IoT Sensor System Using ESP8266

Amr Mohsen, 58-21006

Abdulrahman Waleed, 58-2089

Abdelrahman El-Abyad, 58-4464

Supervisors:

Dr. Tallal Elshabrawy

Dr. Lucian Emad

Abstract

This report presents the design and implementation of an IoT-based smart monitoring and alert system using ESP8266 Wi-Fi modules and DHT-11 sensors. The system monitors temperature and humidity levels, periodically transmitting data to a server for analysis. Using socket programming and the client-server model, the server evaluates sensor readings and provides real-time feedback to the ESP8266 modules. Abnormal readings trigger specific LED indicators, ensuring precise alerts for high or low temperatures and abnormal humidity levels. This system demonstrates a practical approach to IoT-based environmental monitoring with reliable feedback mechanisms.

1 Introduction

The Internet of Things (IoT) allows devices to communicate, exchange data, and automate tasks, transforming everyday operations in homes and industries. This project leverages IoT to create a smart system for monitoring environmental conditions using ESP8266 Wi-Fi modules and DHT-11 sensors.

The system operates on the client-server model, where the ESP8266 modules act as clients, transmitting temperature and humidity readings to a central server. The server evaluates the data and provides feedback to the clients. If readings fall outside predefined ranges, corresponding LEDs light up to indicate high or low temperature or abnormal humidity levels.

This project highlights the practical application of IoT for real-time monitoring, utilizing socket programming for efficient communication and threading for handling multiple client connections. The design ensures independent operation of sensors and LEDs, enabling precise alerts for distinct anomalies in environmental conditions.

2 System Objectives

1. Monitor environmental conditions using IoT sensors.
2. Transmit sensor readings to a central server periodically via Wi-Fi.
3. Analyze readings on the server and provide feedback to the clients (ESP8266 modules).
4. Light LEDs for abnormal readings, ensuring distinct alerts for high and low temperatures and abnormal humidity levels.

3 System Design

3.1 Hardware Components

- 2 × micro-USB data cables
- 2 × ESP8266 Wi-Fi modules
- 2 x Temperature and humidity sensors (DHT-11)

- LEDs (Red, Yellow, Blue)
- Breadboard and jumper wires

3.2 Software Tools

- MicroPython for ESP8266 programming
- Pin tester micro-python code
- Python for server programming
- Socket programming for communication
- Threading for handling multiple client connections

4 Methodology

4.1 Client Code

- **Sensor Reading**: The ESP8266 modules read temperature and humidity values every 5 seconds.
- **Wi-Fi Connection**: Each client connects to the server via TCP sockets.
- **Data Transmission**: Sensor readings are sent to the server at regular intervals of 5 seconds.
- **LED Feedback**: Clients interpret the server's response to control LEDs:
  - Normal Temperature and Humidity: No action.
  - Too High Temperature: Red LED lights up.
  - Too Low Temperature: Yellow LED lights up.
  - Abnormal Humidity: Blue LED lights up.

4.2 Server Code

- **Threaded Connections**: Handles multiple clients simultaneously.
- **Data Analysis**: Checks if readings fall within the normal range.
- **Feedback**: Sends appropriate alerts to clients:
  - Normal conditions: Continue operation.
  - Too High: Notify the client of HIGH temperature.
  - Too Low: Notify the client of LOW temperature.
  - Abnormal humidity: Notify the client of abnormality.

4.3 Normal Ranges

- Temperature: 20°C - 30°C
- Humidity: 40% - 70%

5 Implementation

5.1 Circuit Setup

1. Connect each sensor to an ESP8266's analog input pin:
    - Connect the sensor's output GPIO pin 5.
    - Attach the GND pin of the sensor to the ESP8266's ground.
2. Attach LEDs to GPIO pins on the ESP8266:
    - Connect the positive terminals of the LEDs to the ESP8266 GPIO pins 2, 4 and 0 indicating Red, Yellow and Blue respectively.
    - Connect the negative terminals of the LEDs to the ESP8266's ground.
3. Use the breadboard for power distribution, LED connections, and grounding:
    - Connect the ESP8266's 3v3 and GND pins to the breadboard's power rails.
    - Distribute power to all components via the breadboard.

5.2 Code Implementation

Client Code (MicroPython) Reads sensor values, sends data to the server every 5 seconds, and receives feedback to control LEDs.

Server Code (Python) Listens for client connections, processes incoming sensor readings, and sends feedback based on predefined normal ranges.

6 Results

The system was tested under various conditions:

- **Normal (Temperature: 25°C, Humidity: 50%)**: No LEDs lit.
- **Abnormal (Temperature: 35°C, Humidity: 40%)**: Red LED lit for high temperature.
- **Abnormal (Temperature: 15°C, Humidity: 70%)**: Yellow LED lit for low temperature.
- **Abnormal (Temperature: 35°C, Humidity: 71%)**: Red and Blue LEDs lit.

7 Challenges and Solutions

- **ESP8266 Connections**: Some pins didn't work for the ESP8266 so we used a pin tester micro-python code to verify our connections.
- **Power Supply**: Some micro USB cables didn't work for the ESP8266 at first but then it was replaced with a new one to install the drivers and then power it up with the old one.
- **Flash Sizes**: The ESP8266 firmware installation always failed because the default size was set to 2MIB instead of 4MIB.

8 Conclusion

This project demonstrates a functional IoT-based monitoring system using ESP8266 modules and sensors. By integrating server-side processing with client feedback, the system provides real-time alerts for environmental anomalies. Potential future improvements include cloud integration for remote access and data visualization.