ECC591: Project I (A)

Professor: Dr. Dalia Nandi, Dept. of ECE, IIIT Kalyani

Project Report

IoT with Embedded Systems

IoT Room Conditions Monitoring Device

By:

Shirish Manoj Bobde
ECE/21152 — 812
Geetansh Jangid
ECE/21124 — 784
Divyanshu Kumar
ECE/21123 — 783

Tuesday, November 27th, 2023

Contents

1 Overview

Room conditions monitoring is fundamentally crucial across diverse domains, serving as a linchpin for ensuring occupant well-being, maintaining product quality, and upholding regulatory standards. The impact on human health is profound, with indoor air quality (IAQ) playing a central role. Monitoring parameters such as temperature, humidity, and pollutants is vital to identify and mitigate potential health risks associated with poor air quality, mold growth, and respiratory issues. In industrial and commercial applications, the focus shifts to quality assurance and equipment performance. Monitoring creates a safeguard for products sensitive to environmental conditions, preventing degradation and ensuring quality standards. Additionally, it plays a pivotal role in optimizing HVAC systems for energy efficiency and aligning with sustainable practices. Beyond immediate benefits, room conditions monitoring is integral for compliance with regulatory standards in various industries, averting legal issues and ensuring the safety of occupants and processes. In essence, the need for room conditions monitoring is multifaceted, addressing health, industrial processes, energy efficiency, and regulatory compliance, with far-reaching implications for immediate well-being and overarching sustainability goals.

Internet of Things (IoT) by conceptualizing and implementing a Room Conditions Monitoring device. IoT, a paradigm that interconnects physical devices through the internet, forms the backdrop for this project, wherein the objective was to harness its potential for enhancing our understanding of environmental dynamics within a confined space. With a focus on practical applications, the project sought to contribute to the development of intelligent systems capable of real-time data collection and analysis. At its core, the Room Conditions Monitoring device employed a meticulously curated ensemble of components, including the Arduino UNO R3, NodeMCU ESP8266, DHT-11 for temperature and humidity, and BMP180 for pressure. The amalgamation of these components aimed to furnish an integrated system capable of continuously monitoring and transmitting key environmental parameters. This project, therefore, serves as an immersive introduction to both the expansive realm of IoT and the specific realm of room monitoring, underscoring the potential impact of such technology on various domains, from smart homes to industrial settings. The subsequent sections delve into the detailed methodology, strategies employed, implementation steps, obtained results, and the promising avenues for future enhancements within this comprehensive exploration.

The Arduino UNO R3 stands as a quintessential microcontroller board, revered for its versatility and user-friendly design. At its core is the ATmega328P microcontroller, boasting a myriad of digital and analog input/output pins, allowing seamless interaction with various sensors, actuators, and other peripherals. The open-source nature of Arduino facilitates a vast community of developers, fostering innovation and collaborative learning. The Arduino UNO's simplicity in programming, aided by the Arduino IDE, empowers both beginners and seasoned engineers to prototype and develop projects efficiently. Its adaptability in diverse applications, from educational projects to complex IoT solutions, underscores its significance in the realm of embedded systems.

The NodeMCU ESP8266, an integral component in the Internet of Things (IoT) land-

scape, exemplifies the convergence of affordability and functionality. Powered by the ESP8266 Wi-Fi module, this development board provides a seamless platform for IoT applications. With its onboard Wi-Fi capabilities, the NodeMCU enables wireless communication, transforming everyday devices into smart, connected entities.

The Serial Peripheral Interface (SPI) stands as a synchronous serial communication protocol, designed for efficient data transfer between microcontrollers and peripheral devices. Characterized by its full-duplex nature, SPI employs master-slave architecture, allowing multiple peripheral devices to be connected to a single master. The protocol's simplicity and speed make it ideal for applications demanding real-time data exchange, such as sensor interfacing and memory communication. The presence of four essential lines — Master Out Slave In (MOSI), Master In Slave Out (MISO), Serial Clock (SCK), and Slave Select (SS) — facilitates seamless communication, while the absence of a defined protocol for data transmission size enhances flexibility. SPI's ubiquity in microcontroller-based systems underscores its efficiency in supporting a multitude of applications. Inter-Integrated Circuit (I2C), a two-wire communication protocol, excels in connecting multiple integrated circuits on a shared bus. Developed by Philips Semiconductors (now NXP Semiconductors), I2C simplifies inter-device communication with its master-slave architecture. The protocol's bidirectional nature, employing Serial Data (SDA) and Serial Clock (SCL) lines, fosters efficient data exchange while minimizing the physical complexity of interconnecting devices. I2C's versatility shines in scenarios requiring multiple sensors or devices to communicate seamlessly, such as in robotics, embedded systems, and IoT applications. The presence of unique device addresses ensures clear communication channels within the network, making I2C a preferred choice for simplifying complex interconnections in a wide array of electronic systems.

2 Methodology

Beginning with the integration of sensors, the DHT-11 for temperature and humidity and the BMP180 for pressure were connected to the Arduino UNO R3, with dedicated Arduino code developed for accurate data retrieval. Subsequently, a wired communication link was established between the Arduino and the NodeMCU ESP8266, with the Arduino coded to transmit sensor data to the NodeMCU. Wireless communication was then set up between the NodeMCU and ThingsSpeak, a platform for data visualization. This included configuring the NodeMCU for Wi-Fi connectivity and developing code for secure data transmission to the ThingsSpeak channel using the API. The ThingsSpeak channel was set up to receive and visualize the transmitted data, ensuring the creation of a comprehensive monitoring system. Rigorous testing, calibration of sensors for accuracy, and optimization for energy efficiency were incorporated into the methodology. Throughout the process, documentation was maintained for hardware connections, software configurations, and troubleshooting steps, supported by a version-controlled code repository. The methodology concluded with the verification of real-time data visualization on the ThingsSpeak channel, ensuring the reliability of the implemented Room Conditions Monitoring device.

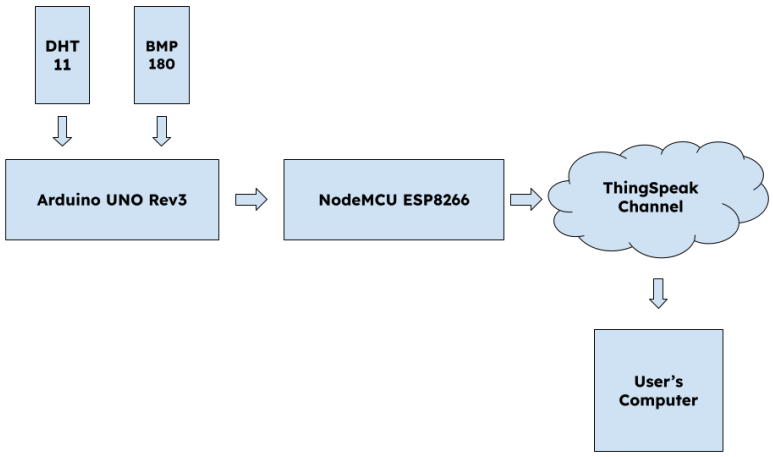

Figure 1: A flowchart of the methodology used

2.1 Implementation Details

A room monitoring system is constructed by connecting the DHT-11 and BMP180 sensors to the Arduino UNO Rev3 Micro-controller's digital input/output pins using I2C, and NodeMCU ESP8266.

The following subsections shed light on the implementation details further.

2.1.1 Hardware Design for the System

The hardware system is made up of four parts: an Arduino UNO Rev3 Micro-controller, a DHT-11 Sensor, and BMP180 Sensor and NodeMCU ESP8266. Arduino UNO Rev3 fetches sensor values from the sensors DHT-11 and BMP180 and sends it to NodeMCU ESP8266.

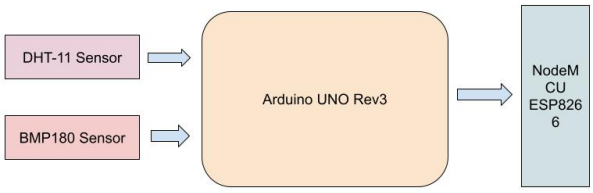

Figure 2: The abstract hardware design

2.1.2 Circuit Design for the system

Fig.3 Depicts the hardware system circuit design created with the Fritzing environment.

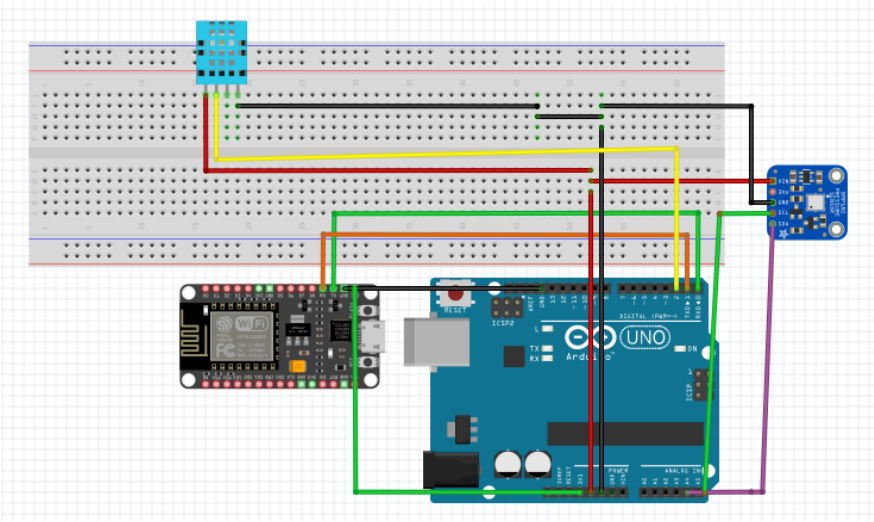

Figure 3: The circuit design.

2.1.3 Physical Implementation of the System

We used Arduino UNO Rev3 with DHT-11/BMP180 and NodeMCU ESP8266. Using Jumper cables (both Male to Male and Female to Male) with Breadboard

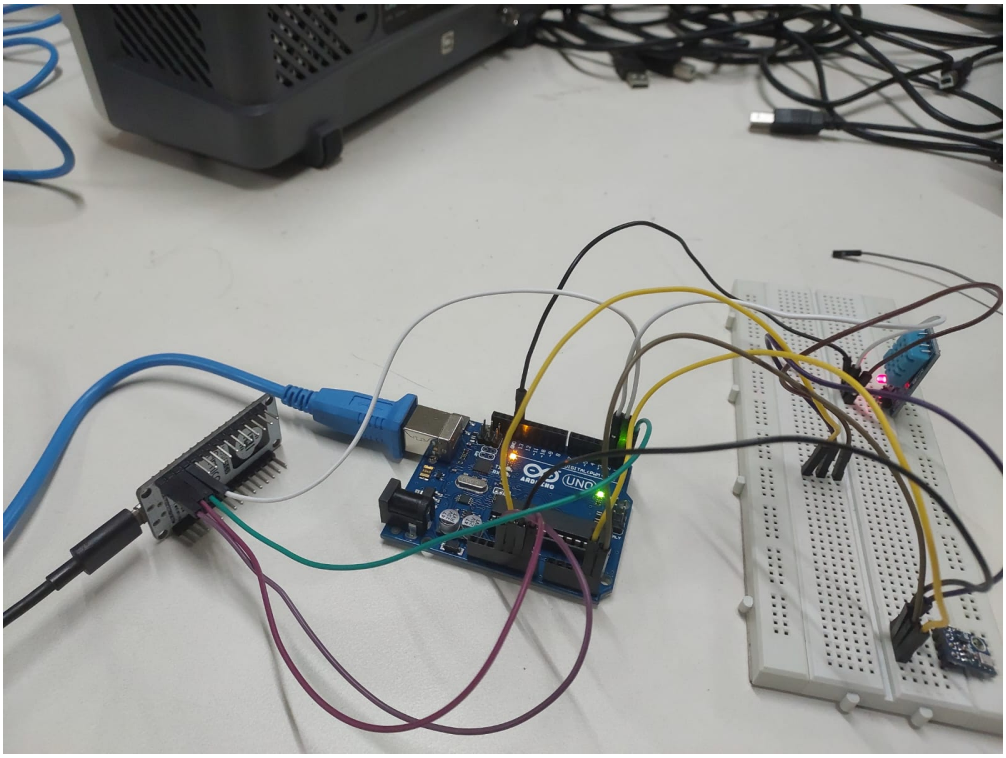

Figure 4: The physical implementation of **Circuit.**

2.1.4 Arduino UNO Rev3 Code:

The Arduino sketch file `arduino.ino` given was written and uploaded to the Arduino UNO Rev3 using the Arduino IDE v2. It takes values from both the sensors and sends the data over Serial Communication to NodeMCU ESP8266 to further flow the data over the cycle.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085.h>
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11

Adafruit_BMP085 bmp;
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
  bmp.begin();
```

```
}

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float p = 0.0;
  if(!bmp.begin())
  {
    p = 0.00;
  }
  else
  {
    p = bmp.readPressure() / 100.0F;
  }

  Serial.print("T:");
  Serial.print(t);
  Serial.print(";H:");
  Serial.print(h);
  Serial.print(";P:");
  Serial.print(p);
  Serial.println();

  delay(1000);
}
```

2.1.5 NodeMCU ESP8266 Code:

The following code uses libraries such as ESP8266WIFI and ESP8266HTTPClient to incorporate HTTP POST Requests to ThingSpeak Server using an API key that can be generated on the ThingSpeak portal. It converts the coming data into a single string where T stands for Temperature, H stands for Humidity and P stands for Pressure in Celsius, Percentage and hPa respectively. It also allows us to view details of the data sent on Serial Monitor at a baud rate of 9600. When NodeMCU ESP8266 is connected to the internet via 2.4GHz Wi-Fi of our choice. This allows NodeMCU ESP8266 to continuously send data to the ThingSpeak server for further Visualization.

```
#include<ESP8266WiFi.h>
#include<WiFiClient.h>
#include<ESP8266HTTPClient.h>
String URL=
"http://api.thingspeak.com/update?api-_key=6JF5WMFN4THKHTH8&field1=";
```

```
void setup(){
  Serial.begin(9600);
  WiFi.disconnect();
  delay(500);
  Serial.print("Start connection");
  WiFi.begin("SSID","Password");
  while((!(WiFi.status()== WL_CONNECTED))){
     delay(200);
     Serial.print("..");
    }
  Serial.println("Connected");
}

void loop(){
  if(Serial.available()>0){
    String data=Serial.readStringUntil('\n');
    Serial.println(data);

    float p = 0.0;
    float t = 0.0;
    float h = 0.0;

    // Create a string stream to parse the input string
    String key;
    float value;

    // Use indexOf to find the position of each parameter
    int start = 0;
    int end = data.indexOf(';');

    // Iterate through parameters and extract values
    while (end != -1) {
      String param = data.substring(start, end);

      // Split parameter into key and value
      int colonIndex = param.indexOf(':');
      key = param.substring(0, colonIndex);
      value = param.substring(colonIndex + 1).toFloat();

      // Assign values based on key
      if (key == "T") {
        t = value;
```

```
    } else if (key == "H") {
      h = value;
    } else if (key == "P") {
      p = value;
    }

    // Move to the next parameter
    start = end + 1;
    end = data.indexOf(';', start);
  }

  // Handle the last parameter (pressure)
  String lastParam = data.substring(start);
  int lastColonIndex = lastParam.indexOf(':');
  key = lastParam.substring(0, lastColonIndex);
  value = lastParam.substring(lastColonIndex + 1).toFloat();
  if (key == "P") {
    p = value;
  }

  // Print the extracted values
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print("Humidity: ");
  Serial.println(h);
  Serial.print("Pressure: ");
  Serial.print(p);

  sendData(t,h,p);
  }

}
void sendData(float t, float h, float p){
  WiFiClient client;
  HTTPClient http;
  String newUrl=URL
        +String(t)+"&field2="+String(h)+"&field3="+String(p);
  http.begin(client,newUrl);
  int responsecode=http.GET();
  String data=http.getString();
  Serial.println(data);
  http.end();
```

3 Results

ThingSpeak is an open-source Internet of Things (IoT) platform designed to facilitate data collection, analysis, and visualization. Its versatility makes it an ideal choice for IoT projects, including room conditions monitoring. The platform operates on a channel-based structure, where users can create channels to which IoT devices, such as NodeMCU ESP8266, can stream data. Each channel represents a specific dataset, allowing for organized management of information. ThingSpeak simplifies the integration process by providing user-friendly APIs and libraries, including HTTP and MQTT, enabling seamless communication between IoT devices and the platform. One of its standout features is its real-time data visualization capabilities. ThingSpeak offers a variety of customizable widgets, including charts and graphs, which enable users to monitor trends and patterns in the collected data with ease. Additionally, the platform supports MATLAB Analysis, allowing for advanced data processing and visualization techniques. Users can access their data remotely through the ThingSpeak website or retrieve it programmatically through APIs for further analysis. ThingSpeak's simplicity, flexibility, and robust visualization tools make it an invaluable asset in the realm of IoT-based room conditions monitoring, enhancing the accessibility and usability of collected data for informed decision-making and optimization of indoor environments.

Using ThingSpeak for IoT Data Collection

1. **Create a ThingSpeak Account:**
    - Start by visiting the ThingSpeak website.
    - Sign up for a ThingSpeak account if you don't have one.
2. **Create a Channel:**
    - Inside your ThingSpeak account, navigate to the Channels section.
    - Create a new channel for each set of data you want to collect.
    - Think of a channel as a virtual container for your data.
3. **Configure Channel Fields:**
    - Define specific fields for each channel to store different types of data (e.g., temperature, humidity, pressure).
    - Keep in mind that ThingSpeak channels have a limit of 8 fields.
4. **Obtain API Key:**

- ThingSpeak provides an API key for each channel.
- This key is essential for authenticating your device when sending data to ThingSpeak.

5. **Program Your Device (Arduino/NodeMCU):**
   - Write a program for your microcontroller (Arduino/NodeMCU) that reads data from your sensors (BMP180 and DHT11).
   - Use the obtained API key to authenticate and send data to ThingSpeak using HTTP requests.
6. **HTTP POST Requests:**
   - Configure your program to send data to ThingSpeak using HTTP POST requests.
   - Include the API key in the request for authentication.
   - Specify the URL corresponding to your ThingSpeak channel.
7. **Data Storage:**
   - ThingSpeak stores the data in your channel and timestamps it.
   - Specify the update frequency for your data (e.g., every 15 seconds).
8. **Visualization and Analysis:**
   - Explore the web interface provided by ThingSpeak to visualize your data in real-time graphs.
   - Perform basic analysis using built-in MATLAB functions.
9. **Make Changes in Visualization and Analysis:**
   - Experiment with different visualization options provided by ThingSpeak.
   - Adjust the analysis parameters to suit your needs.
10. **Access Control:**
    - Utilize ThingSpeak's access control features to manage who can access your data.
    - Choose between keeping the data private or sharing it with others (public access).
11. **Export Data:**
    - ThingSpeak allows you to download your data in various formats, including JSON, XML, and CSV.
    - Use these export options for further analysis or storage.

ThingSpeak Channel Integration for Room Conditions Monitoring

The incorporation of a ThingSpeak channel into our room conditions monitoring project provides a robust platform for visualizing and analyzing the dynamic changes in temperature, humidity, and pressure within the monitored environment.

Data Visualization

The cornerstone feature of our ThingSpeak channel is its capability to visually represent data on a time-based graph. This graphical representation facilitates a real-time observation of temperature, humidity, and pressure trends, offering a comprehensive view of the evolving environmental conditions.

Data Formats and Download Options

Our ThingSpeak channel has been configured to organize data into three distinct formats:

- **JSON (JavaScript Object Notation):** This lightweight and human-readable data format provide a structured representation of the collected data, allowing for seamless integration into various applications.
- **CSV (Comma-Separated Values):** The widely used CSV format simplifies data exchange and supports easy importation into spreadsheet software, facilitating further analysis and visualization.
- **XML (eXtensible Markup Language):** XML, being both human-readable and machine-readable, serves as an effective markup language for structured data representation.

Users have the flexibility to download data in their preferred format, catering to diverse data analysis tools and platforms.

Predictive Accuracy

In controlled conditions with a set temperature of 25°C, our room conditions monitoring device demonstrated impressive predictive accuracy. The device precisely forecasted temperature, pressure, and humidity within specified ranges: 0-50°C for temperature, 20-90

This high level of accuracy reaffirms the reliability of our monitoring device in capturing and interpreting real-time environmental conditions.

Accessing the ThingSpeak Channel

The valuable insights derived from the collected data and visualizations are accessible through our dedicated ThingSpeak channel. To explore the graphs, download data, and draw conclusions from observed trends in temperature, humidity, and pressure, please visit the following link: `https://thingspeak.com/channels/2358684`.

The integration of ThingSpeak into our project not only enhances our understanding of room conditions but also provides a solid foundation for informed decision-making based on real-time data.

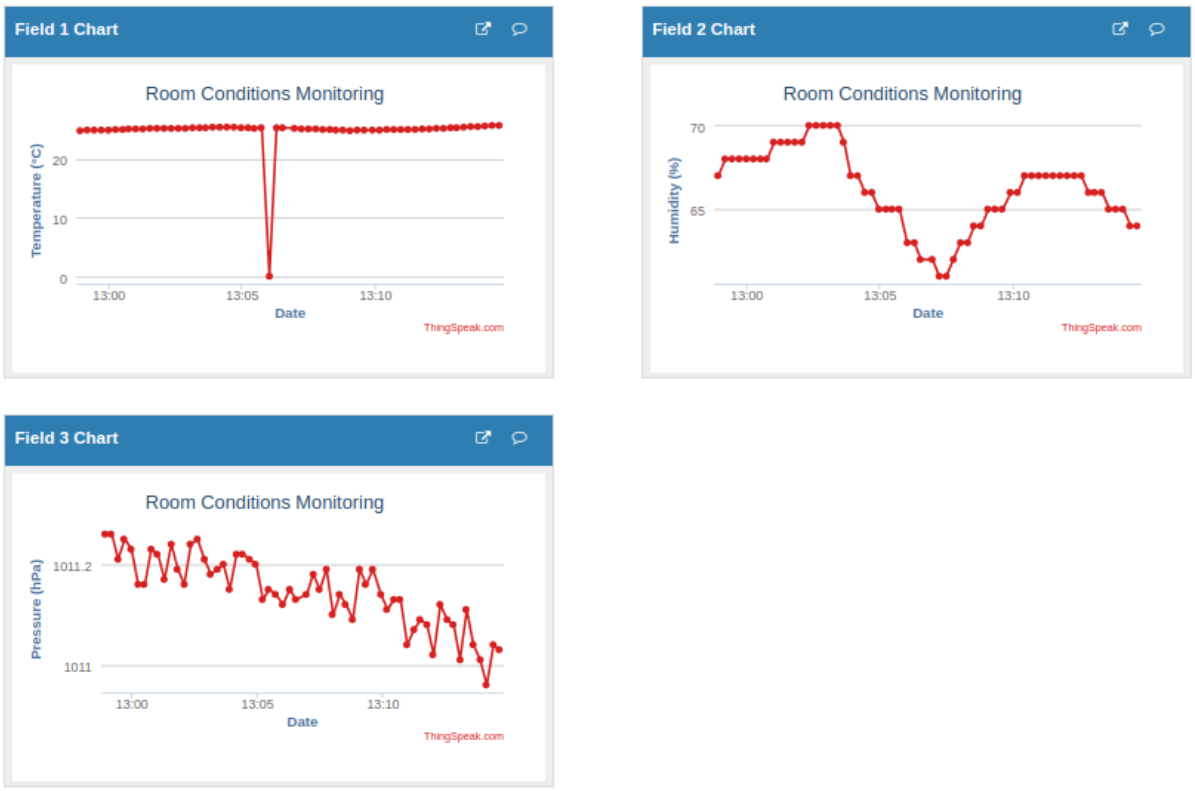

Figure 5: Data visualization on ThingSpeak.

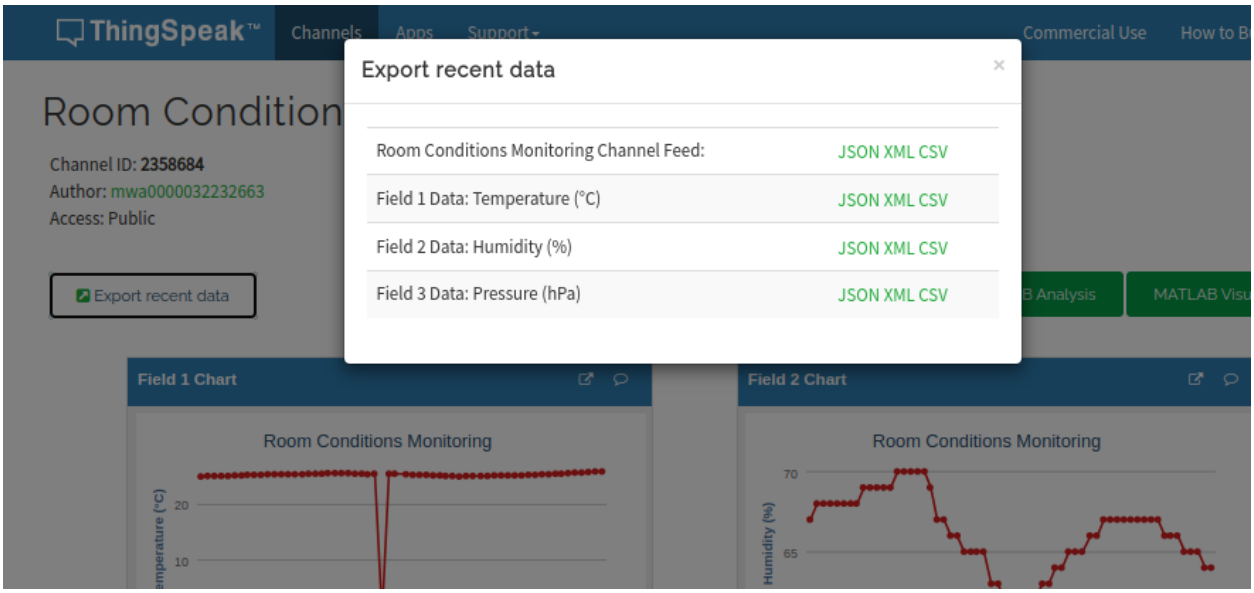

Figure 6: ThingSpeak provides us with various Data Formats.

	A	B	C	D	E
1	created_at	entry_id	temperature	Humidity	Pressure
2	2023-11-28 07:17:39 UTC	38	25.1	67	1011.53
3	2023-11-28 07:17:55 UTC	39	25.1	68	1011.48
4	2023-11-28 07:18:11 UTC	40	25.1	68	1011.52
5	2023-11-28 07:18:26 UTC	41	25.2	68	1011.51
6	2023-11-28 07:18:42 UTC	42	25.2	68	1011.46
7	2023-11-28 07:18:58 UTC	43	25.3	68	1011.51
8	2023-11-28 07:19:13 UTC	44	25.3	68	1011.45
9	2023-11-28 07:19:29 UTC	45	25.4	68	1011.52
10	2023-11-28 07:19:45 UTC	46	25.4	68	1011.48
11	2023-11-28 07:20:00 UTC	47	25.4	68	1011.48
12	2023-11-28 07:20:16 UTC	48	25.5	68	1011.54
13	2023-11-28 07:20:44 UTC	49	25.4	69	1011.48
14	2023-11-28 07:21:00 UTC	50	25.6	68	1011.48

Figure 7: A peek at the CSV file created.

4 Problems Encountered

- **NodeMCU Local to Global Server Configuration:** Initially, data was only accessible when connected to the same local server as NodeMCU. To enable global access,

we integrated the ESP8266HTTPClient.h and WiFiClient.h libraries. This modification facilitated data transmission to a website and a cloud, allowing users to access the data from anywhere.

- **Baud Rate Alignment:** Initially, default baud rates for NodeMCU (115200) and Arduino (9600) were causing communication issues. To resolve this, we standardized the baud rate to 9600 on both devices, ensuring seamless serial data transfer.

- **Data Extraction Enhancement:** Arduino transmitted data in string format to NodeMCU, but website integration required individual float data for Temperature, Humidity, and Pressure. To address this, we implemented code modifications to extract and format data appropriately.

- **Data Loss Mitigation:** During data transmission from Arduino to NodeMCU, delays and communication issues caused partial string loss on the NodeMCU end. We initially introduced delays on the NodeMCU side and later implemented a minimum string size in the NodeMCU code to resolve this intermittent data loss.

- **BMP180 Sensor Adaptation:** The BMP180 sensor, lacking pre-soldered pins, presented challenges in data retrieval. To address this, we utilized standard wires for connections and incorporated a conditional check (!bmp.begin()) to handle scenarios where the BMP180 was not functioning properly.

- **Code Uploading Considerations:** When uploading code to the Arduino board, it is crucial to disconnect RX and TX pins. Failure to do so can result in unsuccessful code uploads.

5 Future Scope

This system could be implemented and interacted more naturally on a web-page created and hosted on a local machine, which can also provide broadcasting it using technologies like Fast API and Flask in Python. With the integration of machine learning, the Room Conditions Monitoring Device stands as a transformative prospect. By implementing machine learning algorithms, the device can evolve beyond mere data collection and analysis, venturing into predictive capabilities. This involves leveraging historical data to forecast future room conditions, enabling preemptive actions to optimize the environment. Furthermore, the incorporation of personalized recommendations adds a layer of sophistication, tailoring the indoor setting based on individual preferences and habits. The CSV files generated can be used as datasets to train lucrative models according to specific use cases.

Simultaneously, extending the device's capabilities through integration with the broader Internet of Things (IoT) ecosystem opens doors to enhanced functionality and collaboration. Connecting with various IoT devices and platforms creates a networked environment, fostering interoperability and the potential for cross-device automation. Such integration not

only allows the Room Conditions Monitoring Device to contribute to a more comprehensive smart home or industrial IoT solution but also aligns with industry standards, ensuring compatibility and versatility in diverse IoT ecosystems. In industries dealing with delicate materials that require careful monitoring of conditions, such as storing seeds in farms, wax museums, dehumidifiers, and smart home applications, as well as air conditioning systems, these future scopes collectively propel the device beyond monitoring into the realms of intelligent prediction and collaborative connectivity, promising a more nuanced and adaptive approach to room conditions management.

We believe we have effectively addressed the task at hand. We present our system for thorough examination and further evaluation.

6 References

- P. F. Gabriel and Z. Wang, "Design and Implementation of Home Automation system using Arduino Uno and NodeMCU ESP8266 IoT Platform," 2022 International Conference on Advanced Mechatronic Systems (ICAMechS), Toyama, Japan, 2022, pp. 161-166, doi: 10.1109/ICAMechS57222.2022.10003361.

- Razali, M.A.A., Kassim, M., Sulaiman, N.A. and Saaidin, S., 2020, June. A ThingSpeak IoT on real-time room condition monitoring system. In 2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS) (pp. 206-211). IEEE.

- G. M. Debele and X. Qian, "Automatic Room Temperature Control System Using Arduino UNO R3 and DHT11 Sensor," 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2020, pp. 428-432, doi: 10.1109/ICCWAMTIP51612.2020.9317307.

- P. Agrawal and G. Chitranshi, "Internet of Things for monitoring the environmental parameters," 2016 International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds, Noida, India, 2016, pp. 48-52, doi: 10.1109/INCITE.2016.7857588.