CONTENT

EX.No : 1 **DATE :**	**LED LIGHT ON/OFF USING PUSH BUTTON**

AIM:

To create an IOT program to turn on/off LED Light (5V).

ALGORITHM:

Step 1: Arduino UNO board, Breadboard, LED Light, Resistor, Push Button, Five Jumper wires and fix it.

Step 2: Set up the Hardware by connecting the LED to the microcontroler digital Pin.

Step 3: Connect the LED to the Arduino board (Ground and 13) using positive and negative.

Step 4: In your computer System open Arduino 2.3.2 version & Type a code.

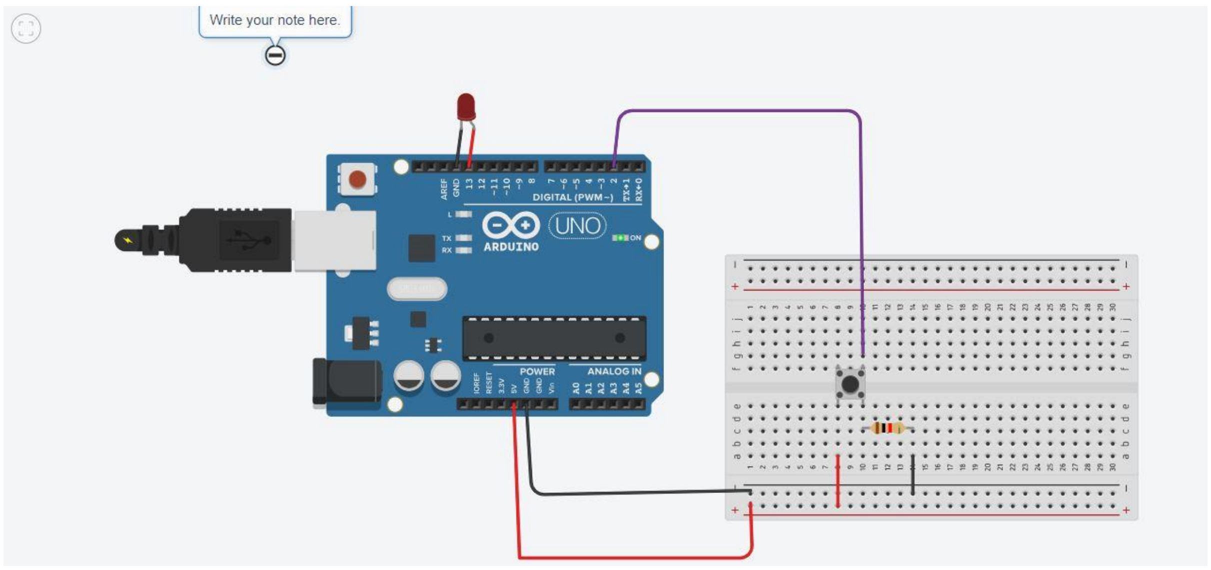
Step 5: Next connect the USB port in Arduino board, After click right symbol and Run the code.

PROGRAM:

```
void setup()
{

    pinMode(2,INPUT);
    pinMode(13,OUTPUT);
}

void loop()
{
   if(digitalRead(2) == 1)
  {
      digitalWrite(13,HIGH);
  }
    else
   {
      digitalWrite(13,LOW);
   }
 }
```

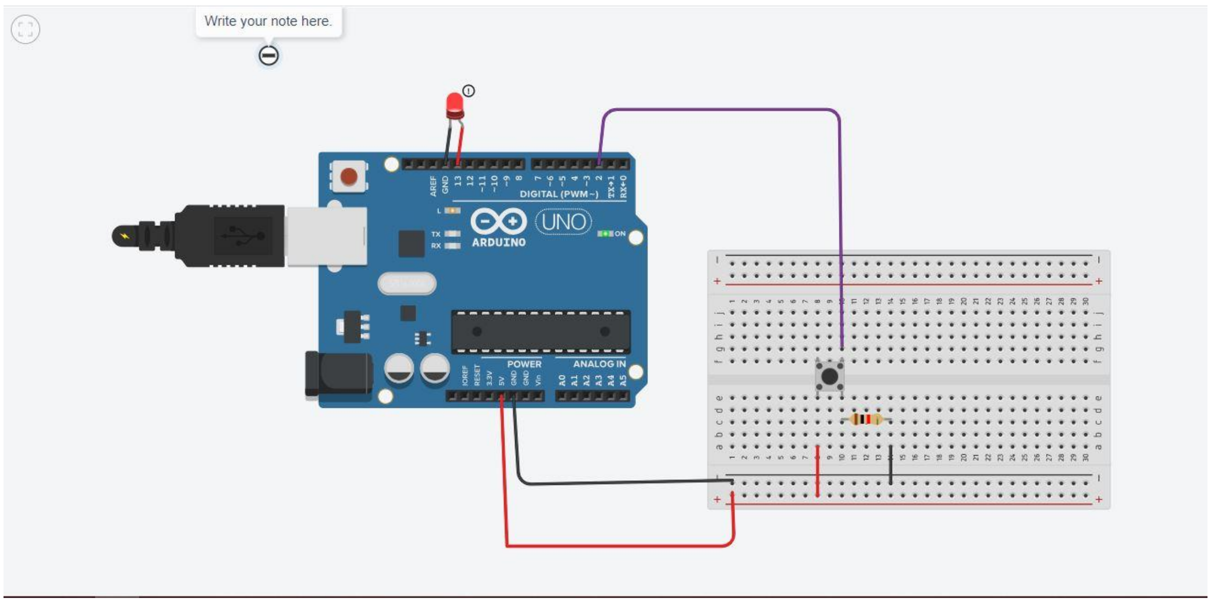
DIAGRAM:

Before didn't click Push Button



OUTPUT:

After Click and Hold Push Button



RESULT:

Thus the above program is verified and executed successfully.

EX.No : 2 **DATE :**	**TO DEVALOP AN IOT PROGRAM USING IR SENSOR (Smart Garbage Monitoring)**

AIM:

To develop an IOT program using IR sensor (smart garbage monitoring system).

ALGORITHM:

Step 1: Components: Arduino UNO, IR sensor module, Servo motor, Jumper wires and Garbage bin.

Step 2**:** connect IR module to Arduino UNO , connect servo motor to Arduino UNO, mount IR sensor inside the garbage bin, facin downwards.

Step 3: Wiring: connect Vcc pin of IR sensor to 5v on arduino. Connect GND pin of IR sensor to GND pin of arduino. Connect OUTPIN of IR to any digital pin on arduino [eg. Pin d].

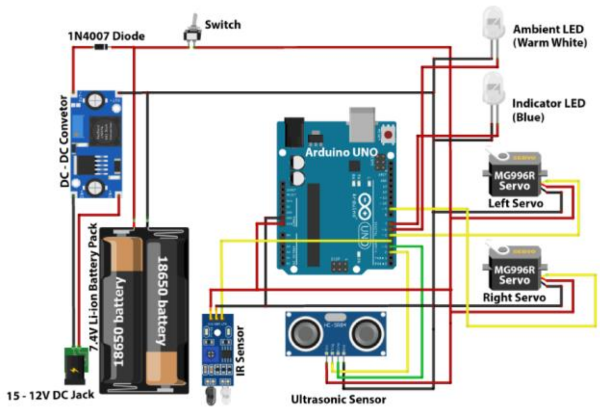
Step 4: Install servo library for controlling servo motor, update the source code.

Step 5: Place on object (garbage) inside the bin. The servo motor open rid when object blocks IR sensor. Adjust threshold value and output received successfully

PROGRAM:

```
#include <servo.h>

Servo my servo;

int irsensor pin = 2;

int threshold = 100;

void setup() {

        myservo.attach(9);

        pinMode (irsensor,INPUT);

        serial.begin (9600);

}

void loop() {

         int sensor value = digital Read (irsensor pin);

         serial.println (sensor value);

if  (sensor value <threshold) {

          openlid();

          delay (5000);

           close lid();

}

}

void openLid(){

           myservo.write (90);

}

void close Lid() {

            myservo.write (0);

}
```

DIAGRAM:



OUTPUT:

RESULT:

Thus the above program is verified and executed successfully.

EX.No : 3 **DATE :**	**TO DEVELOP AN IOT PROGRAM USING HUINITY AND TEMPERATURE MONITORING**

AIM:

To develop an Humidity and temperature monitoring [forest fire detecting, weather monitoring.

ALGORITHM:

Step 1: We have to connect the Equipments: Arduino UNO, DHT11 sensor, 16X2 LCD display and I2C module with female to male jumper wires and also male to male jumper wires.

Step 2: Install humidity and temperature sensors at each monitoring station. These sensors should be placed at a suitable height above ground level and shielded from direct sunlight and moisture to ensure accurate readings.

Step 3: And check the Boards in Arduino UNO & port with COM3. Over the library we have to install the **LiquidCrystal I2C** and wire master.

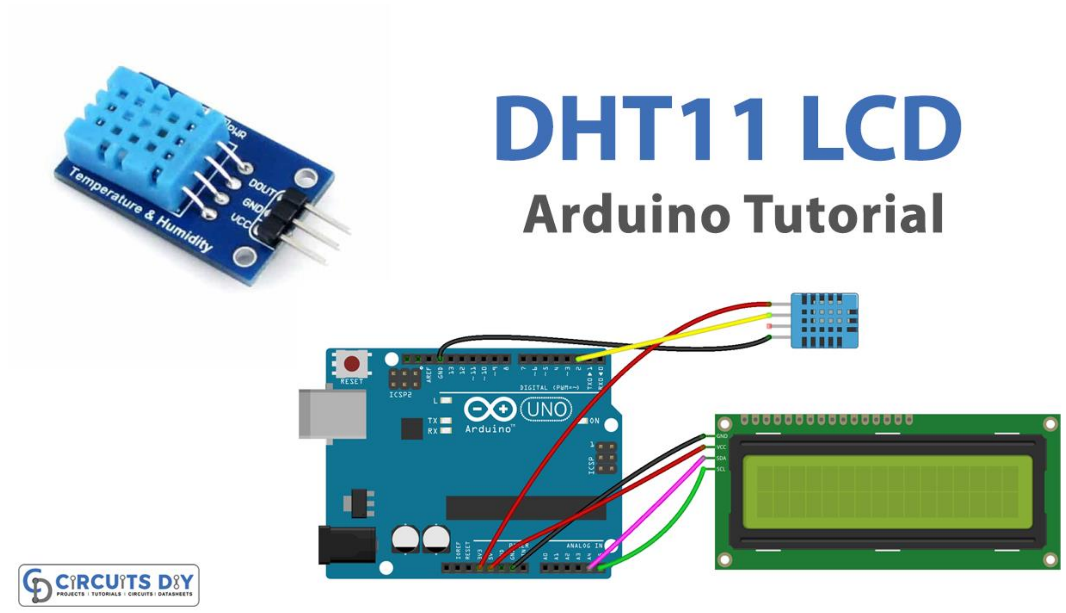
Step 4**:** After these all Data Transmission: Set up a system for transmitting data from the sensors to a central monitoring station. This can be done using wired or wireless communication methods, depending on the availability of infrastructure in the area.

Step 5: Finally verify the program and upload it.

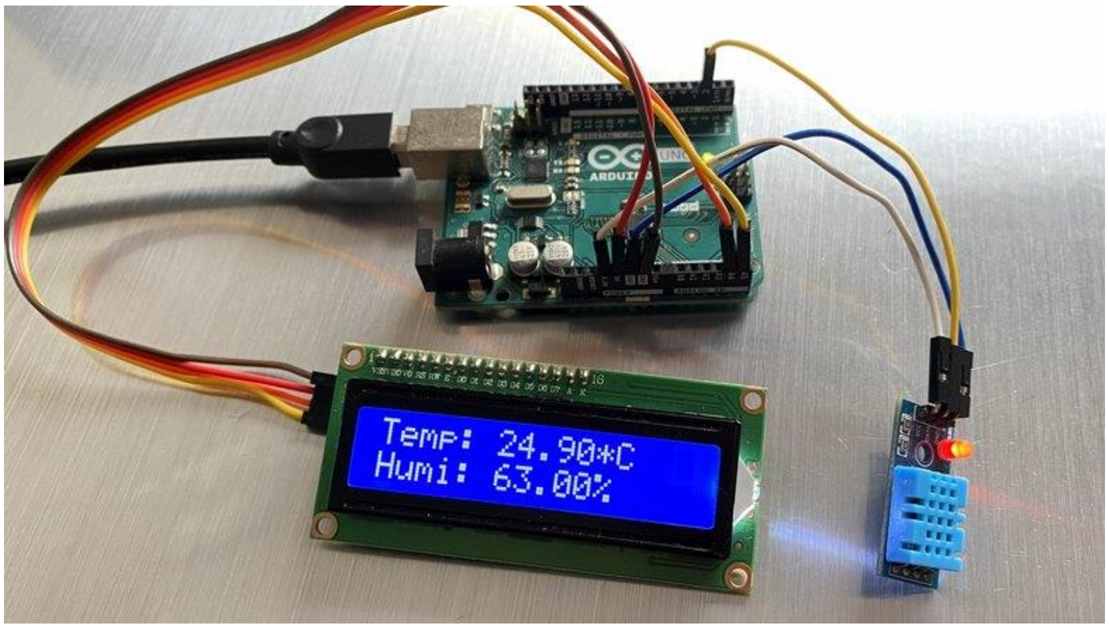
PROGRAM:

```
#include <LiquidCrystal.h>
LiquidCrystal C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line
display
byte degree_symbol[8] =
              {
                0b00111,
                0b00101,
                0b00111,
                0b00000,
                0b00000,
                0b00000,
                0b00000,
                0b00000
              };q
int gate=11;
volatile unsigned long duration=0;
unsigned char i[5];
unsigned int j[40];
unsigned char value=0;
unsigned answer=0;
int z=0;
int b=1;
void setup()
{
 lcd.init();                       // initialize the lcd
 lcd.init();
 lcd.backlight();
 lcd.print("Temp = ");
 lcd.setCursor(0,1);
 lcd.print("Humidity = ");
 lcd.createChar(1, degree_symbol);
 lcd.setCursor(9,0);
 lcd.write(1);
 lcd.print("C");
 lcd.setCursor(13,1);
 lcd.print("%");
}

void loop()
{

 delay(1000);
 while(1)
 {
  delay(1000);
  pinMode(gate,OUTPUT);
  digitalWrite(gate,LOW);
  delay(20);
  digitalWrite(gate,HIGH);
  pinMode(gate,INPUT_PULLUP);//by default it will become high due to internal pull up
 // delayMicroseconds(40);


  duration=pulseIn(gate,LOW);
  if(duration <= 84 && duration >= 72)
  {
      while(1)
      {
        duration=pulseIn(gate, HIGH);

        if(duration <= 26 && duration >= 20){
        value=0;}

        else if(duration <= 74 && duration >= 65){
        value=1;}

        else if(z==40){
        break;}

        i[z/8]|=value<<(7- (z%8));
        j[z]=value;
        z++;
      }
    }
answer=i[0]+i[1]+i[2]+i[3];

if(answer==i[4] && answer!=0)
{
lcd.setCursor(7,0);
lcd.print(i[2]);
lcd.setCursor(11,1);
lcd.print(i[0]);
}

z=0;
i[0]=i[1]=i[2]=i[3]=i[4]=0;
 }
}
```

DIAGRAM:



OUTPUT:



RESULT:

Thus the above program is verified and executed successfully.

EX.No : 4 **DATE :**	**TO DEVELOP AN IOT WEB SERVER PROGRAM FOR LOCAL HOSTING**

AIM:

To Develop an IOT WEB program for Local Hosting.

ALGORITHM:

Step 1: Take the required Components.

- ARDUINO
- WFI Module
- JUMPER WIRE (M –M , M-F)
- RESISTOR
- USB CABLE
- LED light.

Step 2: Connect the LED with Resistor in a Pin of 12 And GND.

Step 3: Get link the TX in module with Pin of D2, link the RX in module with Pin of -3.

Step 4: Join Power line with 3.3V with Power module, Connect GND and a power module through a resistor.

Step 5: Connect the Arduino to Your System using USB cable.

Step 6: Insert and Upload your Arduino Code and Run the Program.

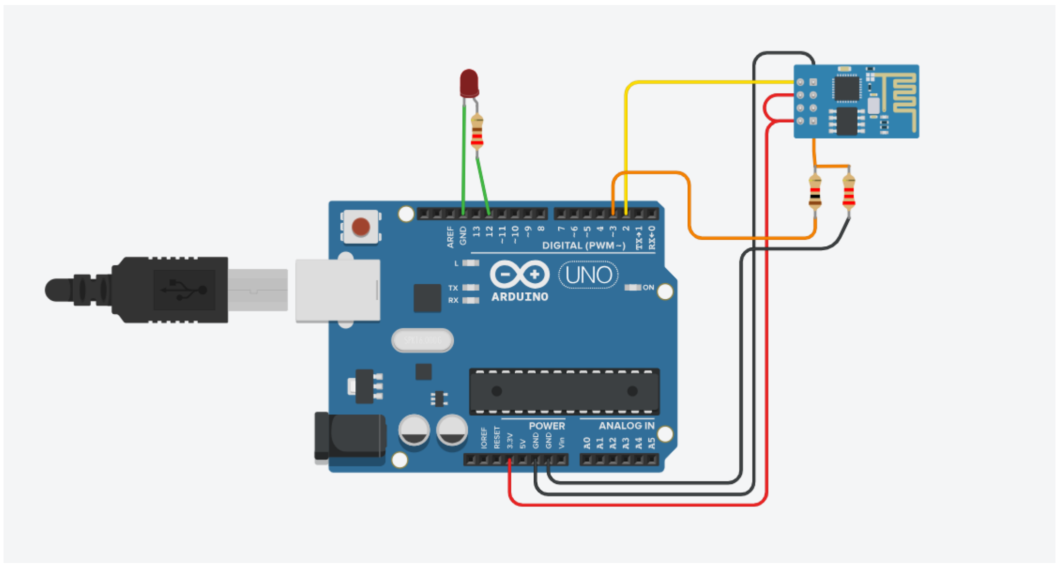
PROGRAM:

```
   #include <SoftwareSerial.h>
   #define DEBUG true
   SoftwareSerial esp8266(2,3);
    int LED=12; //LED PIN
    int itsONled[] = {0,0}; // para encender el led

   void setup()
   {
     pinMode(LED, OUTPUT);
     Serial.begin(9600);
     esp8266.begin(115200); // depende del modulo esp wifi
     sendData("AT+RST\r\n",2000,DEBUG);
     //sendData("AT+CWMODE=1\r\n",1000,DEBUG); // Configura la salid como access
point mode
     sendData("AT+CWMODE=1\r\n",1000,DEBUG); //
     sendData("AT+CWJAP=\"yourssid\",\"pass\"\r\n", 6000, DEBUG);
     sendData("AT+CIFSR\r\n",2000,DEBUG); // para declarar la ip
     sendData("AT+CIPMUX=1\r\n",1000,DEBUG); // configure for multiple connections
     sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // abre puerto 80 de internet
   }

void loop()
{

  if(esp8266.available()) // chequear el envio del modulo ESP
  {
   if(esp8266.find("+IPD,"))
   {

    int connectionId = esp8266.read()-48;
```

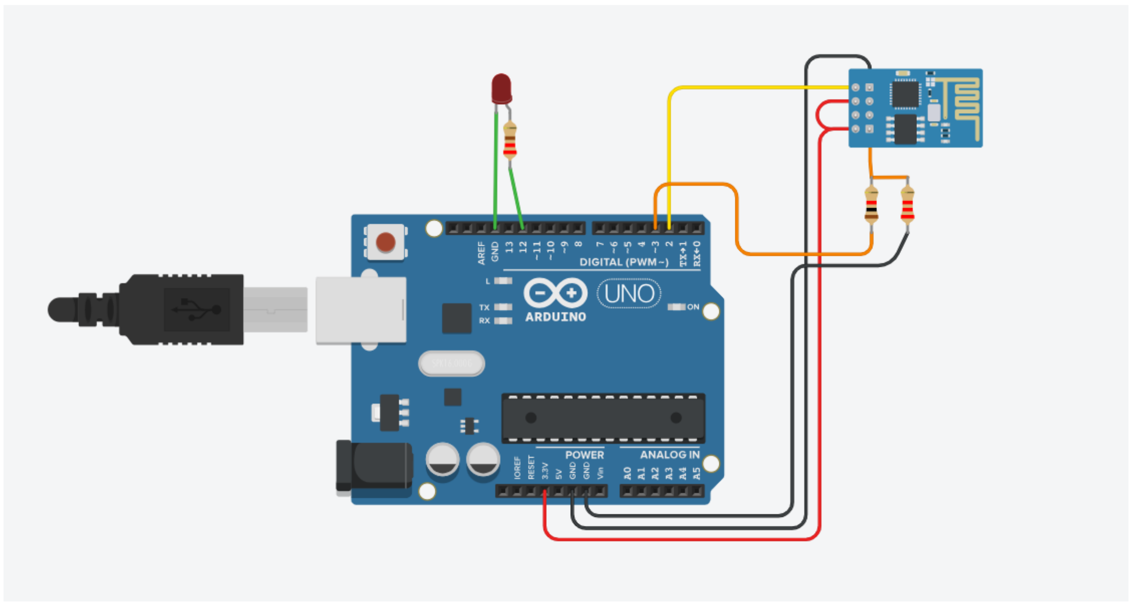
```
//lectura del string enviado al cliente
String msg;
esp8266.find("?");
delay(100);
msg = esp8266.readStringUntil(' ');
String command1 = msg.substring(0);
// CODIGO HTML
String webpage = "<html><head><title>Atencion 24/7</title>";
webpage += "<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0\"><style>.button {background-color: orange;border: none;color: white;padding:
15px 32px;text-align: center;display: inline-block;font-size: 16px;} .centre {text-align:
center;}</style>";
webpage += "</head><body class=\"centre\"><h1 class=\"centre\">Bienvenido</h1>";
//COMNADO PARA ENCENDER Y APAGAR LED
           if (command1 == "T"){
             if (itsONled[1] == 1)
             {
               digitalWrite(LED, LOW);
               itsONled[1] = 0;
               webpage += "<p>apagado</p>";
             }
             else
             {
               digitalWrite(LED, HIGH);
               itsONled[1] = 1;
               webpage += "<p>encendido</p>";
             }
          }
webpage += "<a class=\"button\" href=\"?T\">TAP</a></body></html>";
String cipSend = "AT+CIPSEND=";
cipSend += connectionId;
```

```
     cipSend += ",";
     cipSend +=webpage.length();
     cipSend +="\r\n";
     sendData(cipSend,500,DEBUG);
     sendData(webpage,500,DEBUG);
     //BELOW THIS LINE CLOSE THE CONNECTION
     String closeCommand = "AT+CIPCLOSE=";
     closeCommand+=connectionId; // append connection id
     closeCommand+="\r\n";
     sendData(closeCommand,500,DEBUG);
    }
  }

}

//envio de datos por el modulo ESP
void sendData(String command, const int timeout, boolean debug)
{
    esp8266.print(command); // send the read character to the esp8266
    long int time = millis();

    while( (time+timeout) > millis())
    {
      while(esp8266.available())
      {
        //Para que se vea por el monitor serial
        Serial.write(esp8266.read());
      }
    }
}
```

DIAGRAM:



OUTPUT:



RESULT:

Thus the above program is verified and executed successfully.

EX.No : 5 **DATE :**	**TO DEVELOP IOT PROGRAM USING SOIL MOISTURE SENSOR**

AIM:

To develop an IOT program using soil moisture sensor.

ALGORITHM:

Step 1: Materials required: Arduino board(Uno), soil moisture sensor module, jumper wires, USB cable

Step 2: Wiring: Connect the soil moisture sensor to the Arduino board:

Soil moisture sensor VCC -> Arduino 5V

- Soil moisture sensor GND -> Arduino GND
- Soil moisture sensor OUT -> Arduino analog pin (e.g., A0)
- Ensure the connections are secure and well-insulated to prevent short circuits.

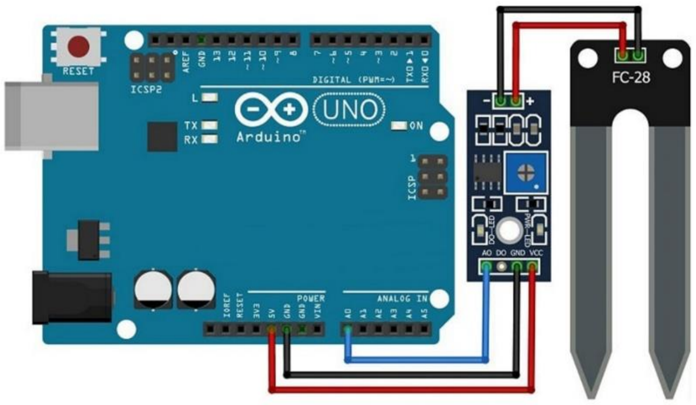
Step 4: Upload the code: Connect your Arduino board to your computer using a USB cable. Select the correct board and port from the "Tools" menu in the Arduino IDE. Click the "Upload" button to upload the code to your Arduino board.

Step 5: Observation: Observe the soil moisture readings displayed on the serial monitor. You will see values ranging from 0 (dry) to 100 (wet).

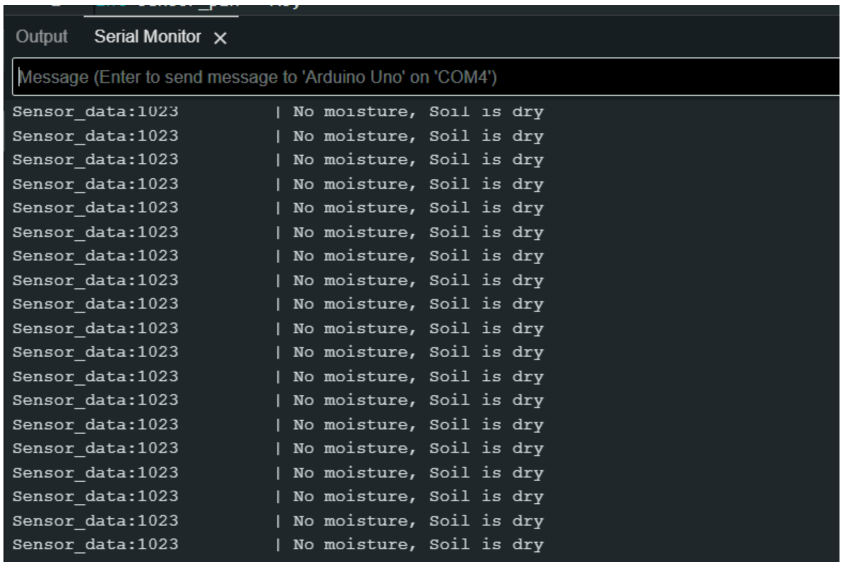
PROGRAM:

```
const int sensor_pin = A1;      /* Soil moisture sensor O/P pin */


void setup() {
  Serial.begin(9600);   /* Define baud rate for serial communication */
}


void loop() {
  float moisture_percentage;
  int sensor_analog;
  sensor_analog = analogRead(sensor_pin);
  moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
  Serial.print("Moisture Percentage = ");
  Serial.print(moisture_percentage);
  Serial.print("%\n\n");
  delay(1000);
}
```

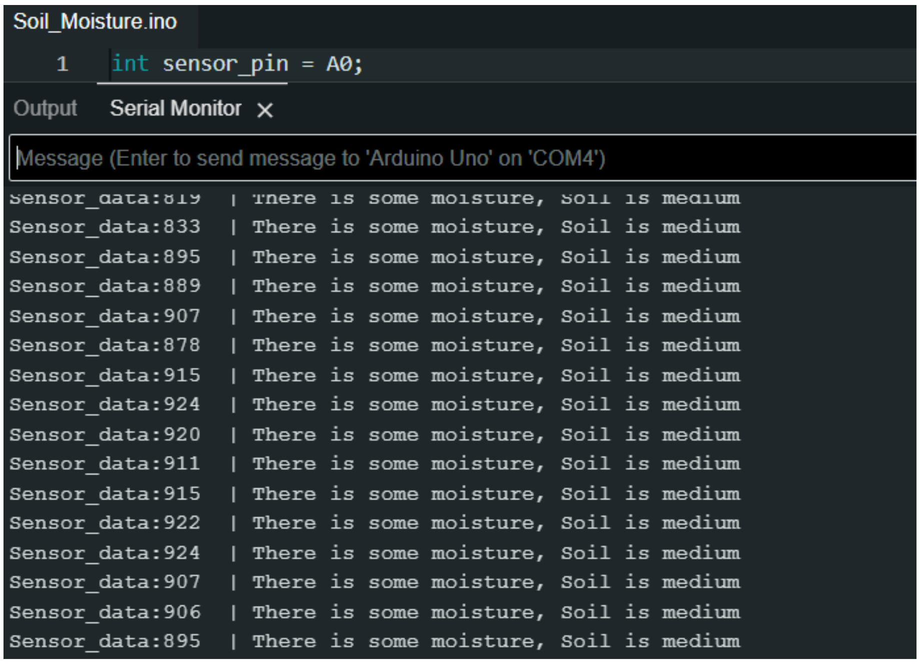
DIAGRAM:



OUTPUT:



```
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
Sensor_data:1023        | No moisture, Soil is dry
```



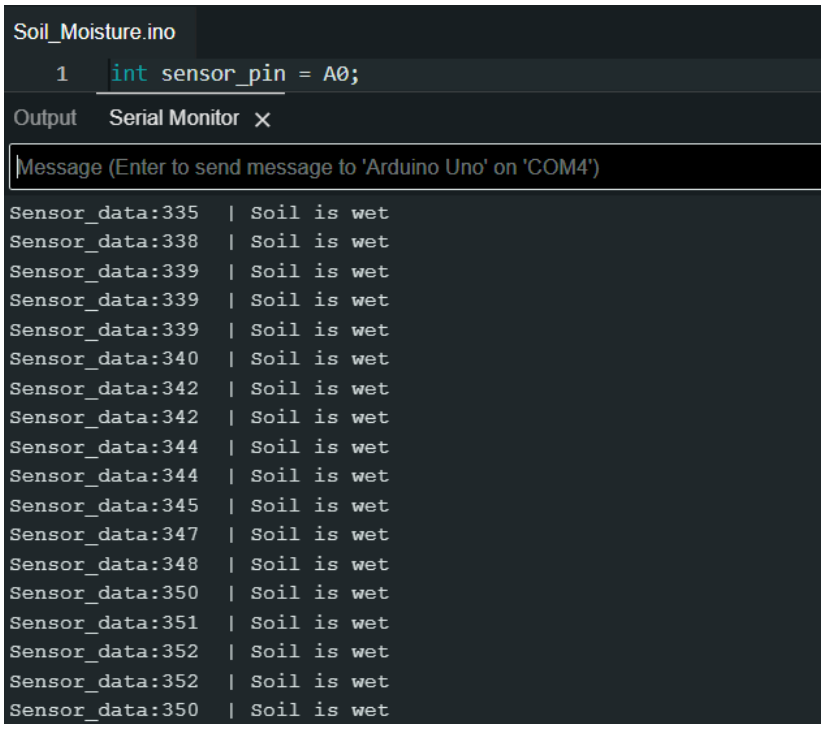
Soil_Moisture.ino

```
int sensor_pin = A0;
```

Output   Serial Monitor ×

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

```
Sensor_data:819  | There is some moisture, Soil is medium
Sensor_data:833  | There is some moisture, Soil is medium
Sensor_data:895  | There is some moisture, Soil is medium
Sensor_data:889  | There is some moisture, Soil is medium
Sensor_data:907  | There is some moisture, Soil is medium
Sensor_data:878  | There is some moisture, Soil is medium
Sensor_data:915  | There is some moisture, Soil is medium
Sensor_data:924  | There is some moisture, Soil is medium
Sensor_data:920  | There is some moisture, Soil is medium
Sensor_data:911  | There is some moisture, Soil is medium
Sensor_data:915  | There is some moisture, Soil is medium
Sensor_data:922  | There is some moisture, Soil is medium
Sensor_data:924  | There is some moisture, Soil is medium
Sensor_data:907  | There is some moisture, Soil is medium
Sensor_data:906  | There is some moisture, Soil is medium
Sensor_data:895  | There is some moisture, Soil is medium
```



Soil_Moisture.ino

```
int sensor_pin = A0;
```

Output   Serial Monitor ×

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

```
Sensor_data:335  | Soil is wet
Sensor_data:338  | Soil is wet
Sensor_data:339  | Soil is wet
Sensor_data:339  | Soil is wet
Sensor_data:339  | Soil is wet
Sensor_data:340  | Soil is wet
Sensor_data:342  | Soil is wet
Sensor_data:342  | Soil is wet
Sensor_data:344  | Soil is wet
Sensor_data:344  | Soil is wet
Sensor_data:345  | Soil is wet
Sensor_data:347  | Soil is wet
Sensor_data:348  | Soil is wet
Sensor_data:350  | Soil is wet
Sensor_data:351  | Soil is wet
Sensor_data:352  | Soil is wet
Sensor_data:352  | Soil is wet
Sensor_data:350  | Soil is wet
```

RESULT:

Thus the above program is verified and executed successfully.

EX.No : 6 **DATE :**	**TO DEVELOP AN IOT PROGRAM USING ULTRASONIC SENSOR (DISTANCE MEASURMENT)**

AIM:

To Develop an iot program using ultrasonic sensor (distance measurment)

ALGORITHM:

Step 1: Components need: Arduino board(UNO), ultrasonic sensor(ie: HC-SRO4), jumper wires, bread board, LCD display.

Step 2: Connect VCC pin of ultrasonic sensor to the 5V pin on Arduino and connect GND pin of sensor to GND (arduino).

Step 3: Then connect TRIG pin of sensor to didgital pin 9 on Arduino and connect ECHO pin of sensor to digital pin 10 on the Arduino.

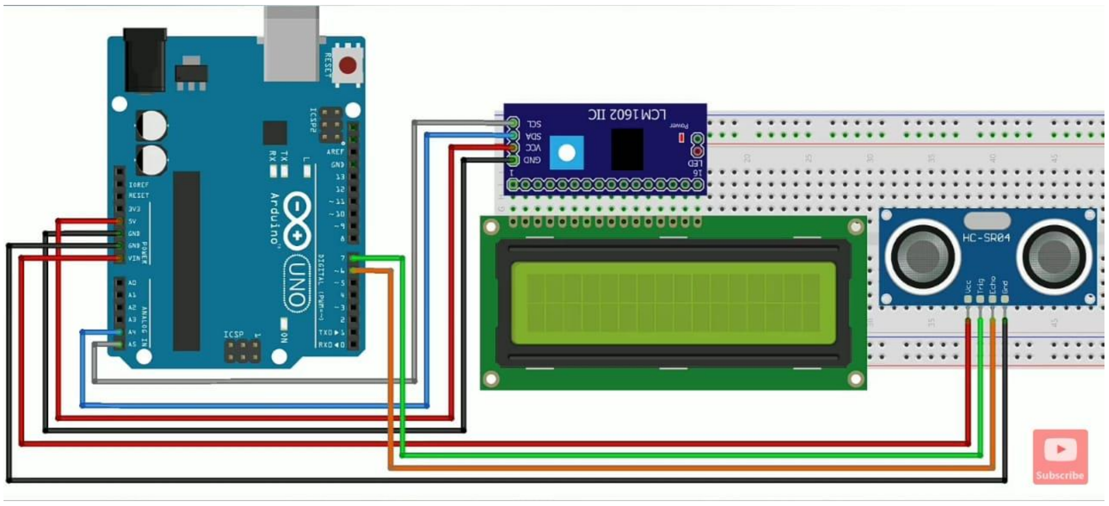
Step 4: **:** Open new Sketch in Araduino IDE write a code and connect Arduino board to your computer using USB cable.

Step 5: **:** Open new Sketch in Araduino IDE write a code and connect Arduino board to your computer using USB cable.

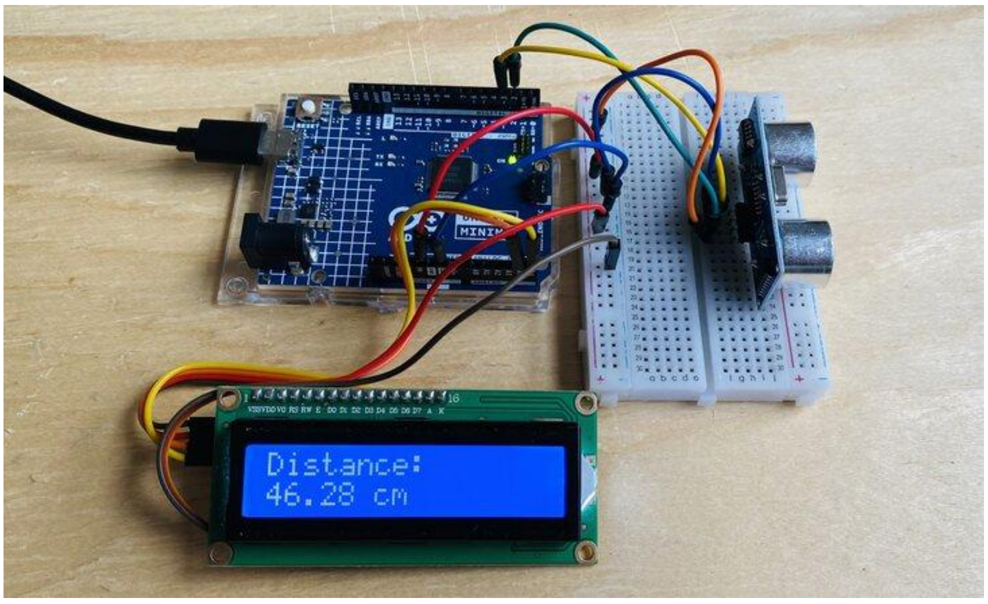
PROGRAM:

```
#include <Wire.h> // Library for I2C communication
#include <LiquidCrystal_I2C.h> // Library for LCD
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x3F, 16, 2); // Change to (0x27,16,2) for 16x2
LCD.my address is (0x3F)
int trigPin = A0;
int echoPin = A1;
long distance;
long distanceInch;
long duration;

void setup(){
 lcd.init();
 lcd.backlight();
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);

 lcd.setCursor(0,0);
 lcd.print("   WELCOME TO");
 lcd.setCursor(0,1);
 lcd.print("  MKINVENTIONS");
 delay(2000);
}

void loop() {
 ULTRASONIC();
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("DISTANCE CM:");
 lcd.print(distance);
 lcd.setCursor(0,1);
 lcd.print("DISTANCE INCH:");
 lcd.print(distanceInch);

// lcd.setCursor(0,1);
// lcd.print("DISTANCE MM:");
// lcd.print(distanceInch);
}

void ULTRASONIC(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration*0.034/2;
  distanceInch = duration*0.0133/2;
}
```

DIAGRAM:



OUTPUT:



RESULT:

Thus the above program is verified and executed successfully.

EX.No : 7 DATE :	TO DEVELOP AN REAL-TIME IOT PROGRAM USING REALY MODULE (SMART HOME AUTOMATION WITH 230V)

AIM:

To write a program to develop an real time iot program using relaymodule (smart home automation with 230v).

ALGORITHM:

Step 1: Start the Program using Arduino Board

- ➢ Arduino Board
- ➢ Jumper Wire
- ➢ Relay
- ➢ 230 V Bulb
- ➢ Volder

Step 2: Take the Jumper wire.

Step 3: Connect relay with arduino board and Insert Volder.

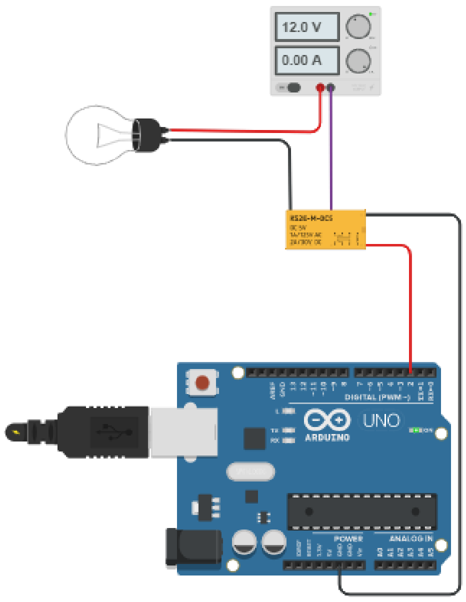
Step 4: Connect arduino board with 230 volt bulb , Type the program

Step 5: Run the program

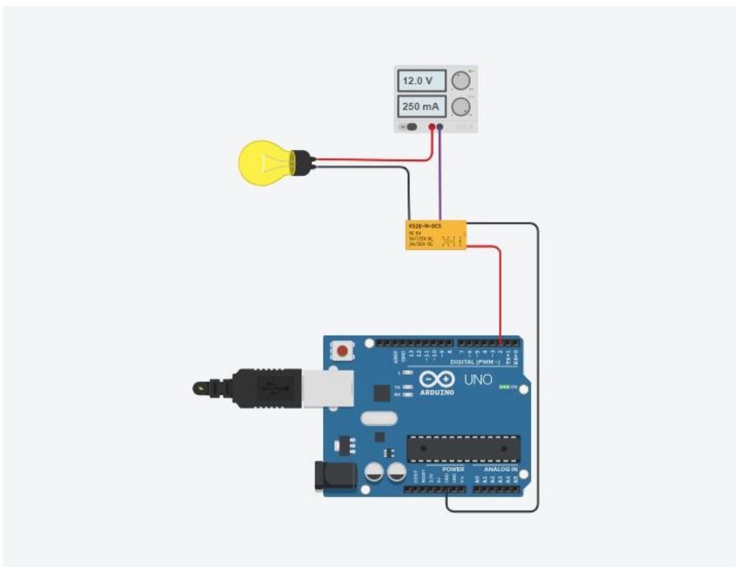
PROGRAM:

```
    Int relay=8;
    Void setup() {
   //put your setup code here, to run once:
   Pinmode (relay, OUTPUT);
   }

  Void loop(){
 //put your main code here to run repeatedly:
 Digitalwrite(relay,HIGHT);
Delay(1000);
Digitalwrite(relay,LOW);
Delay(1000);
}
```

DIAGRAM:



OUTPUT:



RESULT:

Thus the above program is verified and executed successfully.

EX.No : 8 **DATE :**	**TO DEVELOP AN IOT PROGRAM FOR FIRE DETECTION (HOME, INDUSTRY)**

AIM:

To Develop an IOT program For Fire Detection.

ALGORITHM:

Step 1: APPARATUS NEEDED

- ARDUINO
- FIRE SENSOR MODULE
- JUMPER WIRE (M –M , M-F)
- BUZZER
- BREADBOARD
- USB CABLE

Step 2 : Take the required Components Place the buzzer which connects E and F in the Breadboard.

Step 3: Get link the Arduino Pin ~9 with the Buzzer's Anode Get link to 5V to the Cathode of Breadboard

Step 4: Connect the Sensor Wires to,

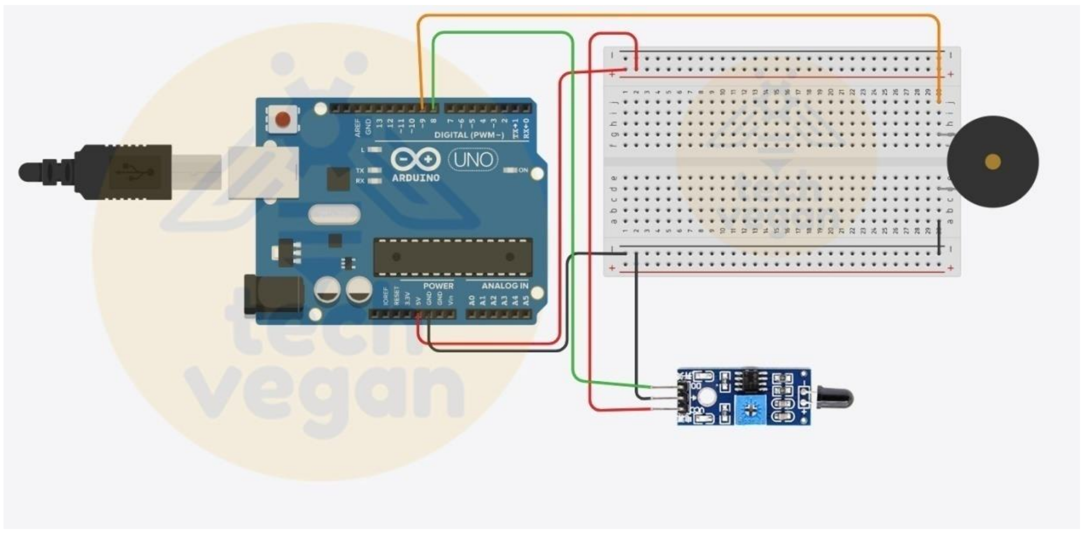
- Sensor D0 Pin – Arduino Pin 8
- Sensor GND Pin - Cathode
- Sensor VCC Pin – First Cathode.

Step 5 : Get link to The Buzzers Anode to the Breadboard to the Cathode

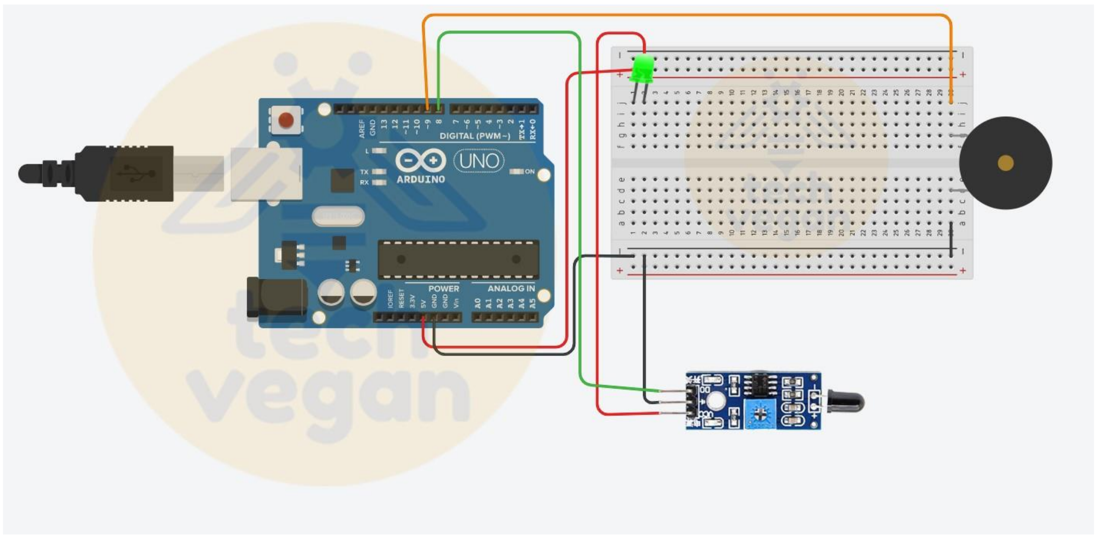
PROGRAM:

```
const int buzzerPin = 9;
const int fireSensorPin = 8;
void setup()
{
  Serial.begin(9600);
  pinMode(buzzerPin, OUTPUT);
  pinMode(fireSensorPin, INPUT);
}
void loop()
{
  int fireValue = digitalRead(fireSensorPin);
  Serial.println(fireValue);
  if(fireValue == 0)
  {
    digitalWrite(buzzerPin, HIGH);
    delay(5000);
  }
  else
  {
    digitalWrite(buzzerPin, LOW);
  }
  delay(500);
}
```

DIAGRAM:



OUTPUT:



RESULT:

Thus the above program is verified and executed successfully.

EX.No : 9 DATE :	**TO DEVELOP AN IOT PROGRAM FOR GAS LEAKAGE DETECTION (HOME, INDUSTRY)**

AIM:

To Develop An Iot Program For Gas Sensor Leakage Detection.

ALGORITHM:

Step1: Materials Needed

- Arduino board (e.g., Arduino Uno)
- Buzzer module
- LED(Red, green)(optional, for visual alert)
- Jumper wires
- Breadboard
- Power source (e.g., battery or USB cable)

Step 2: Connect the temperature sensor module to the Arduino board using jumper wires.

Step 3: Connect the buzzer or alarm module to the Arduino for audible alerts, Optionally, connect an LED for visual feedback.

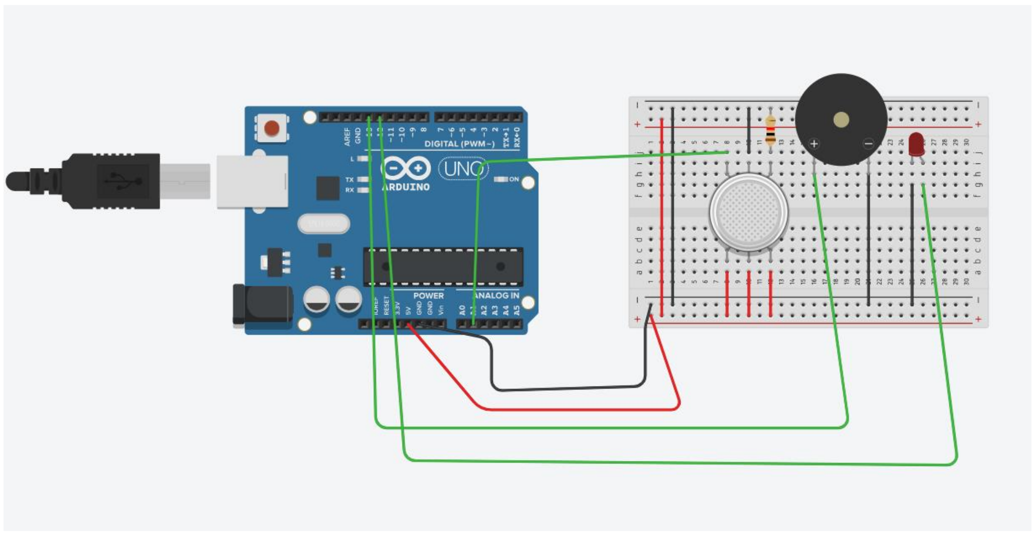
Step 4: We will connect the MQ-135 gas sensor with Arduino to detect Gas, Code Implementation arduino Uno.

Step 5: Test the system in a controlled environment with a heat source to simulate a fire. Once tested and calibrated, install the system in the desired location where fire detection is needed.

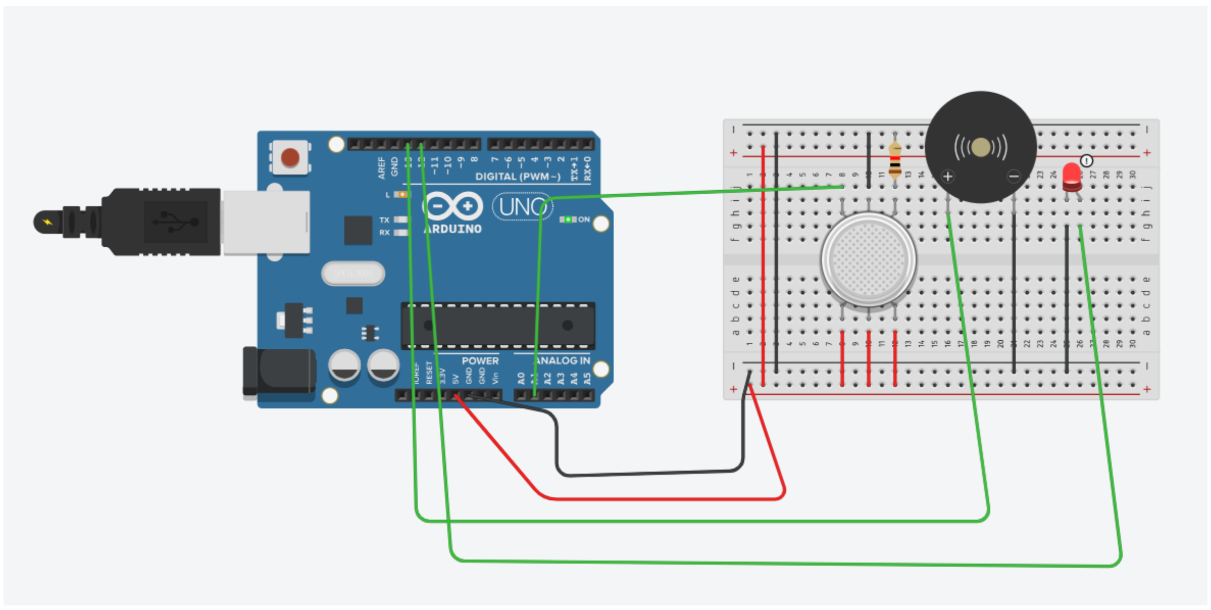
PROGRAM:

```
int sensorValue;
int buzzerPin = 10;
int greenLED = 11;
int redLED = 12;
int sensorPin = 5;

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  pinMode(buzzerPin, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(sensorPin, INPUT);

}
void loop() {
  sensorValue = analogRead(sensorPin);       // read analog input pin A5
  Serial.print("AirQuality Value: ");
  Serial.println(sensorValue, DEC);              // prints the value read

  if (sensorValue > 600) {
    digitalWrite(greenLED, LOW);
    digitalWrite(buzzerPin, HIGH);
    digitalWrite(redLED, HIGH);
    Serial.println("Alert!!!");
    delay(2000); // wait 2000ms
  }

  else {
    digitalWrite(greenLED, HIGH);
    digitalWrite(redLED, LOW);
    digitalWrite(buzzerPin, LOW);
    Serial.println("Normal");
    delay(500); // wait 500ms
  }
  delay(100);                              // wait 100ms for next reading
}
```

DIAGRAM:



OUTPUT:



RESULT:

Thus the above program is verified and executed successfully.

EX.No : 10 **DATE :**	**TO DEVELOP AN IOT PROGRAM USING HEARTBEAT SENSOR**

AIM:

To Develop An Iot Program Using Heartbeat Sensor.

ALGORITHM:

Step 1: Start the Program using Arduino Board

- ❖ Arduino Uno Board
- ❖ Jumper Wire
- ❖ 16x2 LCD Display
- ❖ Heartbeat Sensor Module with Probe (finger based)

Step 2: Take the Jumper wire

Step 3: Connect Heartbeat Sensor and 16x2 LCD Display with arduino board

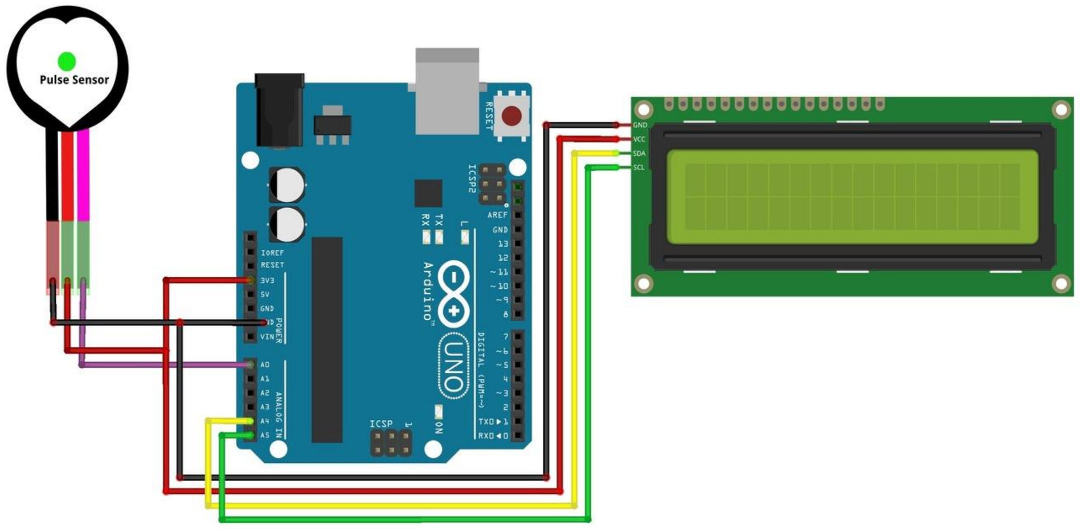
Step 4: Insert Jumper wires type the program.

Step 5: Run the program

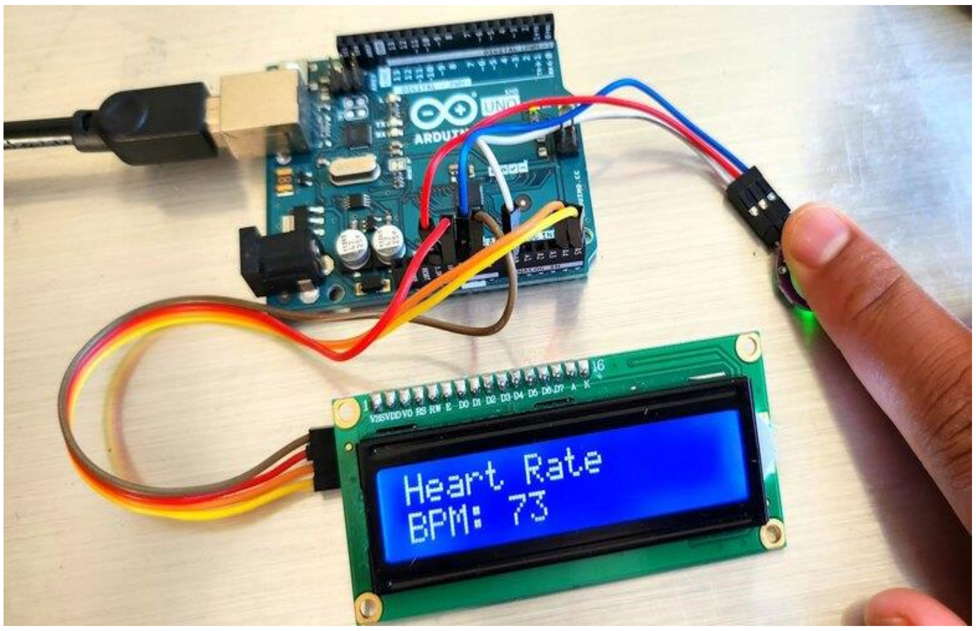
PROGRAM:

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3f, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
int pulsePin = A0;                // Pulse Sensor purple wire connected to analog pin A0
int blinkPin = 13;               // pin to blink led at each beat

// Volatile Variables, used in the interrupt service routine!
volatile int BPM;                   // int that holds raw Analog in 0. updated every 2mS
volatile int Signal;                // holds the incoming raw data
volatile int IBI = 600;             // int that holds the time interval between beats! Must be
seeded!
volatile boolean Pulse = false;     // "True" when User's live heartbeat is detected. "False"
when not a "live beat".
volatile boolean QS = false;        // becomes true when Arduoino finds a beat.

static boolean serialVisual = true;   // Set to 'false' by Default.  Re-set to 'true' to see Arduino
Serial Monitor ASCII Visual Pulse

volatile int rate[10];                      // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0;          // used to determine pulse timing
volatile unsigned long lastBeatTime = 0;           // used to find IBI
volatile int P = 512;                      // used to find peak in pulse wave, seeded
volatile int T = 512;                     // used to find trough in pulse wave, seeded
volatile int thresh = 525;                // used to find instant moment of heart beat, seeded
volatile int amp = 100;                   // used to hold amplitude of pulse waveform, seeded
volatile boolean firstBeat = true;        // used to seed rate array so we startup with reasonable
BPM
volatile boolean secondBeat = false;      // used to seed rate array so we startup with
reasonable BPM

void setup()
{

  Serial.begin(115200);
  // we agree to talk fast!
  interruptSetup();                 // sets up to read Pulse Sensor signal every 2mS
                                   // IF YOU ARE POWERING The Pulse Sensor AT VOLTAGE LESS
THAN THE BOARD VOLTAGE,
                                   // UN-COMMENT THE NEXT LINE AND APPLY THAT
VOLTAGE TO THE A-REF PIN
                                   //   analogReference(EXTERNAL);
 lcd.begin(16,2);

}


//  Where the Magic Happens
void loop()
{
   serialOutput();

  if (QS == true) // A Heartbeat Was Found
    {
      // BPM and IBI have been Determined
      // Quantified Self "QS" true when arduino finds a heartbeat
      serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.
      QS = false; // reset the Quantified Self flag for next time
    }

  delay(20); //  take a break
}


void interruptSetup()
{
  // Initializes Timer2 to throw an interrupt every 2mS.
  TCCR2A = 0x02;     // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO
CTC MODE
  TCCR2B = 0x06;     // DON'T FORCE COMPARE, 256 PRESCALER
  OCR2A = 0X7C;      // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE
RATE
  TIMSK2 = 0x02;     // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND
OCR2A
  sei();             // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}

void serialOutput()
{   // Decide How To Output Serial.
 if (serialVisual == true)
  {
     arduinoSerialMonitorVisual('-', Signal);   // goes to function that makes Serial Monitor
Visualizer
  }
 else
  {
      sendDataToSerial('S', Signal);     // goes to sendDataToSerial function
   }
}

void serialOutputWhenBeatHappens()
{
 if (serialVisual == true) //  Code to Make the Serial Monitor Visualizer Work
   {   lcd.setCursor(0,0);
     Serial.print(" Heart-Beat Found ");  //ASCII Art Madness
     Serial.print("BPM: ");
     Serial.println(BPM);
     lcd.print("Heart-Beat Found  ");
     lcd.setCursor(1,1);
     lcd.print("BPM: ");
     lcd.setCursor(5,1);
     lcd.print(BPM);
     delay(300);

    lcd.clear();

   }
 else
   {
     sendDataToSerial('B',BPM);   // send heart rate with a 'B' prefix
     sendDataToSerial('Q',IBI);   // send time between beats with a 'Q' prefix
   }
}

void arduinoSerialMonitorVisual(char symbol, int data )
{
  const int sensorMin = 0;      // sensor minimum, discovered through experiment
  const int sensorMax = 1024;    // sensor maximum, discovered through experiment
  int sensorReading = data; // map the sensor range to a range of 12 options:
  int range = map(sensorReading, sensorMin, sensorMax, 0, 11);
  // do something different depending on the
  // range value:
}


void sendDataToSerial(char symbol, int data )
{
   Serial.print(symbol);
   Serial.println(data);
}

ISR(TIMER2_COMPA_vect) //triggered when Timer2 counts to 124
{
  cli();                                      // disable interrupts while we do this
  Signal = analogRead(pulsePin);              // read the Pulse Sensor
  sampleCounter += 2;                         // keep track of the time in mS with this variable
  int N = sampleCounter - lastBeatTime;       // monitor the time since the last beat to avoid
noise
                                              //  find the peak and trough of the pulse wave
  if(Signal < thresh && N > (IBI/5)*3) // avoid dichrotic noise by waiting 3/5 of last IBI
    {
      if (Signal < T) // T is the trough
      {
        T = Signal; // keep track of lowest point in pulse wave
      }
    }

  if(Signal > thresh && Signal > P)
    {          // thresh condition helps avoid noise
      P = Signal;                             // P is the peak
    }                                   // keep track of highest point in pulse wave

  //  NOW IT'S TIME TO LOOK FOR THE HEART BEAT
  // signal surges up in value every time there is a pulse
  if (N > 250)
  {                              // avoid high frequency noise
    if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )
      {
        Pulse = true;                          // set the Pulse flag when we think there is a pulse
        digitalWrite(blinkPin,HIGH);               // turn on pin 13 LED
        IBI = sampleCounter - lastBeatTime;         // measure time between beats in mS
        lastBeatTime = sampleCounter;              // keep track of time for next pulse

        if(secondBeat)
        {                       // if this is the second beat, if secondBeat == TRUE
          secondBeat = false;                 // clear secondBeat flag
          for(int i=0; i<=9; i++) // seed the running total to get a realisitic BPM at startup
          {
            rate[i] = IBI;
          }
        }

        if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE
        {
          firstBeat = false;                  // clear firstBeat flag
          secondBeat = true;                  // set the second beat flag
          sei();                              // enable interrupts again
          return;                             // IBI value is unreliable so discard it
        }
      // keep a running total of the last 10 IBI values
      word runningTotal = 0;                  // clear the runningTotal variable

      for(int i=0; i<=8; i++)
        {               // shift data in the rate array
          rate[i] = rate[i+1];                  // and drop the oldest IBI value
          runningTotal += rate[i];              // add up the 9 oldest IBI values
        }

      rate[9] = IBI;                          // add the latest IBI to the rate array
      runningTotal += rate[9];                // add the latest IBI to runningTotal
      runningTotal /= 10;                     // average the last 10 IBI values
      BPM = 60000/runningTotal;               // how many beats can fit into a minute? that's
BPM!
      QS = true;                              // set Quantified Self flag
      // QS FLAG IS NOT CLEARED INSIDE THIS ISR
    }
  }

  if (Signal < thresh && Pulse == true)
    {   // when the values are going down, the beat is over
      digitalWrite(blinkPin,LOW);            // turn off pin 13 LED
      Pulse = false;                         // reset the Pulse flag so we can do it again
      amp = P - T;                           // get amplitude of the pulse wave
      thresh = amp/2 + T;                    // set thresh at 50% of the amplitude
      P = thresh;                            // reset these for next time
      T = thresh;
    }

  if (N > 2500)
    {                          // if 2.5 seconds go by without a beat
      thresh = 512;                          // set thresh default
      P = 512;                               // set P default
      T = 512;                               // set T default
      lastBeatTime = sampleCounter;          // bring the lastBeatTime up to date
      firstBeat = true;                      // set these to avoid noise
      secondBeat = false;                    // when we get the heartbeat back
    }

  sei();                                     // enable interrupts when youre done!
```

DIAGRAM:



OUTPUT:



RESULT:

Thus the above program is verified and executed successfully.