=calculatedistance(pingPin2 , echoPin2);
digitalWrite(red1,LOW);
```

```

digitalWrite(green2,LOW);
digitalWrite(red2,HIGH);
digitalWrite(green1,HIGH);
digitalWrite(yellow1,LOW);
digitalWrite(yellow2,LOW);
}
}
}
long microsecondsToCentimeters(long microseconds)
{
    return microseconds / 29 / 2;
}
int calculatedistance(int pingPin , int echoPin){
    long duration, inches, cm,meter;
    pinMode(pingPin, OUTPUT);
    digitalWrite(pingPin, LOW);
    delayMicroseconds(2);
    digitalWrite(pingPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(pingPin, LOW);

    pinMode(echoPin, INPUT);
    duration = pulseIn(echoPin, HIGH);

    cm = microsecondsToCentimeters(duration);
    meter = cm/100;
    return meter;
}

```

OUTPUT:

[illegible]

INFERENCE:

It is anticipated that the smart traffic light system utilizing ultrasonic sensors will outperform traditional fixed-timer systems in terms of traffic efficiency. The dynamic adjustment of signal timings based on real-time traffic data will lead to reduced congestion, shorter travel times, and improved overall traffic flow. The experiment aims to quantify these improvements and provide evidence for the feasibility and effectiveness of the proposed system.

RESULT:

The integration of ultrasonic sensors into a smart traffic light system has the potential to revolutionize traffic management by enabling real-time adjustments to signal timings based on actual traffic conditions. This experiment seeks to demonstrate the benefits of such a system in terms of reducing congestion and improving traffic efficiency. The findings from this study can inform future implementations of smart traffic light systems, contributing to more sustainable and intelligent transportation systems.