An IoT-Based Real-Time Weather Monitoring System

Riya Vardhan
Department of Electronics and Communication,
MIT Manipal
Section: B, Roll No: 16
Registration No: 200907110

Thaejus Surya M T
Department of Electronics and Communication,
MIT Manipal
Section: B, Roll No: 14
Registration No: 200907102

Sagnnik Biswas
Department of Electronics and Communication,
MIT Manipal
Section: B, Roll No: 13
Registration No: 200907098

***Abstract* — This project focuses on real-time monitoring of ambient temperature, humidity, and sunlight intensity using a weather monitoring system. It explores the practical uses of Internet of Things (IoT) and its potential applications, particularly in the context of the MQTT protocol and the I2C protocol.**

Keywords — NodeMCU, ESP8266 module, MQTT & I2C

I. Introduction

This is a prototype of a contemporary weather monitoring system that leverages IoT technology for processing information. It provides users with real-time updates on the ambient temperature, humidity levels, and sunlight intensity in their local area. The data is gathered through temperature sensors, humidity sensors, and LDRs and transmitted to ThingSpeak, a cloud-based IoT platform, for monitoring and analysis. Users can access the refined data on the platform or through an LCD display. The entire system is controlled by NodeMCU, which is equipped with an ESP8266 wifi module.

II. Components

The following are the significant components of this electronic project:

A. Hardware Components

- **NodeMCU ESP2866 Development Board**, which is an open-source development board based on the ESP8266 Wi-Fi Chip. It provides the necessary hardware and software to connect the sensors to the internet and send data to a cloud-based platform like ThingSpeak.
- **DHT 11 Sensor**, to read the humidity and temperature of the current environment of the user.
- **LDR Sensor**, to detect and read the light intensity of the user environment.
- **BMP180 Sensor**, to read the barometric or atmospheric pressure of the user environment.
- **Resistors**
- **LCD**
- **LEDs**
- **Breadboard**

B. Software Components

- **TinkerCAD**, for the software simulation of the circuit.
- **Arduino**, to write and load code to made the hardware operable.
- **ThingSpeak Platform**, to record the user's current environment conditions using all the weather parameters and also providing a graphical interface to monitor their weather conditions over a period of time.

III. Working

The NodeMCU is used as the main processor. Each sensor reads its own weather information and sends it to NodeMCU. An integrated WiFi system-on-chip (SoC) called the ESP8266 may connect to a cloud platform, ThingSpeak in this case, to save data for later processing.

The project's IoT cloud platform, called ThingSpeak, enables us to gather, visualize, and analyze real-time data from connected devices. It uses the MQTT protocol to operate. A machine-to-machine internet of things networking protocol is called MQTT, or Machine Queueing Telemetry Transport. Clients can publish and subscribe to the messages they want to send and receive using a publish and subscribe mechanism.

To display real-time weather information, the sensor data is also shown on an LCD. The LCD and NodeMCU communicate with each other via the I2C protocol.

A. MQTT Protocol

A lightweight messaging protocol called MQTT (Message Queuing Telemetry Transport) is frequently used in IoT devices to make it easier for various IoT system components to communicate with one another. MQTT is intended to be straightforward, effective, and dependable, which makes it perfect for Internet of Things applications when network bandwidth and battery life are constrained.

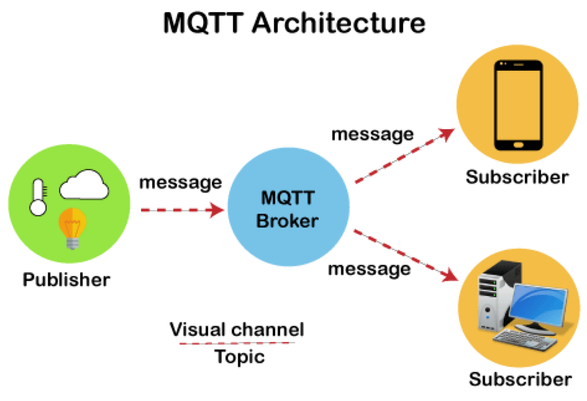


Fig 1. The Publisher-Subscriber Model of the MQTT Architecture

The publisher-subscriber model, the broker, and the client are the three main parts of the MQTT architecture.

1. **Client:** An IoT device that is linked to a network and desires to exchange messages with other devices is known as a MQTT client. In order to receive messages from other devices, a client can subscribe to a topic or publish messages to it. With the help of TCP/IP or another network protocol, clients can communicate with a broker.
2. **MQTT broker:** An MQTT broker is a server that receives messages from MQTT clients and sends them to other clients in accordance with the topics to which they are subscribed. The broker is in charge of overseeing message delivery and routing in the MQTT network. For customers who are disconnected, the broker can store messages for them and make sure they are delivered when the client reconnects. The choice of MQTT Broker in this project is the ThingSpeak cloud platform.
3. **Publisher-Subscriber Model:** The MQTT protocol employs a publisher-subscriber paradigm, in which a publisher posts messages to a topic, and a broker then forwards those messages to each subscriber who has signed up for that topic. Regardless of whatever client published the message, subscribers receive messages from the subjects they are subscribed to.

IV. Block Diagram

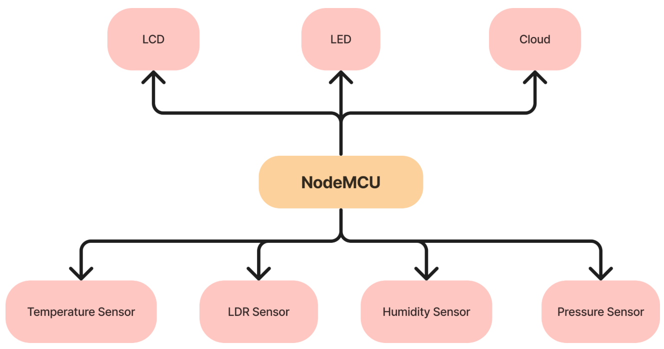


Fig 2. Rough Block Diagram of the Hardware components and their connection to cloud (ThingSpeak)

V. Software Simulation

Before proceeding with the hardware assembling of the project, the circuit was simulated using the TinkerCAD Software in order to provide a rough idea of the project complexity.

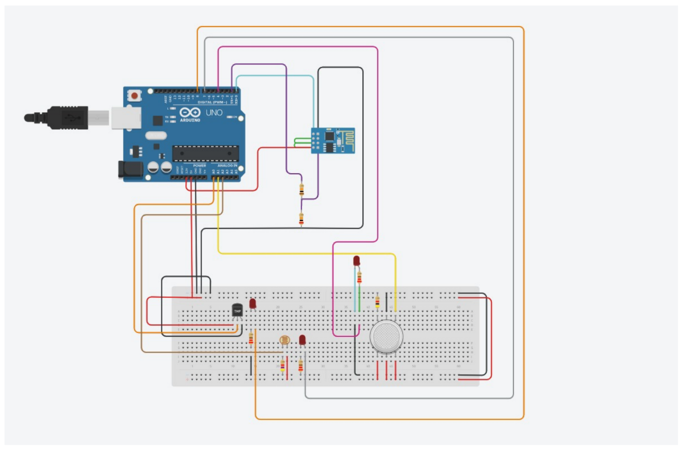
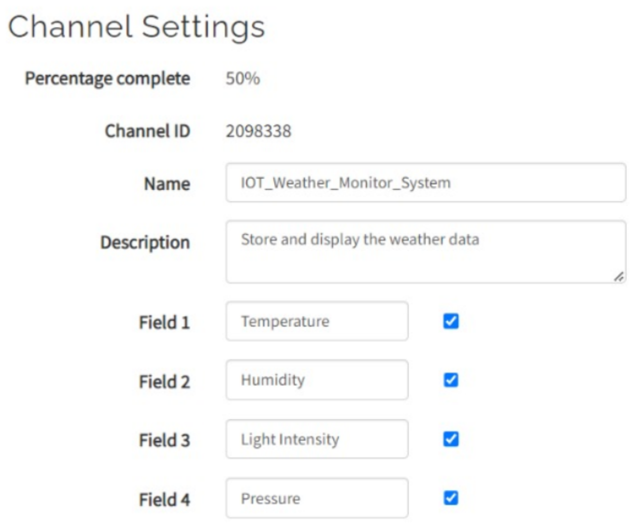


Fig 3. TinkerCAD Simulation of the Weather Monitoring System project

VI. Configuring and Connecting to ThingSpeak

The following settings were used to establish connection with the ThingSpeak API.

1. **MQTT Broker:** mqtt.thingspeak.com
2. **Port:** 1883 (unencrypted), 8883 (encrypted)
3. **Username:** ThingSpeak Channel ID
4. **Password:** ThingSpeak API Key
5. **Topic:** "channels/<channel_ID>/publish/fields/<fieldname>/<API_Key>"



Channel Settings

Percentage complete	50%	
Channel ID	2098338	
Name	IOT_Weather_Monitor_System	
Description	Store and display the weather data	
Field 1	Temperature	✓
Field 2	Humidity	✓
Field 3	Light Intensity	✓
Field 4	Pressure	✓

Fig 4. The channel setting configurations in the project

VII. Circuit Assembling and Software Integration With ThingSpeak

The aforementioned parts were used to put together the circuit, and the NodeMCU Board was programmed using the Arduino source code. Using the corresponding sensors installed in the hardware setup, this will execute and carry out the actions of reading the temperature, pressure, humidity, and light intensity data of the user environment.

The information so obtained is kept in the program's variables. Through the source code that has been uploaded and the necessary ThingSpeak API Key, a connection is made to the ThingSpeak server. For the field string to POST the user environment data, the temperature, pressure, humidity, and light intensity data are attached. Consequently, the information is transmitted to the ThingSpeak cloud platform through an encrypted POST request.As a result, to keep up with changes in the weather, this data is shown on the LCD and has a continuous refresh rate.

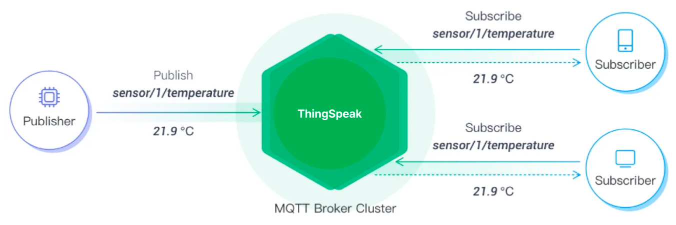


Fig 5. Flowchart describing the course of weather data in the MQTT publisher-subscriber model

VIII. Results and Conclusions

With the aid of the ThingSpeak graphical user interface on the server, the project has successfully been able to obtain the temperature, pressure, humidity, and light intensity data precisely and is able to demonstrate fluctuation of these parameters over time.

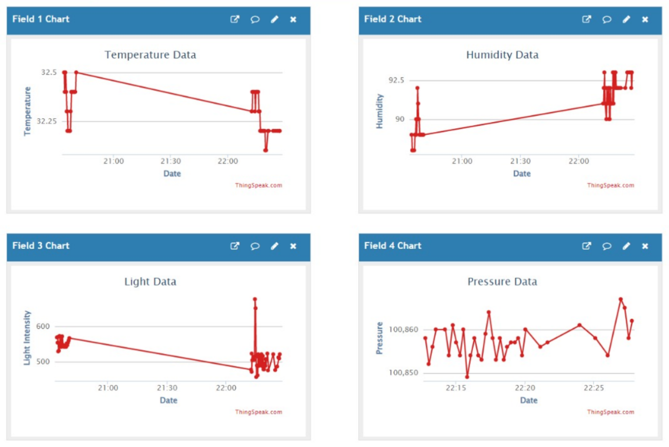


Fig 6. The output at the ThingSpeak server which gives a graphical analysis of the weather conditions on the basis of four parameters (temperature, humidity, pressure, light intensity).

IX. Scope for Improvements in the Project

The need to store and retrieve data from the user has been satisfied by the project that follows. The safe transfer of data between the MQTT publisher and subscriber has also been made possible by the use of encrypted ports and channels. The recent targeted assaults on IoT systems, particularly those employing the 'Mosquitto' tools to attack MQTT servers, have prompted interest in more innovative approaches to protect the privacy of user data. As a result, the project has to be made more secure before it can be deployed on a bigger scale. Using CA certificates and SSH keys would increase security, enhancing the project's overall security.

To assess the level of smoke and dust in the environment, additional parameters for tracking the user environment can be included. The user experience with the gadget would also be enhanced by the addition of sensors to track this data.

References

1. components101.com for datasheets of components
2. circuit-basics.com for I2C communication protocol implementation
3. Javapoint.com for Mqtt protocol