Project Report: ESP32-based Environmental Monitoring System with ThingSpeak

Author: (Udaykuamar Kota) **Date:** September 28, 2025

---

1. Abstract

This project details the design and implementation of an IoT-based environmental monitoring system. The system utilizes an ESP32 microcontroller to capture real-time temperature and humidity data from a DHT11 sensor. The collected data is simultaneously displayed on a local 16x2 I2C LCD for immediate viewing and transmitted over WiFi to the ThingSpeak IoT platform for remote logging, visualization, and analysis. This solution provides a simple yet effective way to monitor environmental conditions from anywhere with an internet connection.

---

2. Components and Software

Hardware Components

- ESP32 DevKit V1 Microcontroller
- DHT11 Temperature and Humidity Sensor
- 16x2 LCD Display with I2C Module
- Breadboard
- Jumper Wires

Software and Libraries

- **Arduino IDE:** For programming the ESP32.
- **ThingSpeak:** IoT analytics platform for data logging.
- **Required Libraries:**
    - `WiFi.h`: For ESP32 WiFi connectivity.
    - `ThingSpeak.h`: For communication with the ThingSpeak API.
    - `DHT.h`: For interfacing with the DHT11 sensor.
    - `LiquidCrystal_I2C.h`: For controlling the I2C LCD.

---

3. Circuit Diagram and Connections

The components are wired to the ESP32, which acts as the central controller. The I2C protocol is used for the LCD to minimize pin usage, while the DHT11 communicates over a single digital pin.

Connection Table

Component	Pin	Connects to ESP32 Pin
I2C LCD	GND	**GND**
	VCC	**VIN (5V)**
	SDA	**GPIO 21**
	SCL	**GPIO 22**
DHT11 Sensor	GND (-)	**GND**
	VCC (+)	**VIN (5V)**
	DATA (OUT)	**GPIO 4**

Export to Sheets

---

4. Working Principle

The system operates in a continuous loop with the following steps:

1. **Initialization:** On startup, the ESP32 initializes the DHT11 sensor, the I2C LCD, and the Serial Monitor for debugging.
2. **WiFi Connection:** The device connects to the predefined WiFi network. Connection status is displayed on the LCD.
3. **Data Acquisition:** Every 20 seconds, the ESP32 polls the DHT11 sensor to get the current temperature (in Celsius) and relative humidity (in %).
4. **Local Display:** The acquired temperature and humidity values are immediately formatted and displayed on the 16x2 LCD screen.
5. **Data Transmission:** The same values are sent to a designated ThingSpeak channel using an HTTP request, authenticated with a unique Write API Key.
6. **Cloud Visualization:** ThingSpeak receives the data, timestamps it, and plots it on charts. This allows for historical data analysis and remote monitoring from any web browser or the ThingSpeak mobile app.

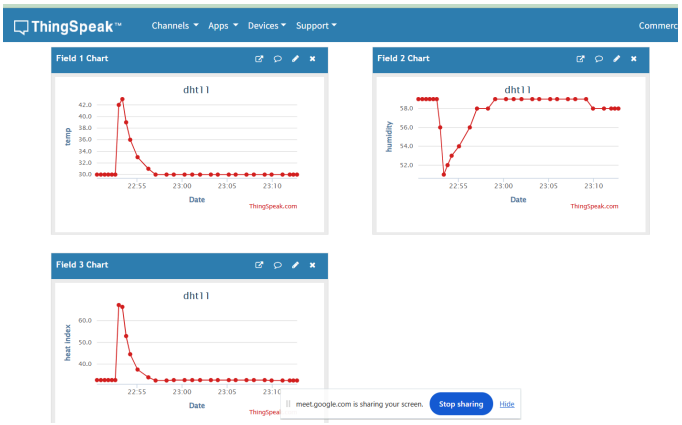
---

5. Results and Conclusion

The system was successfully deployed, and the data was transmitted to the ThingSpeak platform as intended. The screenshot below shows the ThingSpeak dashboard visualizing the data received from the ESP32.

The charts clearly display the logged values for **Field 1 (Temperature)** and **Field 2 (Humidity)** over time. The third chart, **Field 3 (Heat Index)**, is likely a visualization created within ThingSpeak itself by analyzing the incoming temperature and humidity data. This confirms that the hardware is correctly wired, the code is functioning as expected, and the cloud integration is successful.

In conclusion, this project serves as a successful proof-of-concept for a basic IoT monitoring station. It can be easily expanded by adding more sensors or integrating alerts and notifications through ThingSpeak's features.



---

6. Full Arduino Code

C++

```
#include <WiFi.h>
#include "ThingSpeak.h"
#include <LiquidCrystal_I2C.h>
#include "DHT.h"

// ---------- WiFi and ThingSpeak Credentials ----------
const char* ssid = "YOUR_WIFI_SSID";         // Your WiFi network name
const char* password = "YOUR_WIFI_PASSWORD"; // Your WiFi password

unsigned long myChannelNumber = 1234567; // CHANGE THIS to your Channel ID
const char* myWriteAPIKey = "2U856PX49VJK4FYI"; // Your ThingSpeak Write API Key

// ---------- Hardware Pin and I2C Address Definitions ----------
#define DHTPIN 4        // The GPIO pin connected to the DHT11's DATA pin
#define DHTTYPE DHT11   // We are using the DHT11 sensor

// Set the LCD address. Common addresses are 0x27 or 0x3F.
int lcdAddress = 0x27;
```

```
int lcdColumns = 16;
int lcdRows = 2;

// ---------- Object Initialization ----------
WiFiClient client;
LiquidCrystal_I2C lcd(lcdAddress, lcdColumns, lcdRows);
DHT dht(DHTPIN, DHTTYPE);

// ---------- Global Variables ----------
unsigned long lastUpdateTime = 0;
const int updateInterval = 20000; // ThingSpeak free plan allows updates every ~15 seconds.
20s is safe.

// ================================================================
// SETUP FUNCTION
// ================================================================
void setup() {
  Serial.begin(115200);

  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Initializing...");

  // Initialize DHT Sensor
  dht.begin();

  // Connect to WiFi
  connectWiFi();

  // Initialize ThingSpeak
  ThingSpeak.begin(client);
}

// ================================================================
// MAIN LOOP
// ================================================================
void loop() {
  // The loop() function should run quickly. We use millis() for timing.
  if (millis() - lastUpdateTime > updateInterval) {
    readSensorAndUpload();
    lastUpdateTime = millis();
  }
}

// ================================================================
// SUPPORTING FUNCTIONS
```

```
// ============================================================

void connectWiFi() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Connecting WiFi");
  Serial.println("Connecting to WiFi...");

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  int counter = 0;
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    lcd.setCursor(counter, 1);
    lcd.print(".");
    counter++;
    if (counter > 15) { // If it takes too long, reset the dots on the LCD
       lcd.setCursor(0, 1);
       lcd.print("               ");
       counter = 0;
    }
  }

  Serial.println("\nWiFi Connected!");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("WiFi Connected!");
  delay(1000);
}

void readSensorAndUpload() {
  // Read Humidity and Temperature from DHT11
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature(); // Celsius

  // Check if any reads failed and exit early to try again.
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Sensor Read");
    lcd.setCursor(0, 1);
    lcd.print("Error!");
    return;
  }
```

```
  // --- Print to Serial Monitor for debugging ---
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" *C");

  // --- Display on LCD ---
  lcd.clear();
  // Line 0: Temperature
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
  lcd.print((char)223); // Degree symbol
  lcd.print("C");
  // Line 1: Humidity
  lcd.setCursor(0, 1);
  lcd.print("Humi: ");
  lcd.print(humidity);
  lcd.print(" %");

  // --- Send data to ThingSpeak ---
  ThingSpeak.setField(1, temperature);
  ThingSpeak.setField(2, humidity);

  int httpCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

  if (httpCode == 200) {
    Serial.println("Channel update successful.");
  } else {
    Serial.println("Problem updating channel. HTTP error code " + String(httpCode));
  }
}
```