DATE :	**Introduction to Arduino, ESP 32, Arduino IDE, Tinkercad, Fritzing, and some practical examples**
EXP NO : 1	

AIM :

To introduce and demonstrate the basics of Arduino, ESP32, Arduino IDE, Tinkercad, and Fritzing, along with some practical examples.

COMPONENTS REQUIRED :

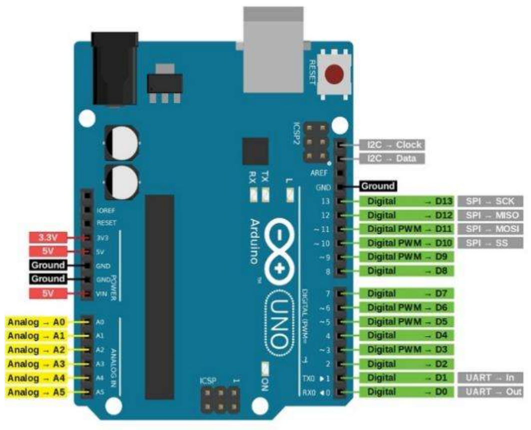
Arduino Uno, ESP32, Breadboard, LEDs, Resistors, Jumper wires, USB cables (for power and programming), Computer with Arduino IDE installed, Internet connection for Tinkercad and Fritzing

THEORY :

Arduino Uno :

The Arduino Uno is an open-source microcontroller board based on the ATmega328P. It has 14 digital input/output pins (6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. It is widely used for prototyping and educational purposes. The board can be programmed using the Arduino IDE, which supports C and C++ languages.

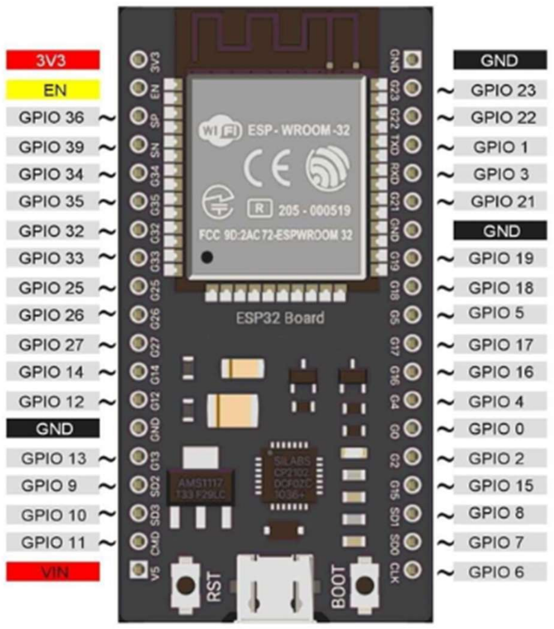
Pin Diagram:



ESP32 :

The ESP32 is a powerful microcontroller with integrated Wi-Fi and Bluetooth capabilities. It is based on the Xtensa LX6 dual-core processor and has 32 programmable GPIOs. The ESP32 can be programmed using the Arduino IDE, which makes it accessible for beginners and hobbyists. It is commonly used in IoT projects due to its connectivity features.

Pin Diagram:



Arduino IDE:

The Arduino Integrated Development Environment (IDE) is an open-source software used for writing, compiling, and uploading code to Arduino-compatible boards. It supports a wide range of libraries and has a simple interface, making it easy for beginners to get started with programming microcontrollers.

Tinkercad :

Tinkercad is a free, online 3D modeling and circuit simulation tool. It allows users to create and simulate circuits using virtual components. Tinkercad is especially useful for beginners to test and visualize their circuits before building them in real life.

Fritzing:

Fritzing is an open-source hardware initiative that makes electronics accessible as a creative material for anyone. It provides a software tool to create circuit designs, which can be transformed into real projects. Fritzing is used to document circuits, create PCB layouts, and produce schematics.

PROCEDURE :

1. Connect the sensors and components to the microcontroller following their respective datasheets.
2. Write a program to initialize the microcontroller and configure the input/output pins.
3. Upload the program to the microcontroller using the Arduino IDE.
4. Observe the output on the serial monitor and verify the behavior of the connected components.

OUTPUT:

INFERENCE :

DATE :	**Design Embedded Systems and work with analog, and digital inputs & outputs using sensors.**
EXP NO : 2	

AIM :

To design embedded systems and work with analog and digital inputs and outputs using sensors.

COMPONENTS REQUIRED :

Arduino Uno, ESP32, Breadboard, LEDs, Resistors, Jumper wires, USB cables (for power and programming), Temperature sensor (e.g., LM35), Light sensor (e.g., LDR), Push buttons, Potentiometer, Computer with Arduino IDE installed

THEORY :

Embedded Systems :

Embedded systems are specialized computing systems that perform dedicated functions or tasks within larger mechanical or electrical systems. They often include microcontrollers or microprocessors to process inputs and control outputs.

Sensors:

Sensors convert physical phenomena (like temperature, light, or pressure) into electrical signals that can be processed by microcontrollers. For instance, an IR sensor provides an analog voltage proportional to the proximity of an object, and an LDR (Light Dependent Resistor) changes resistance based on light intensity.

IR Sensor :

An IR (Infrared) sensor detects infrared radiation emitted or reflected by objects in its vicinity, allowing it to sense their presence or proximity. It is commonly used for object detection, distance measurement, and motion tracking in various applications.

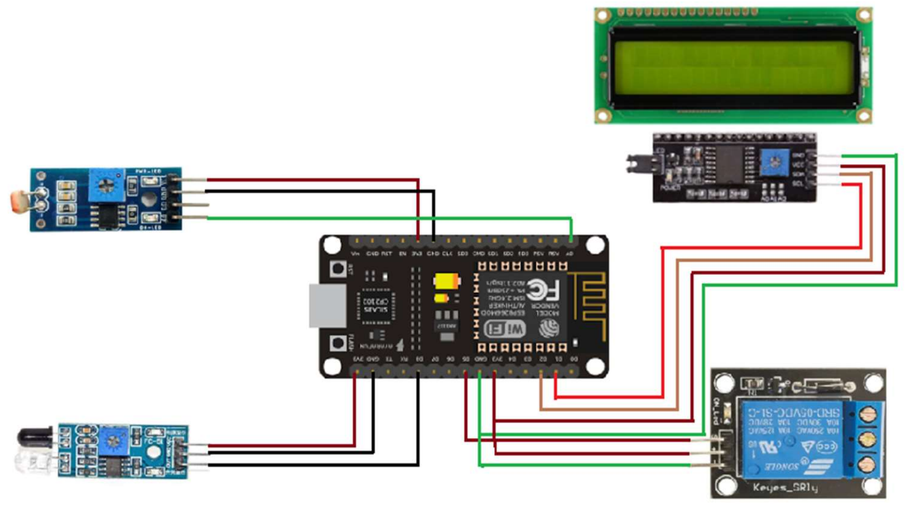
LDR Sensor :

An LDR (Light Dependent Resistor) is a sensor that changes its resistance based on the intensity of light falling on it, decreasing resistance as light intensity increases. It is widely used in applications such as automatic lighting systems and light-sensitive alarms.

PROCEDURE :

1. Connect the sensors and components to the microcontroller following their respective datasheets.
2. Write a program to initialize the microcontroller and configure the input/output pins.
3. Upload the program to the microcontroller using the Arduino IDE.
4. Observe the output on the serial monitor and verify the behavior of the connected components.

CIRCUIT DIAGRAM :\

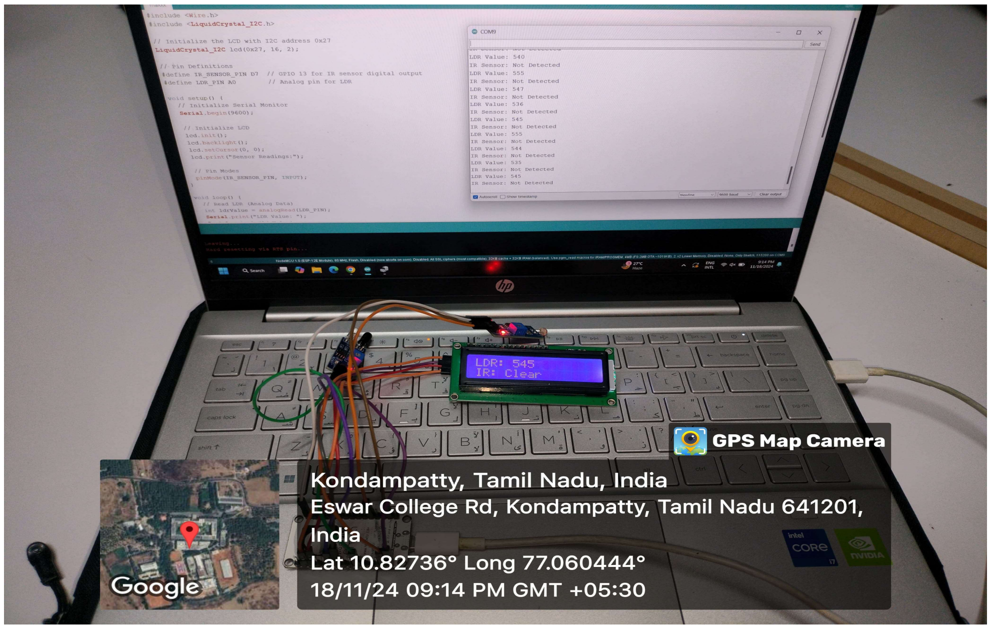


CODE:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h> // Include the I2C library

// Define I2C LCD address and dimensions
LiquidCrystal_I2C lcd(0x27, 16, 2); // Adjust the I2C address if needed

// Pin definitions
```

```
#define IR_SENSOR_PIN D6 // GPIO12 for IR sensor
#define LDR_SENSOR_PIN A0 // Analog pin for LDR

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Initialize I2C LCD with dimensions
  lcd.begin(16, 2); // Specify 16 columns and 2 rows
  lcd.backlight();  // Turn on the LCD backlight

  // Initialize pins
  pinMode(IR_SENSOR_PIN, INPUT);

  // Display initial message
  lcd.setCursor(0, 0);
  lcd.print("IR: ");
  lcd.setCursor(0, 1);
  lcd.print("LDR: ");
}

void loop() {
  // Read values from IR and LDR sensors
  int irValue = digitalRead(IR_SENSOR_PIN); // Digital value (0 or 1)
  int ldrValue = analogRead(LDR_SENSOR_PIN); // Analog value (0 to 1023)

  // Debugging values on Serial Monitor
  Serial.print("IR Value: ");
  Serial.println(irValue);
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);

  // Display values on I2C LCD
  lcd.setCursor(4, 0);
  lcd.print(irValue == 1 ? "Detected" : "Not Det");
  lcd.setCursor(5, 1);
  lcd.print(ldrValue);
  lcd.print("    "); // Clear trailing digits

  delay(500); // Update every 500ms
}
```

OUTPUT:



```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Initialize the LCD with I2C address 0x27
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Pin Definitions
#define IR_SENSOR_PIN D7  // GPIO 13 for IR sensor digital output
#define LDR_PIN A0        // Analog pin for LDR

void setup() {
  // Initialize Serial Monitor
  Serial.begin(9600);

  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Sensor Readings:");

  // Pin Modes
  pinMode(IR_SENSOR_PIN, INPUT);
}

void loop() {
  // Read LDR (Analog Data)
  int ldrValue = analogRead(LDR_PIN);
  Serial.print("LDR Value: ");
```

```
Leaving...
Hard resetting via RTS pin...
```

```
NodeMCU 1.0 (ESP-12E Module), 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM9
```

```
LDR Value: 540
IR Sensor: Not Detected
LDR Value: 555
IR Sensor: Not Detected
LDR Value: 547
IR Sensor: Not Detected
LDR Value: 536
IR Sensor: Not Detected
LDR Value: 545
IR Sensor: Not Detected
LDR Value: 555
IR Sensor: Not Detected
LDR Value: 544
IR Sensor: Not Detected
LDR Value: 535
IR Sensor: Not Detected
LDR Value: 545
IR Sensor: Not Detected
```

```
LDR: 545
IR: Clear
```

Kondampatty, Tamil Nadu, India
Eswar College Rd, Kondampatty, Tamil Nadu 641201, India
Lat 10.82736° Long 77.060444°
18/11/24 09:14 PM GMT +05:30

INFERENCE :

DATE :	**Design a circuit interfacing OLED, microcontroller, and sensors-based Projects.**
EXP NO : 3	

AIM :

To design a circuit interfacing an OLED display, microcontroller, and sensors for various projects.

COMPONENTS REQUIRED :

Microcontroller (e.g., Arduino Uno or ESP32), 0.96" OLED 64x128 Display Module, Temperature sensor (e.g., LM35), Light sensor (e.g., LDR), Humidity sensor (e.g., DHT11), Breadboard, Resistors, Jumper wires, USB cable (for power and programming), Computer with Arduino IDE installed

THEORY :

Microcontroller:

A microcontroller is a small computer on a single chip that controls embedded systems by processing inputs, executing instructions, and managing outputs. It integrates a processor, memory, and I/O peripherals to interface with sensors and displays.

LDR Sensor :

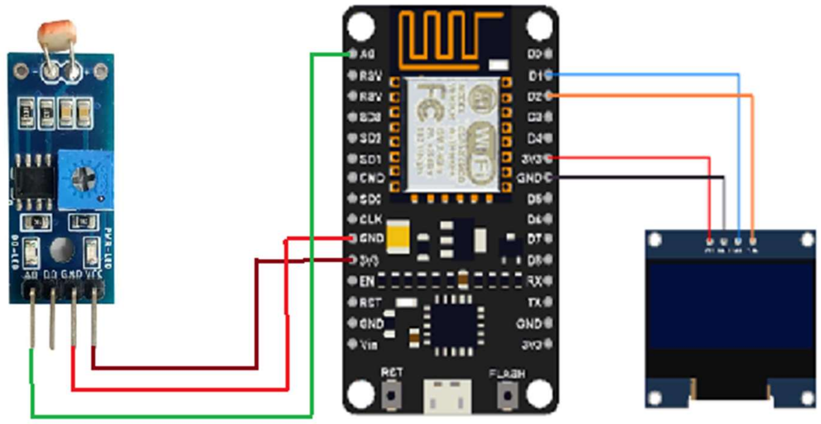
An LDR (Light Dependent Resistor) is a sensor that changes its resistance based on the intensity of light falling on it, decreasing resistance as light intensity increases. It is widely used in applications such as automatic lighting systems and light-sensitive alarms.

PROCEDURE :

1. Connect the OLED display to the microcontroller (VCC to power, GND to ground, SDA to data pin, SCL to clock pin).
2. Connect the sensors to the microcontroller (VCC to power, GND to ground, output pins to analog or digital input pins).
3. Write a program to initialize the microcontroller, configure the input/output pins, and interface with the OLED display.

4. Include code to read sensor data and display it on the OLED screen.
5. Upload the program to the microcontroller using the Arduino IDE.
6. Observe the OLED display to verify that the sensor data is being correctly displayed.

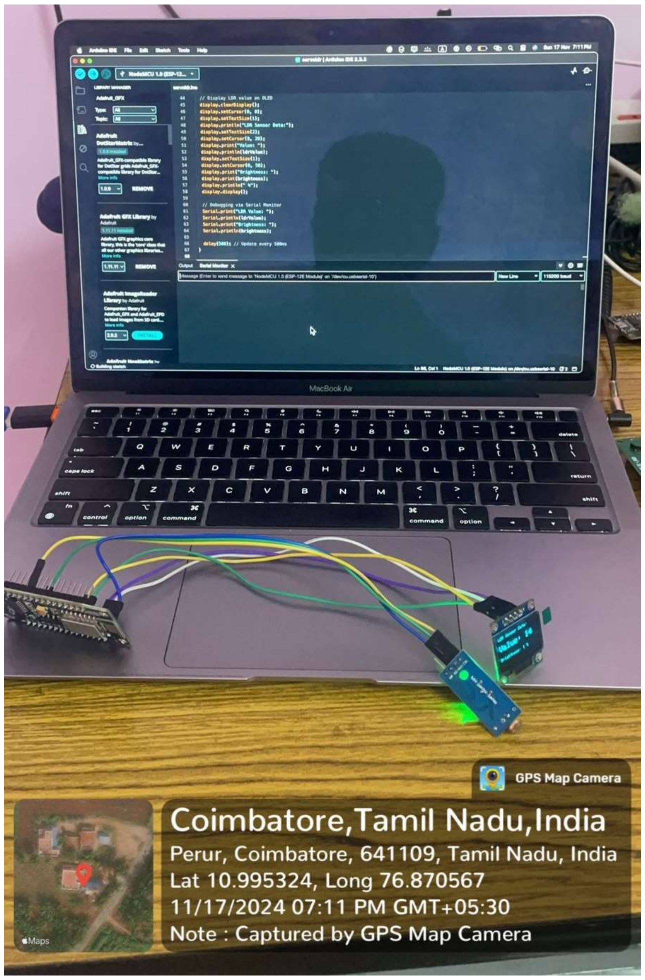
CIRCUIT DIAGRAM :



CODE:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// OLED display dimensions
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// OLED I2C address
#define OLED_ADDR 0x3C  // You can check if the address is 0x3C or 0x3D
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// LDR sensor pin
#define LDR_PIN A0

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Initialize OLED display with I2C address
  if (!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)) {  // Correct initialization
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);  // Halt if initialization fails
  }
  // Clear the display and set initial text
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println(F("LDR Sensor Ready"));
  display.display();
  delay(2000);  // Wait for 2 seconds
}

void loop() {
  // Read LDR sensor value
  int ldrValue = analogRead(LDR_PIN);

  // Map LDR value to brightness level (optional)
  int brightness = map(ldrValue, 0, 1023, 0, 100); // Scale to 0-100%

  // Display LDR value on OLED
  display.clearDisplay();
  display.setCursor(0, 0);
  display.setTextSize(1);
  display.println("LDR Sensor Data:");
  display.setTextSize(2);
  display.setCursor(0, 20);
  display.print("Value: ");
  display.println(ldrValue);
  display.setTextSize(1);
  display.setCursor(0, 50);
  display.print("Brightness: ");
  display.print(brightness);
  display.println(" %");
  display.display();

  // Debugging via Serial Monitor
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);
  Serial.print("Brightness: ");
  Serial.println(brightness);

  delay(500); // Update every 500ms
}
```

OUTPUT:



```
// Display LDR value on OLED
display.clearDisplay();
display.setCursor(0, 0);
display.setTextSize(1);
display.println("LDR Sensor Data:");
display.setTextSize(2);
display.setCursor(0, 20);
display.print("Value: ");
display.println(ldrValue);
display.setTextSize(1);
display.setCursor(0, 50);
display.print("Brightness: ");
display.print(brightness);
display.println(" %");
display.display();

// Debugging via Serial Monitor
Serial.print("LDR Value: ");
Serial.println(ldrValue);
Serial.print("Brightness: ");
Serial.println(brightness);

delay(500); // Update every 500ms
}
```

Sun 17 Nov 7:11 PM
servoldr | Arduino IDE 2.3.3
NodeMCU 1.0 (ESP-12...
servoldr.ino
Output
Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on '/dev/cu.usbserial-10')
New Line
115200 baud
REMOVE
INSTALL
MacBook Air
GPS Map Camera
Coimbatore,Tamil Nadu,India
Perur, Coimbatore, 641109, Tamil Nadu, India
Lat 10.995324, Long 76.870567
11/17/2024 07:11 PM GMT+05:30
Note : Captured by GPS Map Camera
Maps

INFERENCE :

DATE :	**Design a circuit interfacing GSM, GPS, microcontroller, and sensors for tracking applications.**
EXP NO : 4	

AIM :

To design a circuit interfacing GSM, GPS, a microcontroller, and sensors for tracking applications.

COMPONENTS REQUIRED :

Microcontroller (e.g., Arduino Uno or ESP32), GSM Module (e.g., SIM800L), GPS Module (e.g., NEO-6M), Breadboard, Jumper wires, Sensors (e.g., accelerometer, temperature sensor), Resistors, USB cable (for power and programming), Computer with Arduino IDE installed

THEORY :

Microcontroller:

A microcontroller processes input from sensors and manages communication with GSM and GPS modules to handle tracking data and notifications.

GSM Module:

The GSM module enables mobile communication by sending and receiving SMS messages or making calls, allowing for remote tracking and alerts.

GPS Module:

The GPS module provides geographical location data by receiving signals from satellites, used to determine the current position for tracking purposes.

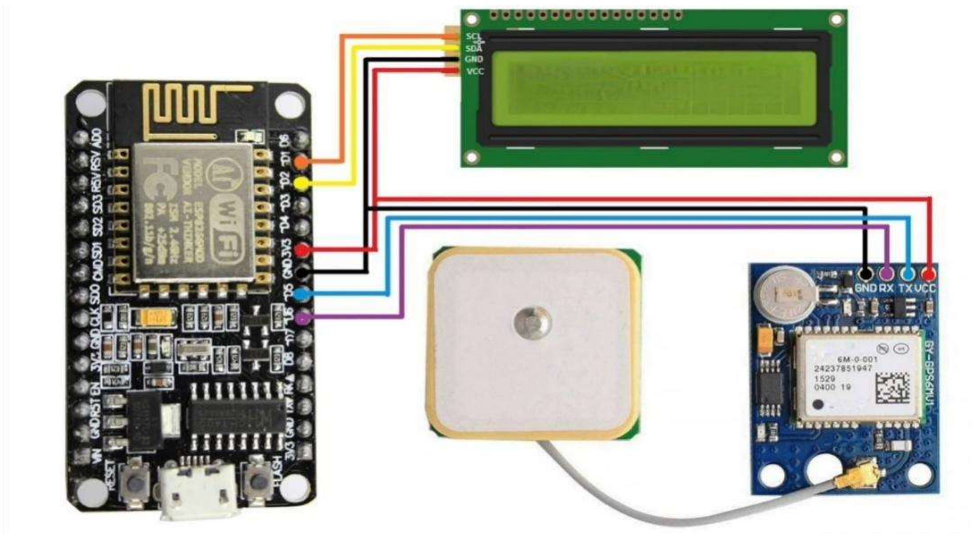
DHT11 Sensors:

The DHT11 sensor is a digital sensor that measures both temperature and humidity, providing calibrated and easy-to-read data. It is commonly used in environmental monitoring systems due to its reliability and low cost.

PROCEDURE :

1. Connect GSM and GPS modules to the microcontroller.
2. Connect sensors to the microcontroller.
3. Write the program for module and sensor integration.
4. Upload the program to the microcontroller.
5. Observe and test the outputs.

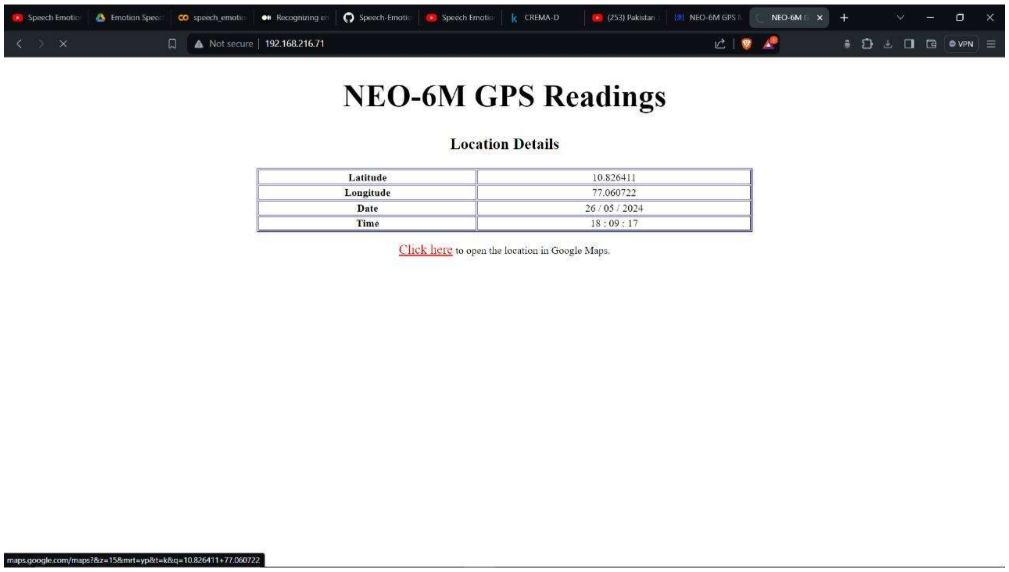
CIRCUIT DIAGRAM :



CODE:

```
#include <DHT.h>
#include <SoftwareSerial.h>
#define DHTPIN D5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
SoftwareSerial gpsSerial(D2, D1);

void setup() {
  Serial.begin(9600);
  gpsSerial.begin(9600);
  dht.begin();
  delay(1000);
}
void loop() {
  // Read temperature and humidity from the DHT11 sensor
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Read and parse GPS data
  String gpsData = readGPSData();
  Serial.println("GPS Data: " + gpsData);

  // Output sensor data
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");

  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");
  delay(10000);
}

String readGPSData() {
  String gpsString = "";
  while (gpsSerial.available()) {
    char c = gpsSerial.read();
    gpsString += c;
    if (c == '\n') break;
  }
  return parseGPSData(gpsString);
}

String parseGPSData(String rawData) {
  String latitude = "Unknown";
  String longitude = "Unknown";
  if (rawData.startsWith("$GPGGA")) {
    int latIndex = rawData.indexOf(",") + 1;
    int lonIndex = rawData.indexOf(",", latIndex + 9) + 1;
    latitude = rawData.substring(latIndex, latIndex + 9);
    longitude = rawData.substring(lonIndex, lonIndex + 10);
  }
  return "Latitude: " + latitude + ", Longitude: " + longitude;
}
```

OUTPUT:



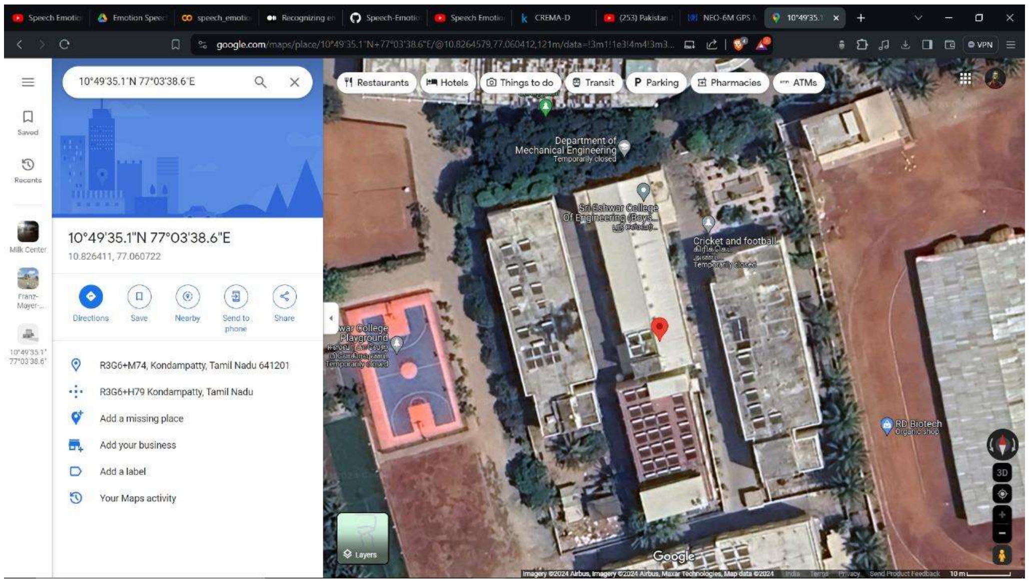
NEO-6M GPS Readings

Location Details

Latitude	10.826411
Longitude	77.060722
Date	26 / 05 / 2024
Time	18 : 09 : 17

Click here to open the location in Google Maps.

Not secure | 192.168.216.71
NEO-6M GPS
maps.google.com/maps?&z=15&mrt=yp&t=k&q=10.826411+77.060722



INFERENCE :

DATE :	**Design a circuit interfacing Data logger, microcontroller and sensors for manufacturing application.**
EXP NO : 5	

AIM :

To design a circuit interfacing a data logger, microcontroller, and sensors for manufacturing applications.

COMPONENTS REQUIRED :

Microcontroller (e.g., Arduino Uno or ESP32), Data Logger (e.g., SD Card module), Sensors (e.g., temperature sensor, pressure sensor), Breadboard, Jumper wires, Resistors, USB cable (for power and programming), Computer with Arduino IDE installed

THEORY :

Microcontroller:

A microcontroller manages data acquisition from sensors and communicates with the data logger to store data for further analysis.

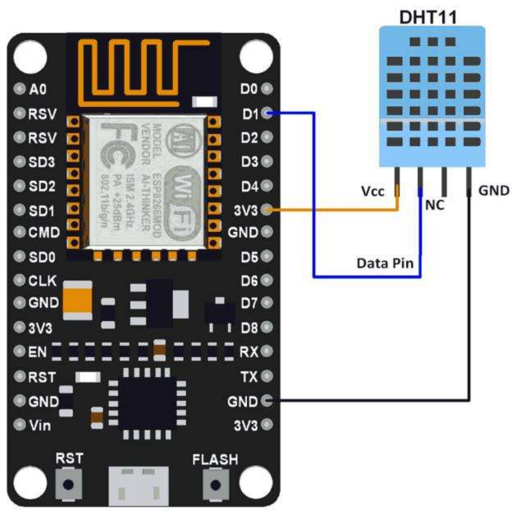
Data Logger:

A data logger records data from sensors onto storage media (e.g., SD card), providing a permanent record of measurements over time for manufacturing process monitoring.

PROCEDURE :

1. Connect the data logger to the microcontroller.
2. Connect the sensors to the microcontroller.
3. Write the program for sensor data logging.
4. Upload the program to the microcontroller.
5. Observe and test the data logging.

CIRCUIT DIAGRAM :

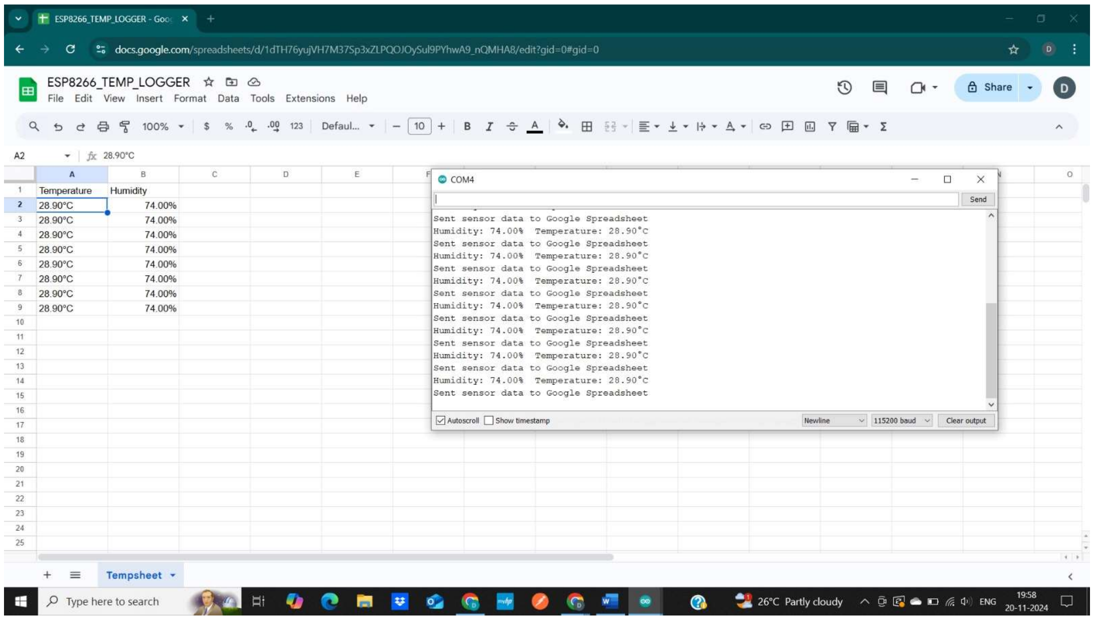


CODE:

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <DHT.h>
#define DHTPIN D4  // DHT sensor pin
#define DHTTYPE DHT11  // DHT11 sensor type
DHT dht(DHTPIN, DHTTYPE);
float h;
float t;
String sheetHumid = "";
String sheetTemp = "";
const char* ssid = "VivoT1";  // Replace with your WiFi SSID
const char* password = "mahizhan";  // Replace with your WiFi password
const char* host = "script.google.com";
const char* GScriptId =
"AKfycbxgjyDQM5TVAw0jVjZeDrrS3PiRIxzfl_EvNnDdKNj7MdfQxyTuupejmBMtabD
kcTnrSAee";  // Replace with your own Google Script ID
const int httpsPort = 443;  // HTTPS port
String url = String("/macros/s/") + GScriptId + "/exec?value=Temperature";
String url2 = String("/macros/s/") + GScriptId + "/exec?cal";
String payload_base = "{\"command\": \"appendRow\", "
                    "\"sheet_name\": \"TempSheet\", "
```

```
                      "\"values\": ";
String payload = "";
WiFiClientSecure client;
void setup() {
  delay(1000);
  Serial.begin(115200);
  dht.begin();
  Serial.println();
  Serial.print("Connecting to WiFi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setInsecure();  // Disable SSL certificate validation
  client.setTimeout(10000);  // Set timeout for client connection
  Serial.print("Connecting to ");
  Serial.println(host);
  if (!client.connect(host, httpsPort)) {
    Serial.println("Connection failed.");
    return;
  }
  Serial.println("Connected to server");
  client.print("GET " + url + " HTTP/1.1\r\n");
  client.print("Host: " + String(host) + "\r\n");
  client.print("Connection: close\r\n\r\n");
  Serial.println("Sent GET request to fetch data");
  client.print("GET " + url2 + " HTTP/1.1\r\n");
  client.print("Host: " + String(host) + "\r\n");
  client.print("Connection: close\r\n\r\n");
  Serial.println("Sent GET request to fetch calendar data");
}
void loop() {
  h = dht.readHumidity();
  t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  sheetHumid = String(h) + "%";
  Serial.print("%  Temperature: ");
  Serial.print(t);
  Serial.println("°C");
  sheetTemp = String(t) + "°C";
  payload = payload_base + "\"" + sheetTemp + "," + sheetHumid + "\"}";
  if (!client.connected()) {
    Serial.println("Disconnected, reconnecting...");
    if (!client.connect(host, httpsPort)) {
      Serial.println("Reconnection failed.");
      return;
    }
  }
  client.print("POST " + url2 + " HTTP/1.1\r\n");
  client.print("Host: " + String(host) + "\r\n");
  client.print("Content-Type: application/json\r\n");
  client.print("Content-Length: " + String(payload.length()) + "\r\n");
  client.print("\r\n");
  client.print(payload);
  Serial.println("Sent sensor data to Google Spreadsheet");
  delay(3000);  // Delay for DHT sensor reading interval
}
```

OUTPUT:


ESP8266_TEMP_LOGGER - Goo
docs.google.com/spreadsheets/d/1dTH76yujVH7M37Sp3xZLPQOJOySul9PYhwA9_nQMHA8/edit?gid=0#gid=0
ESP8266_TEMP_LOGGER
File Edit View Insert Format Data Tools Extensions Help
Share
A2 28.90°C
COM4
Send
Tempsheet
Type here to search
26°C Partly cloudy
ENG
19:58
20-11-2024

	A	B
1	Temperature	Humidity
2	28.90°C	74.00%
3	28.90°C	74.00%
4	28.90°C	74.00%
5	28.90°C	74.00%
6	28.90°C	74.00%
7	28.90°C	74.00%
8	28.90°C	74.00%
9	28.90°C	74.00%

```
Sent sensor data to Google Spreadsheet
Humidity: 74.00%  Temperature: 28.90°C
Sent sensor data to Google Spreadsheet
Humidity: 74.00%  Temperature: 28.90°C
Sent sensor data to Google Spreadsheet
Humidity: 74.00%  Temperature: 28.90°C
Sent sensor data to Google Spreadsheet
Humidity: 74.00%  Temperature: 28.90°C
Sent sensor data to Google Spreadsheet
Humidity: 74.00%  Temperature: 28.90°C
Sent sensor data to Google Spreadsheet
Humidity: 74.00%  Temperature: 28.90°C
Sent sensor data to Google Spreadsheet
Humidity: 74.00%  Temperature: 28.90°C
Sent sensor data to Google Spreadsheet
```

☑ Autoscroll ☐ Show timestamp Newline 115200 baud Clear output

INFERENCE :

DATE :	**Displaying Text/Images using OLED**
EXP NO : 6	

AIM :

To create an app using MIT App Inventor and control hardware components (e.g., microcontroller-based projects) through the app.

COMPONENTS REQUIRED :

MIT App Inventor (online tool), Microcontroller (e.g., Arduino Uno or ESP32), GSM Module (for sending/receiving commands), Sensors or actuators (depending on the project), Bluetooth or Wi-Fi module (for communication), Smartphone or tablet (for running the app).

THEORY :

MIT App Inventor:

MIT App Inventor is a web-based tool for designing and building Android apps using a visual programming language. It allows users to create apps that can interact with hardware components through Bluetooth or Wi-Fi.

Microcontroller:

A microcontroller processes commands received from the app and controls connected hardware components (e.g., sensors, actuators) accordingly.

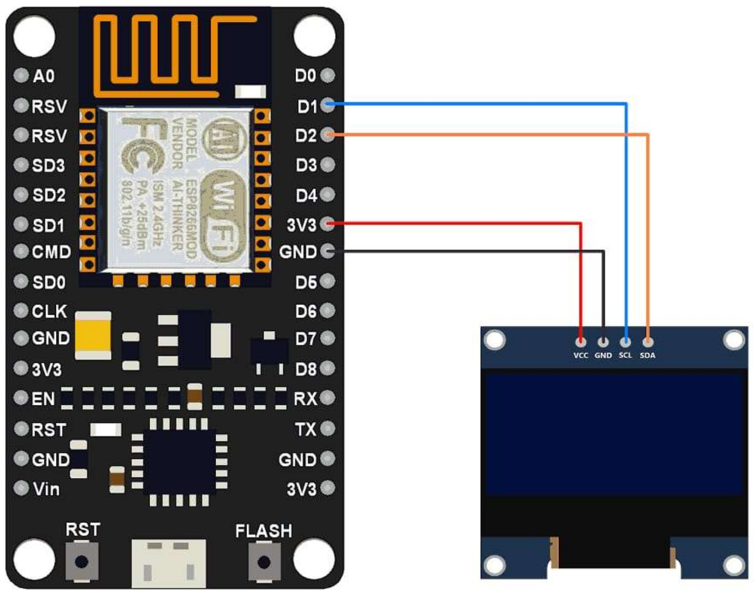
Communication Module:

Bluetooth or Wi-Fi modules enable wireless communication between the app and the microcontroller, allowing remote control of hardware.

PROCEDURE :

1. Design the app in MIT App Inventor.
2. Configure hardware communication with the microcontroller.
3. Connect and test the app with the hardware.
4. Finalize and deploy the app.
5. Observe and debug functionality.

CIRCUIT DIAGRAM :

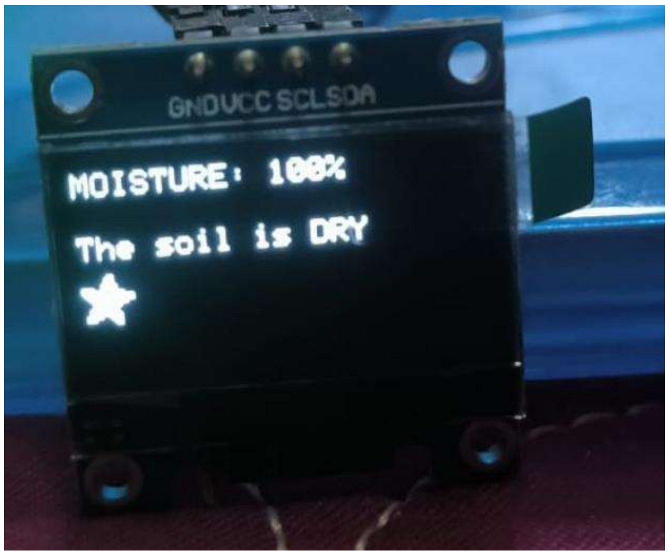


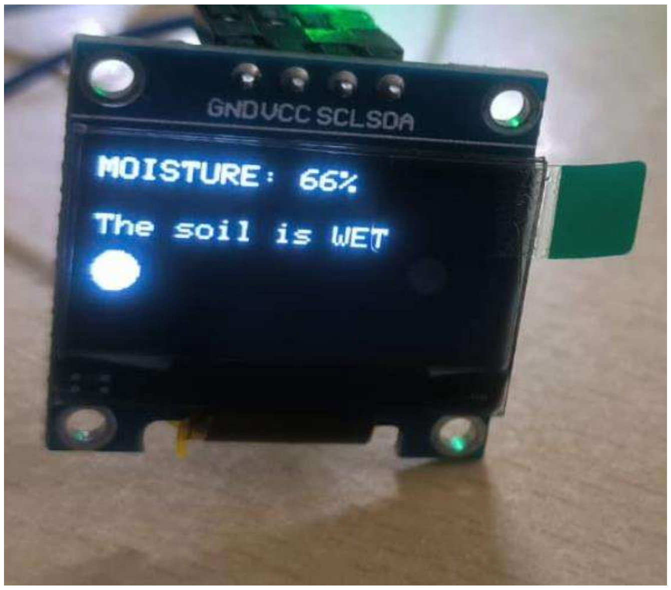
CODE:

```
            #define BLYNK_TEMPLATE_ID "TMPL3HwYX3Wxw"
#define BLYNK_TEMPLATE_NAME "LCD"
#define BLYNK_AUTH_TOKEN "30wtTIrPt7J7PbcRO6VxPC6zmcqegZPY"
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#define BLYNK_PRINT Serial
char ssid[] = "Bharath";
char pass[] = "bharath@22";
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>#define OLED_RESET -1
Adafruit_SSD1306 display(128, 64, &Wire, OLED_RESET);
#define SDA_PIN D2
#define SCL_PIN D1
int sense_Pin = A0;
int value = 0;
const uint8_t PROGMEM dry_soil_bmp[] =
{0b00000000, 0b11000000,
0b00000001, 0b11000000,
0b00000001, 0b11000000,
0b00000011, 0b11100000,
0b11110011, 0b11100000,
0b11111110, 0b11111000,
0b01111110, 0b11111111,
0b00110011, 0b10011111,
0b00011111, 0b11111100,
0b00001101, 0b01110000,
0b00011011, 0b10100000,
0b00111111, 0b11100000,
0b00111111, 0b11110000,
0b01111100, 0b11110000,
0b01110000, 0b01110000,
0b00000000, 0b00110000 };
const uint8_t PROGMEM wet_soil_bmp[] =
{ 0b00000000, 0b00000000,
0b00001111, 0b10000000,
0b00011111, 0b11100000,
0b00111111, 0b11110000,
0b01111111, 0b11111000,
0b01111111, 0b11111000,0b11111111, 0b11111100,
0b11111111, 0b11111100,
0b11111111, 0b11111100,
0b11111111, 0b11111100,
0b01111111, 0b11111000,
0b01111111, 0b11111000,
0b00111111, 0b11110000,
0b00011111, 0b11100000,
0b00001111, 0b10000000,
0b00000000, 0b00000000};
BLYNK_CONNECTED() {
Blynk.syncVirtual(V0);
Blynk.syncVirtual(V1);
}
void setup(){
Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
Wire.begin(SDA_PIN, SCL_PIN);
Serial.begin(9600);
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
Serial.println(F("SSD1306 allocation failed"));
for(;;);
}
display.clearDisplay();
display.display();
}
void loop() {
Blynk.run();
value = analogRead(sense_Pin);
Blynk.virtualWrite(V1,value );
value = (value / 10)-2;
Serial.println(value);
Serial.print("%");
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(0, 0);
display.print("MOISTURE: ");
display.print(value);
display.print("%");
display.setCursor(0, 20);
```

```
if (value < 90) {
display.print("The soil is WET");
Blynk.virtualWrite(V0, 0);
display.drawBitmap(0, 32, wet_soil_bmp, 16, 16, SSD1306_WHITE);
} else {
display.print("The soil is DRY");
Blynk.virtualWrite(V0, 1);
display.drawBitmap(0, 32, dry_soil_bmp, 16, 16, SSD1306_WHITE);
}
display.display();
delay(1000);
}
```

OUTPUT:





INFERENCE :

DATE :	**Introduction to Cloud, some IoT Cloud Platforms publish sensor data to a cloud using Thingspeak**
EXP NO : 7	

AIM :

To introduce cloud computing concepts and demonstrate how to publish sensor data to a cloud platform using ThingSpeak.

COMPONENTS REQUIRED :

Microcontroller with Wi-Fi capability (e.g., ESP8266, ESP32), Sensors (e.g., temperature sensor, humidity sensor), Internet connection, Computer with web browser, ThingSpeak account

THEORY :

Cloud Computing:

Cloud computing provides on-demand access to computing resources over the internet. It enables users to store and analyze data on remote servers rather than local machines.

IoT Cloud Platforms:

IoT cloud platforms offer services to connect, manage, and analyze data from Internet of Things (IoT) devices. ThingSpeak is one such platform that allows users to collect and visualize sensor data in real-time.

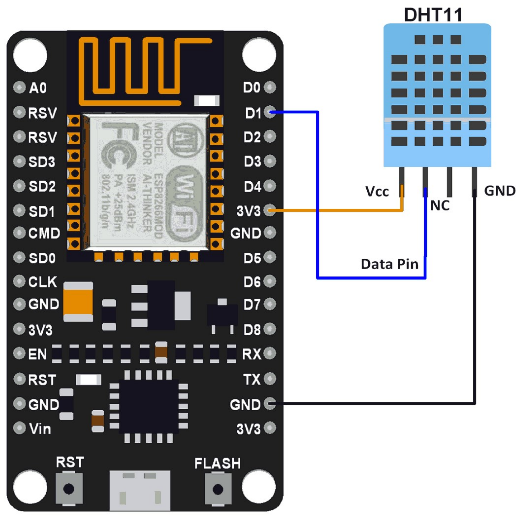
ThingSpeak:

ThingSpeak is an IoT analytics platform that enables users to collect, store, and analyze data from sensors and other IoT devices. It provides tools for data visualization and integration with other web services.

PROCEDURE :

1. Set up ThingSpeak account and channel.
2. Connect sensors to the microcontroller.
3. Write and upload program to publish data to ThingSpeak.
4. Verify data on ThingSpeak channel.
5. Observe and debug data transmission.

CIRCUIT DIAGRAM :



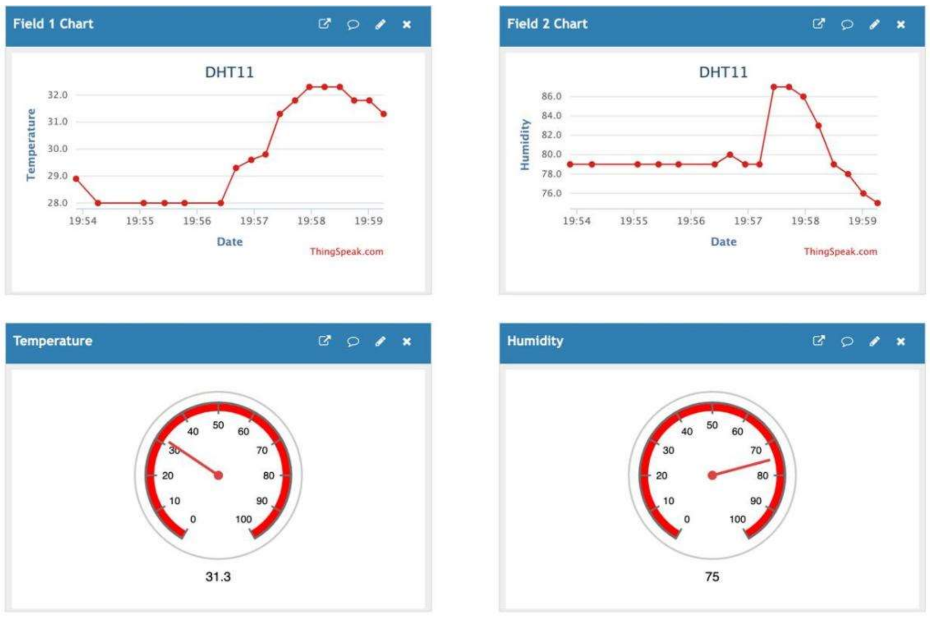
CODE:

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <ThingSpeak.h>

#define DHTPIN D2          // DHT11 data pin connected to D2 (GPIO4) on ESP8266
#define DHTTYPE DHT11      // Define sensor type as DHT11

// Wi-Fi credentials
const char* ssid = "Your_SSID";       // Replace with your Wi-Fi SSID
const char* password = "Your_Password";  // Replace with your Wi-Fi password

// ThingSpeak credentials
unsigned long myChannelNumber = 2749566;    // Replace with your ThingSpeak channel
ID
```

```
const char* myWriteAPIKey = "LS7NZHH13TLY0ED9";     // Replace with your
ThingSpeak Write API Key

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

WiFiClient client;   // Initialize the Wi-Fi client

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);  // Connect to Wi-Fi

  // Wait until connected
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");

  ThingSpeak.begin(client);  // Initialize ThingSpeak
  dht.begin();  // Initialize DHT11 sensor
}

void loop() {
  // Read temperature and humidity
  float temperature = dht.readTemperature();  // Read temperature in Celsius
  float humidity = dht.readHumidity();  // Read humidity

  // Check if any read failed and exit early
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Print the readings to the Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C ");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  // Update ThingSpeak with the new values
  ThingSpeak.setField(1, temperature);  // Set field 1 to temperature
  ThingSpeak.setField(2, humidity);  // Set field 2 to humidity

  // Write the data to ThingSpeak
  int responseCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);  //
Send data to ThingSpeak
```

```
  if (responseCode == 200) {
    Serial.println("Data sent successfully to ThingSpeak.");
  } else {
    Serial.println("Failed to send data to ThingSpeak. Response code: " +
String(responseCode));
  }

  // Wait for 20 seconds before sending the next data
  delay(2000);
}
```

OUTPUT:



INFERENCE :

DATE :	**Develop a system to publish sensor data to a cloud and control your sensor data using Thing speak and MIT APP Inventor**
EXP NO : 8	

AIM :

To develop a system that publishes sensor data to ThingSpeak and allows control and monitoring of the sensor data using an app created in MIT App Inventor.

COMPONENTS REQUIRED :

Microcontroller with Wi-Fi capability (e.g., ESP8266, ESP32), Sensors (e.g., temperature sensor, humidity sensor), Internet connection, ThingSpeak account, MIT App Inventor account, Smartphone or tablet

THEORY :

Microcontroller

A microcontroller is a compact, integrated circuit designed to perform specific tasks in embedded systems, incorporating a processor, memory, and I/O peripherals.

LDR Sensor

An LDR (Light Dependent Resistor) sensor changes its resistance based on the intensity of light, commonly used in light-sensing applications like automatic lighting.

Relay

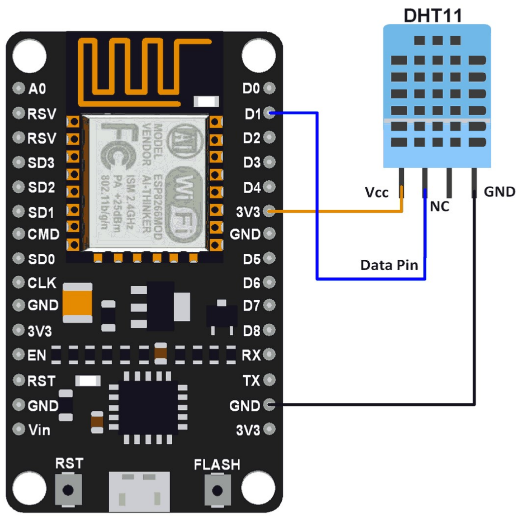
A relay is an electrically operated switch that uses a small input current to control a larger output current, allowing low-power signals to control high-power devices.

PROCEDURE :

1. Set up ThingSpeak account and channel.
2. Connect sensors to the microcontroller.

3. Program the microcontroller to publish data to ThingSpeak.
4. Create and design the app in MIT App Inventor.
5. Integrate ThingSpeak API with the app.
6. Upload and test the microcontroller program.
7. Test and debug the app with ThingSpeak.

CIRCUIT DIAGRAM :

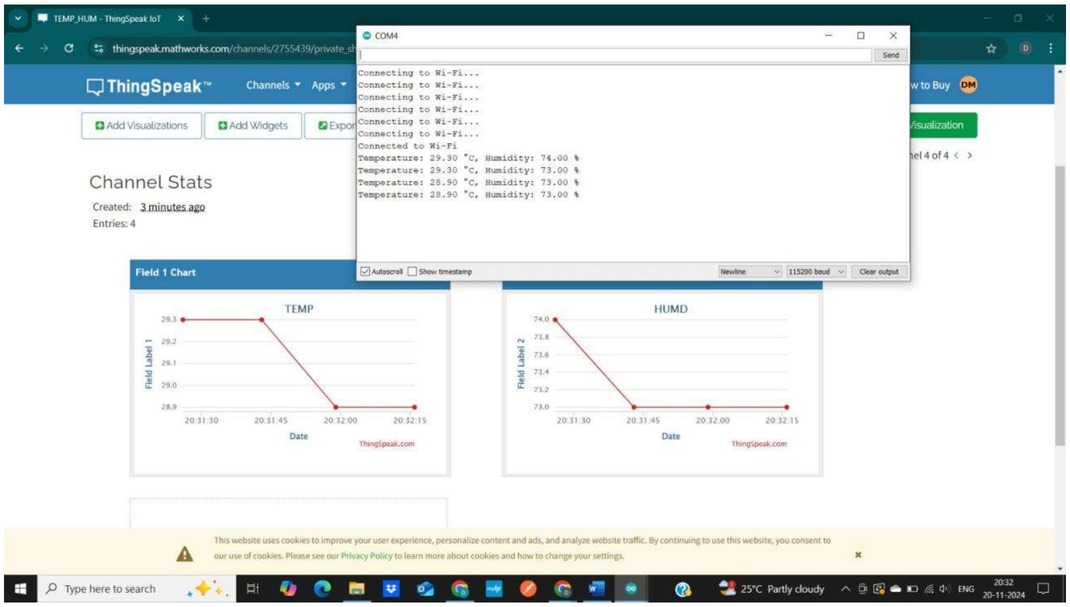


CODE:

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#define DHTPIN D4
#define DHTTYPE DHT11
const char* ssid = "VivoT1";
const char* password = "mahizhan";
const char* server = "api.thingspeak.com";
String apiKey = "OO7YE2L120QJ4E7L";
WiFiClient client;
DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to Wi-Fi...");
  }
  Serial.println("Connected to Wi-Fi");
  dht.begin();
}
void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C, Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");
  if (WiFi.status() == WL_CONNECTED) {
    if (client.connect(server, 80)) {
      String url = "/update?api_key=" + apiKey + "&field1=" + String(temperature) +
"&field2=" + String(humidity);
      client.print("GET " + url + " HTTP/1.1\r\n");
      client.print("Host: " + String(server) + "\r\n");
      client.print("Connection: close\r\n\r\n");
      delay(1000);
    } else {
      Serial.println("Connection failed");
    }
    client.stop();
  } else {
    Serial.println("Wi-Fi not connected");
  }
  delay(15000);
}
```

OUTPUT:



```
Connecting to Wi-Fi...
Connecting to Wi-Fi...
Connecting to Wi-Fi...
Connecting to Wi-Fi...
Connecting to Wi-Fi...
Connecting to Wi-Fi...
Connected to Wi-Fi
Temperature: 29.30 °C, Humidity: 74.00 %
Temperature: 29.30 °C, Humidity: 73.00 %
Temperature: 28.90 °C, Humidity: 73.00 %
Temperature: 28.90 °C, Humidity: 73.00 %
```

INFERENCE:

DATE :	**Develop a system security system and send Email notifications, and app alerts using Blynk cloud**
EXP NO : 9	

AIM :

To develop a security system that sends email notifications and app alerts using Blynk Cloud.

COMPONENTS REQUIRED :

Microcontroller (e.g., ESP8266, ESP32), Security sensors (e.g., motion sensor, door/window contact), Blynk account, Email service setup (e.g., SMTP server), Smartphone or tablet

THEORY :

Microcontroller

A microcontroller is a small, integrated circuit designed to execute specific tasks in embedded systems, containing a processor, memory, and I/O peripherals.

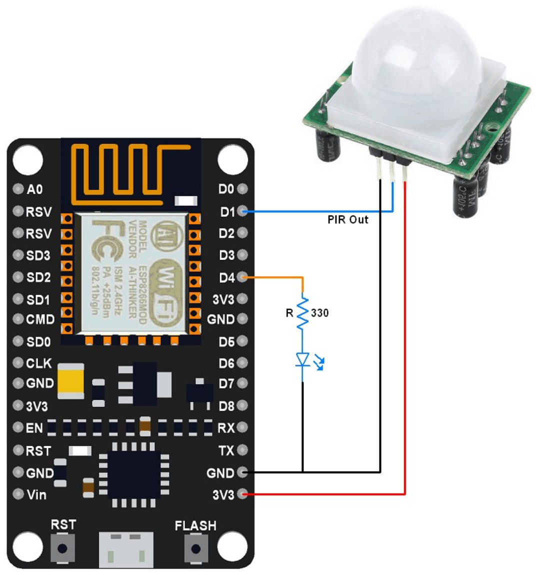
PIR Sensor

A PIR (Passive Infrared) sensor detects motion by measuring infrared radiation changes, commonly used in security systems and automatic lighting.

PROCEDURE :

1. Set up Blynk Cloud account and project.
2. Connect security sensors to the microcontroller.
3. Program the microcontroller for sensor data and Blynk integration.
4. Configure email notifications.
5. Connect microcontroller to Blynk Cloud.
6. Test the system for alerts and notifications.

CIRCUIT DIAGRAM :



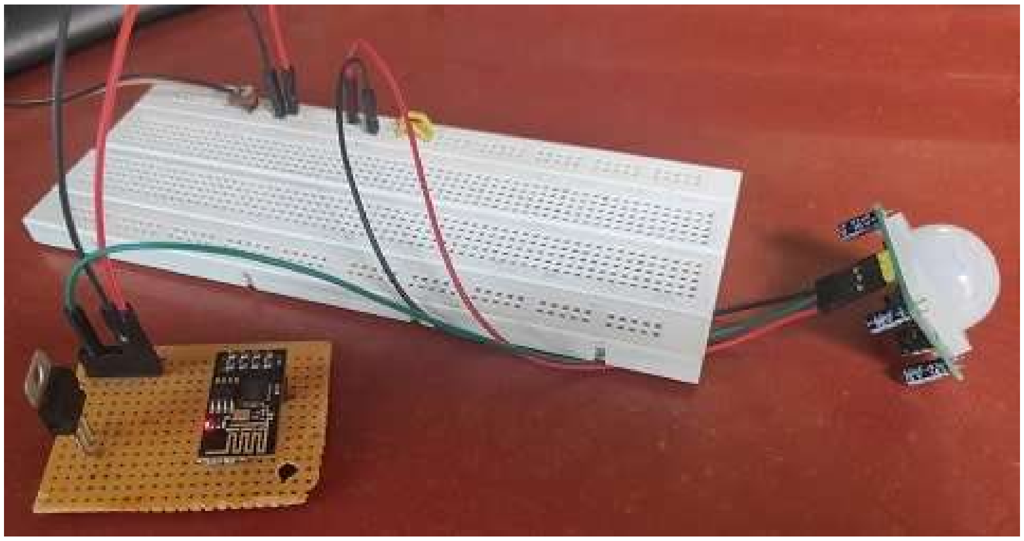
CODE:

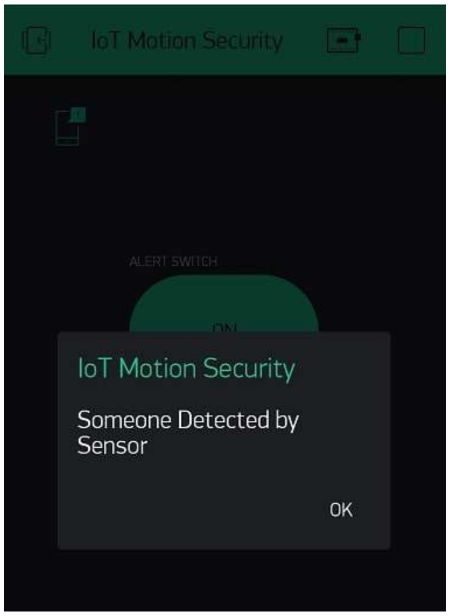
```
#define BLYNK_TEMPLATE_ID "TMPLp_USLgRB"
#define BLYNK_TEMPLATE_NAME "Agri Monitoring"
#define                                      BLYNK_AUTH_TOKEN
"Uw9vmJGf4vvq7Lk4TLjdIcsZoKNYMiBT"
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = BLYNK_AUTH_TOKEN;
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "iotlab";
char pass[] = "password";
```

```
#define DHTPIN D2          // D3
// Uncomment whatever type you're using!
#define DHTTYPE DHT11     // DHT 11
//#define DHTTYPE DHT22   // DHT 22, AM2302, AM2321
//#define DHTTYPE DHT21   // DHT 21, AM2301
DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;
// This function sends Arduino's up time every second to Virtual Pin (5).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.
BLYNK_WRITE(V0)
{
  int value = param.asInt();
  Serial.println(value);
  if(value == 1)
  {
    digitalWrite(D4, HIGH);
    Serial.println("LED ON");
  }
  if(value == 0)
  {
     digitalWrite(D4, LOW);
     Serial.println("LED OFF");
  }
}
void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for
Fahrenheit
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V5, t);
  Blynk.virtualWrite(V6, h);
 // Serial.println("TEMPERATURE=");
//  Serial.print(t);
//  Serial.println("HUMIDITY=");
//  Serial.print(h);
if(t > 34){
   //  Blynk.email("shameer50@gmail.com", "Alert", "Temperature over
28C!");
 //   Blynk.logEvent("temp_alert","Temp above 34 degree");
  }
}
  WidgetLCD lcd(V1);
//int pirValue; // Place to store read PIR Value
void setup()
{
  // Debug console
  Serial.begin(9600);
  pinMode(D4, OUTPUT);
   pinMode(D0, OUTPUT);
  pinMode(D1, INPUT);
  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);
  dht.begin();
  // Setup a function to be called every second
  timer.setInterval(1000L, sendSensor);
}
void loop()
{
  Blynk.run();
  timer.run();
     lcd.print(0, 0, "SMART Home  ");
    lcd.print(0, 1, " Security S/M ");
     delay(2000);
    air1();
     lcd.clear();
}
void air1(void)
{
  if (digitalRead(D1)==LOW)     // PIR sensor / door sensor
   {
    digitalWrite(D0, HIGH);
   Blynk.logEvent("vibration_alert","theft detected");
     Blynk.email("ramece74@gmail.com", "Alert", "Theft Alert in Home");
    lcd.clear();
     lcd.print(0, 0, "THEFT ALERT");
     delay(1000);
   }
```

```
 else
 {
 lcd.clear();
 lcd.print(0, 0, "HOME NORMAL       ");
 digitalWrite(D0, LOW);

   delay(1000);
 }
}
```

OUTPUT:





IoT Motion Security

Someone Detected by Sensor

OK

INFERENCE :

DATE :	**Design and develop NFC based Home security system Using the Blynk app/ Thingspeak.**
EXP NO : 10	

AIM :

To design and develop an NFC-based home security system that integrates with Blynk app or ThingSpeak for remote monitoring and control.

COMPONENTS REQUIRED :

Microcontroller with Wi-Fi capability (e.g., ESP8266, ESP32), NFC reader module, NFC tags/cards, Security sensors (optional, e.g., motion sensors), Blynk account or ThingSpeak account, Smartphone with Blynk app (if using Blynk) or access to ThingSpeak (if using ThingSpeak)

THEORY :

NFC (Near Field Communication):

A short-range wireless communication technology that allows devices to exchange data when in close proximity. NFC can be used for secure authentication and access control.

Blynk App:

A mobile application for creating IoT applications. It provides cloud connectivity and user interface elements to interact with hardware.

ThingSpeak:

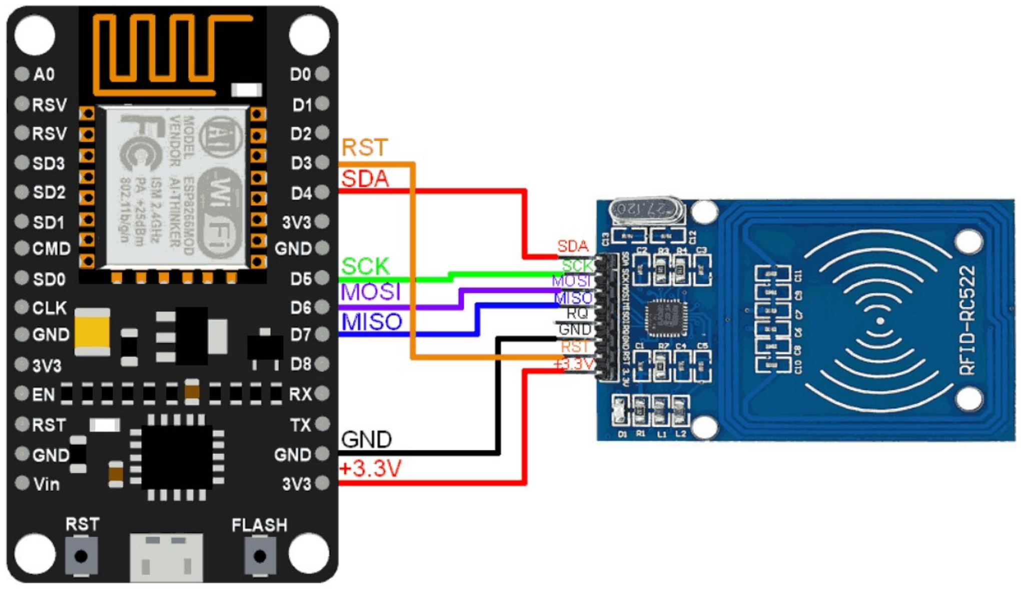
An IoT platform for data collection, storage, and analysis. It provides tools for monitoring and visualizing sensor data.

PROCEDURE :

1. Set up NFC reader and tags.
2. Set up Blynk or ThingSpeak account and project/channel.
3. Program the microcontroller for NFC and cloud integration.
4. Connect microcontroller to Blynk or ThingSpeak.

5. Test the NFC-based security system.

CIRCUIT DIAGRAM :

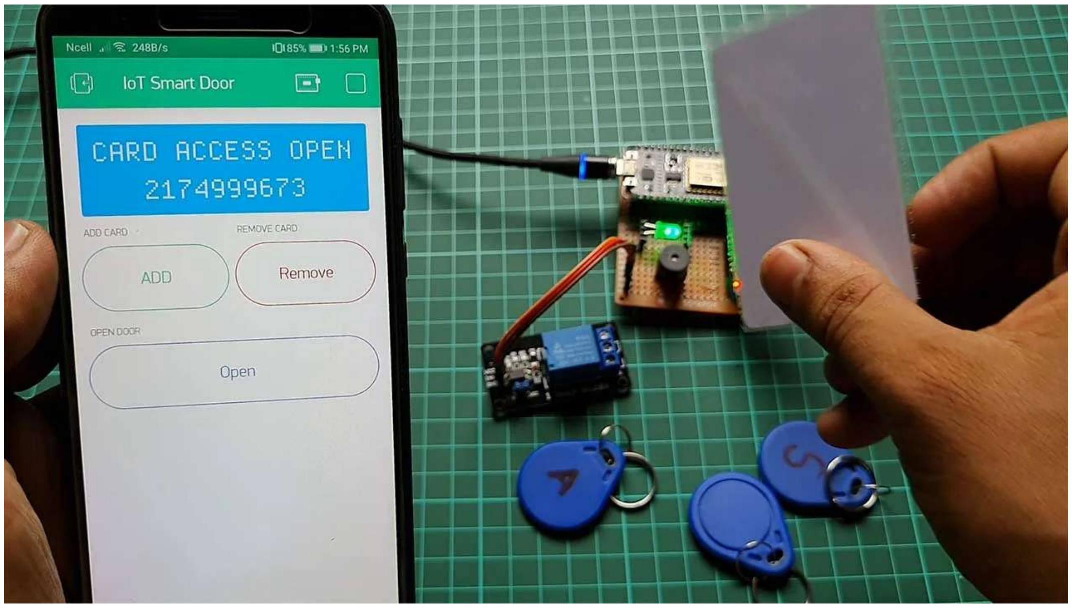


CODE:

```
#define BLYNK_TEMPLATE_ID "TMPLp_USLgRB"
#define BLYNK_TEMPLATE_NAME "Agri Monitoring"
#define BLYNK_AUTH_TOKEN "Uw9vmJGf4vvq7Lk4TLjdIcsZoKNYMiBT"
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
SoftwareSerial rfid(2,3);
String m="";
String st1="";
String data1="400031B20FCC";
String data2="400031EF64FA";
String data3="400031F28E0D";
String data4="400031E649DE";
int sc1=0;
int s1=0;
int d1=0;
int c=0;
int flg=0;
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = BLYNK_AUTH_TOKEN;
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "iotlab";
char pass[] = "password";
#define DHTPIN D2          // D3
// Uncomment whatever type you're using!
#define DHTTYPE DHT11     // DHT 11
//#define DHTTYPE DHT22   // DHT 22, AM2302, AM2321
//#define DHTTYPE DHT21   // DHT 21, AM2301
DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;
// This function sends Arduino's up time every second to Virtual Pin (5).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.
BLYNK_WRITE(V0)
{
  int value = param.asInt();
  Serial.println(value);
  if(value == 1)
  {
    digitalWrite(D4, HIGH);
    Serial.println("LED ON");
  }
  if(value == 0)
  {
     digitalWrite(D4, LOW);
     Serial.println("LED OFF");
  }
```

```
}
void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V5, t);
  Blynk.virtualWrite(V6, h);
 // Serial.println("TEMPERATURE=");
//  Serial.print(t);
//  Serial.println("HUMIDITY=");
//  Serial.print(h);
if(t > 34){
   // Blynk.email("shameer50@gmail.com", "Alert", "Temperature over 28C!");
 //   Blynk.logEvent("temp_alert","Temp above 34 degree");
  }
}
  WidgetLCD lcd(V1);
//int pirValue; // Place to store read PIR Value
void setup()
{
lcd.begin(16,2);
  rfid.begin(9600); // Setting the baud rate of Software Serial Library
  Serial.begin(9600);  //Setting the baud rate of Serial Monitor
  lcd.setCursor(0,0);
  lcd.print("  RFID BASED   ");
  lcd.setCursor(0,1);
  lcd.print("Authentication");
  delay(2000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("   Waiting   ");
 delay(2000);
  // Debug console
  Serial.begin(9600);
  pinMode(D4, OUTPUT);
   pinMode(D0, OUTPUT);
  pinMode(D1, INPUT);
  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);
  dht.begin();
  // Setup a function to be called every second
  timer.setInterval(1000L, sendSensor);
}
void loop()
{
 lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("   Waiting   ");
  readcard();
 delay(1000);
  Blynk.run();
  timer.run();
     lcd.print(0, 0, "SMART Home  ");
    lcd.print(0, 1, " Security S/M ");
     delay(2000);
    air1();
     lcd.clear();
}
void air1(void)
{
  if (digitalRead(D1)==LOW)     // PIR sensor / door sensor
  {
    digitalWrite(D0, HIGH);
   Blynk.logEvent("vibration_alert","theft detected");
     Blynk.email("ramece74@gmail.com", "Alert", "Theft Alert in Home");
    lcd.clear();
     lcd.print(0, 0, "THEFT ALERT");
     delay(1000);
  }
  else
  {
  lcd.clear();
  lcd.print(0, 0, "HOME NORMAL       ");
  digitalWrite(D0, LOW);
    delay(1000);
  }
}
void readcard()
{
  while(rfid.available())
  {
    char n=rfid.read();
    if((n!=13)&&(n!=10))
    m+=n;
    //res+=n-48;
    //Serial.print(n);
    c+=1;
delay(50);
    }
 if(c>11)
 {
  Serial.print("m value=");
  Serial.println(m);
  if(((String)data1)==((String)m))
{
 Serial.println("matched");
 Serial.println(data1);
 }
 else if(((String)data2)==((String)m))
{
 Serial.println("matched");
 Serial.println(data2);
 }
 else if(((String)data3)==((String)m)) {
 Serial.println("matched");
 Serial.println(data3);
 }
 else if(((String)data4)==((String)m)) {
 Serial.println("matched");
 Serial.println(data4);
 }
 else {
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Un Authorized");
   lcd.setCursor(0,1);
   lcd.print("   Ticket");
  digitalWrite(A5,HIGH);
  delay(2000);
  digitalWrite(A5,LOW);
    lcd.clear();
   }
c=0;
m="";
delay(100);
}
}
```

OUTPUT:



INFERENCE :