IoT Internship Task 2 Report

Smart Environment Monitoring System using ESP32, DHT11, Sound Sensor, LCD, and ThingSpeak

1. Introduction and Functional Requirements

The Smart Environment Monitoring System is an IoT-based solution designed to monitor temperature, humidity, and ambient noise levels in real-time using ESP32. This system utilizes the DHT11 sensor for environmental readings, a sound sensor for detecting noise levels, an LCD to display the data locally, and ThingSpeak to upload and visualize the sensor data remotely.
Functional Requirements:
- Monitor temperature and humidity continuously.
- Measure ambient sound levels.
- Display data on an LCD.
- Send data to ThingSpeak IoT cloud platform every 15 seconds.
- Connect via Wi-Fi for wireless data transmission.

2. Resources Used

Hardware:
- ESP32 Development Board
- DHT11 Temperature and Humidity Sensor
- Sound Sensor Module
- 16x2 I2C LCD Display
- Breadboard and Jumper Wires

Software:
- Arduino IDE
- ThingSpeak IoT Platform
- ESP32 Board Library
- DHT Library
- LiquidCrystal_I2C Library

3. System Architecture Diagram

```
[DHT11]       [Sound Sensor]

   \            /

    \          /

     --> [ESP32] <--- Wi-Fi ---> [ThingSpeak Cloud]

           |

           v

        [LCD Display]
```

architecture diagram image here showing ESP32 connected to DHT11, Sound Sensor, LCD, and Wi-Fi to ThingSpeak.

4. Communication Protocols Used

- I2C: Used for communication between ESP32 and 16x2 LCD display.
- UART (Serial): For debugging and monitoring data using the Serial Monitor.
- HTTP: To send sensor data to ThingSpeak over the internet using GET requests.

5. Arduino Code

The following code was used to read sensor data, display it on the LCD, and send it to ThingSpeak:

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

const char* ssid = "AKASH";
const char* password = "90909090";
const char* server = "http://api.thingspeak.com/update";
String apiKey = "XCL1VW0OMBXZH84B";

#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

#define SOUND_SENSOR_PIN 34
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
 Serial.begin(115200);
 lcd.init(); lcd.backlight();
 lcd.setCursor(0, 0); lcd.print("Initializing...");
 dht.begin();
 WiFi.begin(ssid, password);
 lcd.setCursor(0, 1); lcd.print("Connecting WiFi");
 while (WiFi.status() != WL_CONNECTED) {
  delay(1000); Serial.println("Connecting to WiFi...");
 }
 lcd.clear(); lcd.setCursor(0, 0); lcd.print("WiFi Connected");
 delay(2000);
}

void loop() {
 float h = dht.readHumidity();
 float t = dht.readTemperature();
 int soundValue = analogRead(SOUND_SENSOR_PIN);

 if (isnan(h) || isnan(t)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
 }

 Serial.print("Temp: "); Serial.print(t);
 Serial.print(" °C  Humidity: "); Serial.print(h);
 Serial.print(" %  Sound: "); Serial.println(soundValue);

 lcd.clear();
 lcd.setCursor(0, 0); lcd.print("T:"); lcd.print(t); lcd.print("C H:"); lcd.print(h);
 lcd.setCursor(0, 1); lcd.print("Sound:"); lcd.print(soundValue);

 if (WiFi.status() == WL_CONNECTED) {
  HTTPClient http;
  String url = server;
  url += "?api_key=" + apiKey;
  url += "&field1=" + String(t);
  url += "&field2=" + String(h);
  url += "&field3=" + String(soundValue);

  http.begin(url);
  int httpResponseCode = http.GET();
  Serial.println(httpResponseCode > 0 ? "Data sent." : "Error sending data.");
  http.end();
 }
 delay(15000);
}
```

Code explanation:

The code starts by including necessary libraries for WiFi, HTTP communication, DHT11 sensor, and I2C LCD. It defines the sensor pins and initializes the DHT11 sensor and the LCD display. In the setup function, serial communication is started, sensors and LCD are initialized, and the ESP32 connects to the specified WiFi network. The main loop reads temperature and humidity data from the DHT11 sensor and analog sound level from the sound sensor. These values are displayed on the LCD with temperature and humidity on the first row and sound level on the second row. The data is then formatted into a URL string and sent to the ThingSpeak cloud server via an HTTP GET request. The code checks the HTTP response to confirm if the data upload was successful. Finally, a delay of 15 seconds is added to comply with ThingSpeak's data update interval. This process repeats continuously to monitor and send sensor data in real-time.

6. Testing and Snapshots

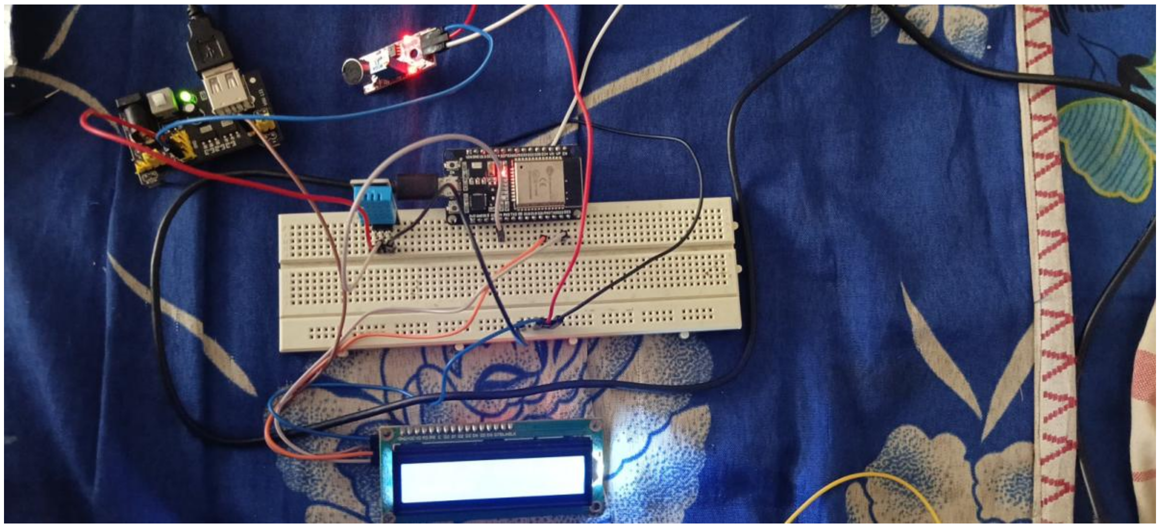
DHT11 Pin	**Connect To ESP32**
VCC	3.3V
GND	GND
DATA	GPIO 4

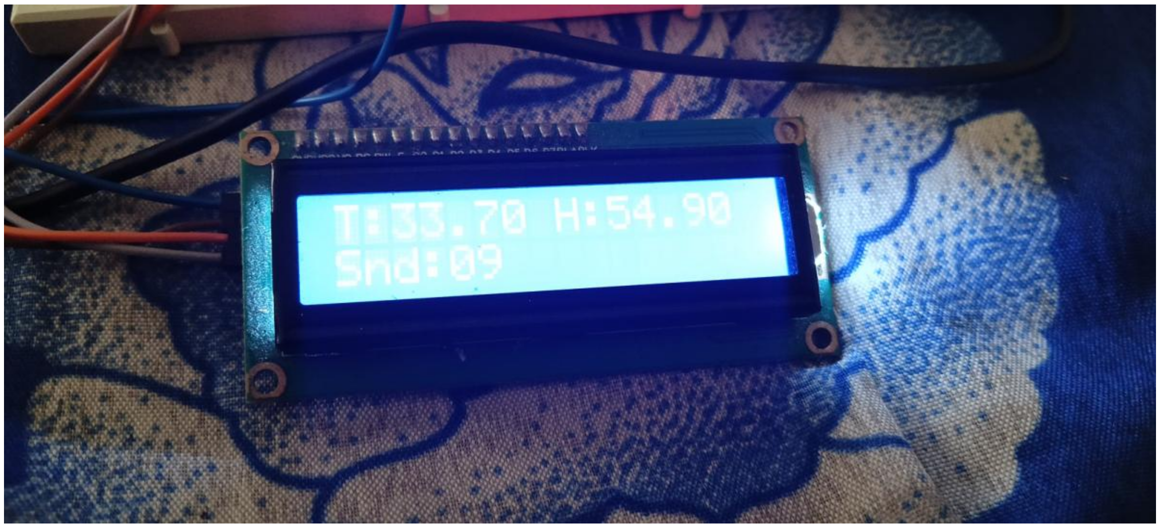
DHT11 Pin	Connect To ESP32

Sound Sensor Pin	Connect To ESP32
VCC	3.3V
GND	GND
OUT	GPIO 34 (A2)

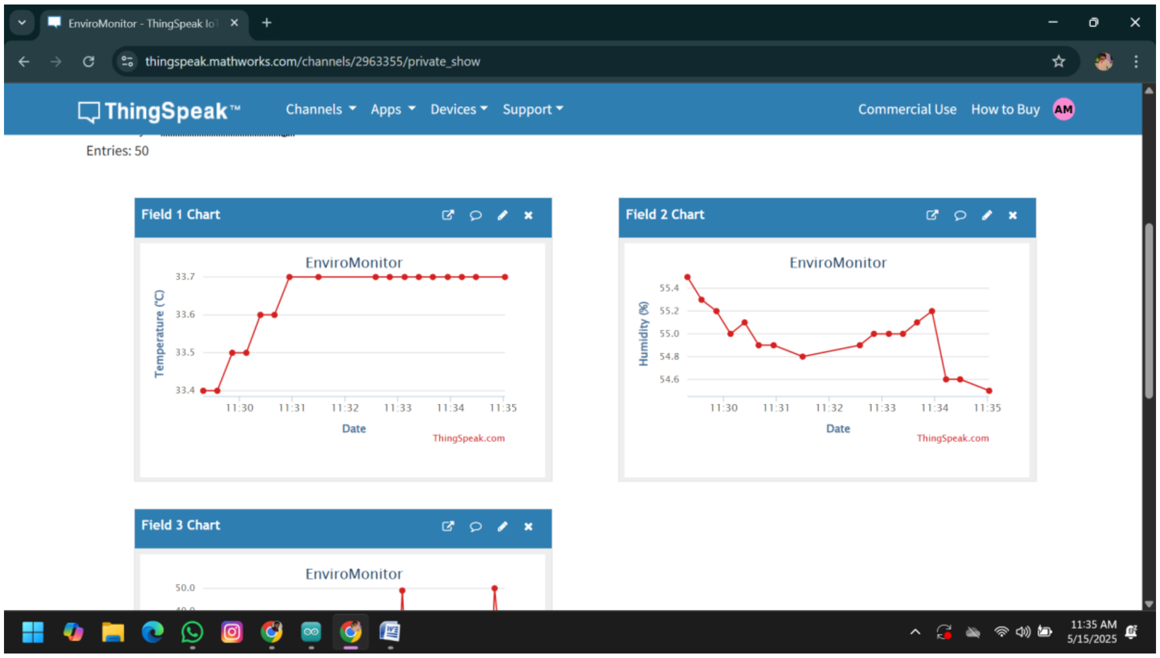
LCD Pin	Connect To ESP32
VCC	5V (or 3.3V*)
GND	GND
SDA	GPIO 21
SCL	GPIO 22

Physical connections

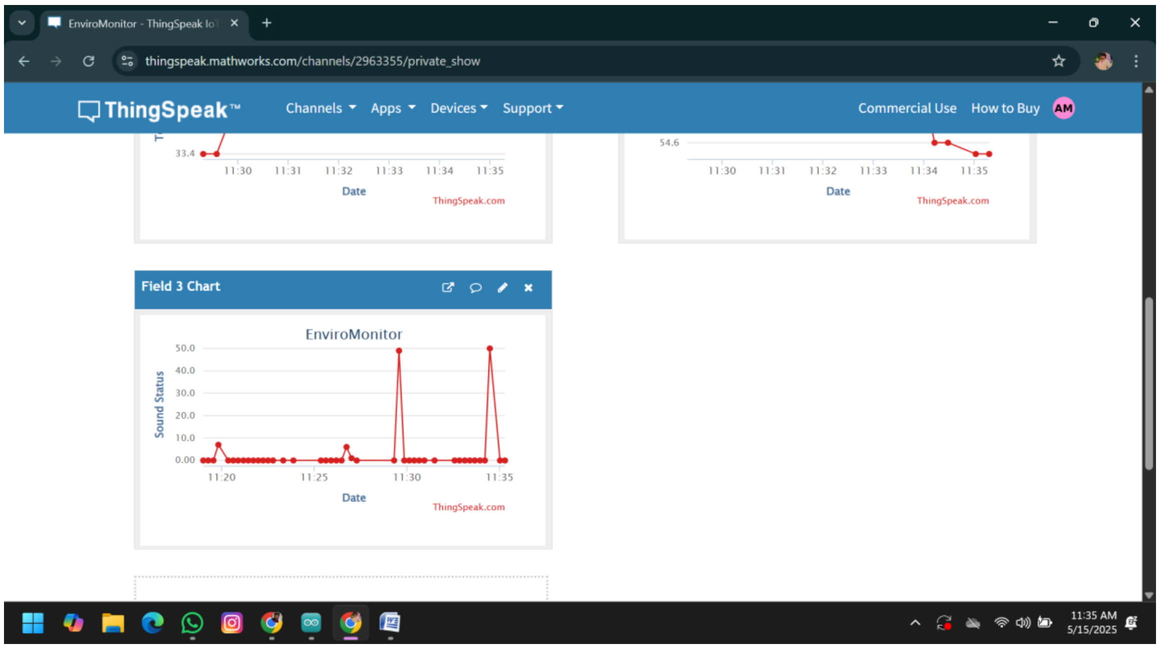




Thingspeak output



Entries: 50



7. Discussion on Issues and Privacy/Security

Issues Faced:
- Sensor read failure due to improper connections.
- Wi-Fi dropouts during data transmission.

Security Considerations:
- The system lacks encryption for HTTP requests.
- Adding HTTPS and authentication could improve privacy.
- ThingSpeak channel privacy should be configured to avoid unauthorized access.

8. References

- Arduino (www.arduino.cc)
- ThingSpeak Documentation (https://thingspeak.com)
- DHT11 Sensor Datasheet
- ESP32 Developer Resources

9.vodcast

Here drive link to see my video explanation:

https://drive.google.com/file/d/1KT6DNob-RdQvFyElUQpH_0X_blpWtPdy/view?usp=sharing