Paper ID: 033

Real-Time Server Room Monitoring System Using Internet of Things (IoT) Technology

[1#]Y Tishan, [2]I Ashly
[1]*Department of Computer Science & Software Engineering, NSBM*

NSBM Green University
Mahenwaththa, Pitipana, Homagama, 10200, Sri Lanka.

[2]*Department of Electrical, Electronics, and Computer Engineering,* NSBM Green University
Mahenwaththa, Pitipana,Homagama, 10200, Sri Lanka.

[#]yasirutishan@hotmail.com

Abstract**—** *With the exponential growth of server rooms and data centres, ensuring their optimal functioning and protection has become a critical concern. This research presents an IoT-based server room monitoring system that utilizes microcontrollers and sensors to continuously monitor key environmental metrics, such as temperature, humidity, and power, while also detecting potential hazards including vibration, fire, and smoke. The system employs the NodeMCU microcontroller, which seamlessly integrates various sensors including Smoke, Flame, AC Voltage, Temperature and humidity, and Vibration sensors. A combination of LED lights and a buzzer is employed to promptly alert users when any monitored factor exceeds its predefined threshold. The system offers user access through both a mobile and web application, allowing for registration and convenient retrieval of pertinent information. By providing real-time monitoring and rapid notifications, this system enhances the reliability and security of server rooms, enabling proactive maintenance and timely resolution of potential issues. This research contributes to the field of IoT-based server room monitoring, addressing the growing need for efficient and robust monitoring solutions in the face of increasing data demands.*

Keywords**—** *IoT, Server Room, Arduino, Monitoring System, Environmental, Sync fusion Flutter, Google IoT Cloud.*

I. INTRODUCTION

This research introduces an **IoT-based Server Room Monitoring System** that utilizes microcontrollers and sensors to enable real-time monitoring[1] of essential metrics and timely detection of potential hazards. The system is designed to efficiently operate and protect server rooms, addressing the increasing demand for data storage and processing. It also includes a user-friendly mobile application interface for easy access and control.

The IoT-based Server Room Monitoring System is designed to monitor key server metrics such as temperature, humidity, and power consumption. Additionally, it incorporates sensors to detect and alert users about potential risks including vibration, fire, and smoke.[1] By employing the ESP-32 microcontroller, the system enables seamless connectivity and integration of multiple sensors, simplifying the monitoring process. The sensors used in the system include the Smoke Sensor, Flame Sensor, AC Voltage Sensor, Temperature and Humidity Sensor, and Vibration Sensor, which collectively gather data on environmental conditions within the server room.

To ensure timely notifications and alerts, the system utilizes both visual and auditory indicators. An LED Light and a Buzzer are employed as output devices to promptly inform users when any monitored factor exceeds[1] its predefined threshold. This enables swift action to be taken in response to critical events, reducing the risk of potential damage to server equipment and data loss.

User interaction with the IoT-based Server Room Monitoring[2] System is facilitated through a mobile application. Users are required to register and access the system through a web-based interface, which provides a convenient means to retrieve real-time information and control the monitoring process. By offering accessibility through mobile devices, users[2] can stay connected and informed about the server room's status regardless of their physical location.

The importance of continuous monitoring and proactive maintenance of server rooms cannot be overstated. Even minor disruptions or failures in these environments can have significant repercussions on business operations. By implementing the IoT-based Server Room Monitoring System, organizations[2][3] can mitigate risks, improve reliability, and ensure the smooth operation of their server infrastructure. This research contributes to the field by presenting an effective and comprehensive solution for server room monitoring, addressing the growing need for robust monitoring systems in today's technology-driven landscape.[3]

II. BACKGROUND

A. Problem Statement

The absence of adequate real-time monitoring functionalities[4] presents notable obstacles in extant server room monitoring systems, resulting in possible performance impediments and an increased likelihood of equipment malfunction or data loss. These systems fail to promptly detect and alert users about critical hazards such as fire, smoke, and excessive environmental conditions, leaving server rooms susceptible to damage and disruption. In addition, the restricted availability of access and deficient user interface functionalities pose a significant obstacle to efficient supervision and regulation, thereby obstructing users' capacity to obtain crucial data and react promptly to potential concerns.[5] Addressing these shortcomings is essential to enhancing the reliability and security of server rooms in the face of escalating data demands.

B. Proposed Solution

The proposed solution is to develop an IoT-based Server Room Monitoring System that offers comprehensive and real-time monitoring of server metrics and timely detection of potential hazards.[5] The system will provide a user-friendly interface accessible through both mobile and web applications, ensuring convenient access, data retrieval, and control for users. [5][6]

The key components and features of the solution include:

- IoT-Based Architecture: The proposed solution utilizes an IoT-based architecture, integrating microcontrollers and sensors for data collection on critical server metrics.
- Sensor Integration: Various sensors, including Smoke, Flame, AC Voltage, Temperature and humidity, and Vibration sensors, are integrated into the system to enable comprehensive monitoring of environmental conditions and hazard detection within the server room.
- Real-Time Monitoring: The system offers real-time monitoring of server metrics, ensuring prompt detection of fluctuations or abnormalities. Users can access and monitor the server room's status in real-time through mobile and web applications.
- Hazard Detection and Alert System: An intelligent hazard detection system analyzes sensor data in real-time, triggering immediate alerts through LED lights and a buzzer when fire, smoke, or excessive environmental conditions are detected. This enables users to quickly address issues and prevent further damage.
- User-Friendly Interface: The mobile and web applications provide a user-friendly interface for easy registration, data retrieval, and control of the monitoring system. Users can set threshold limits, receive notifications, and access historical data for analysis and troubleshooting purposes.
- Remote Accessibility: The solution allows users to remotely access the server room monitoring system through mobile and web applications, ensuring monitoring and alert reception even when users are not physically present.
- Data Logging and Analysis: The system includes a data logging feature that captures and stores historical data. This enables trend analysis, anomaly detection,[7] and long-term performance monitoring, providing insights into server room conditions and facilitating the identification of potential issues.[8]

III. OBJECTIVES

1. Real-Time Temperature and Humidity Monitoring: Develop a system that can continuously monitor and display the current temperature and humidity levels in the server room. The system will use appropriate sensors to gather accurate and reliable data, allowing users to have instant access to the environmental conditions within the server room.
2. Historical Data Analysis: Implement a data logging feature to capture and record temperature and humidity measurements periodically. This data will be stored for subsequent analysis and graphing, providing insights into long-term trends and patterns. The system will enable users to view graphical representations of temperature and humidity variations over time,[9] facilitating informed decision-making and proactive maintenance.
3. Smoke Detection: Integrate a smoke detection mechanism into the system to identify the presence of smoke or potential fire hazards within the server room. The system will employ sensors specifically designed to detect smoke particles,[10] triggering immediate alerts to notify users of any potential dangers and allowing for swift response measures to prevent damage to server equipment and critical data.
4. Power Fluctuation Monitoring: Implement functionality to monitor power fluctuations within the server room. By utilizing appropriate sensors, the system will detect variations in the power supply and promptly notify users in the event of irregularities. This feature will enable proactive measures to prevent power-related issues and ensure the stable and uninterrupted operation of the server room.
5. Data Utilization for Analysis and Graphing: Develop mechanisms to store and organize the collected data in a structured manner, facilitating subsequent analysis and graphing.[9][10] The system will allow users to access historical data, enabling them to identify patterns, anomalies, and correlations. Graphical representations of measurements will aid in visualizing trends and identifying potential issues, leading to informed decision-making and proactive maintenance strategies.

IV. SYSTEM ARCHITECTURE

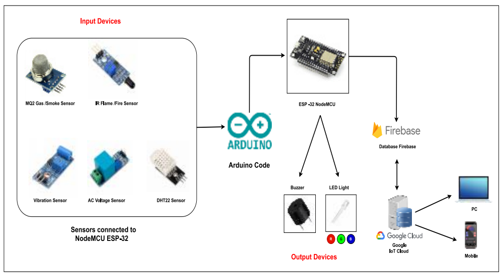


Figure 1 - System Architecture Diagram

This Server Room Monitoring System employs a combination of input and output devices to facilitate its functionality. The input devices utilized in the system include sensors such as the MQ2 Smoke Sensor, Fire Sensor, Vibration Sensor, AC Voltage Sensor, and DHT22 Sensor, which are responsible for gathering data. These sensors are connected to the ESP-32 NodeMCU through the Arduino Code. The collected data from the NodeMCU is then transmitted to the Google IoT Cloud via a Wi-Fi connection. Additionally, the data is stored in the Firebase database, facilitating data exchange between the server and Firebase.

To access the data, users can utilize either a PC or a mobile phone. The data can be accessed through a web application when connected to a PC, or through a mobile application when using a mobile phone. In both cases, the data needs to be hosted on the server. Furthermore, the system includes output devices, namely the Buzzer and LED Light. These devices are connected to the system via the NodeMCU and provide visual and audible alerts. The LED lights are designed with three colours, namely Red (R), Green (G), and Blue (B), to indicate different status conditions.

A. PCB Architecture

The hardware development for this system followed the PCB architecture, and the PCB diagram was designed using the Fritzing software. The design incorporates all the necessary components, including the sensors and the NodeMCU. The connection between these sensors and the NodeMCU is clearly outlined in the Arduino code.

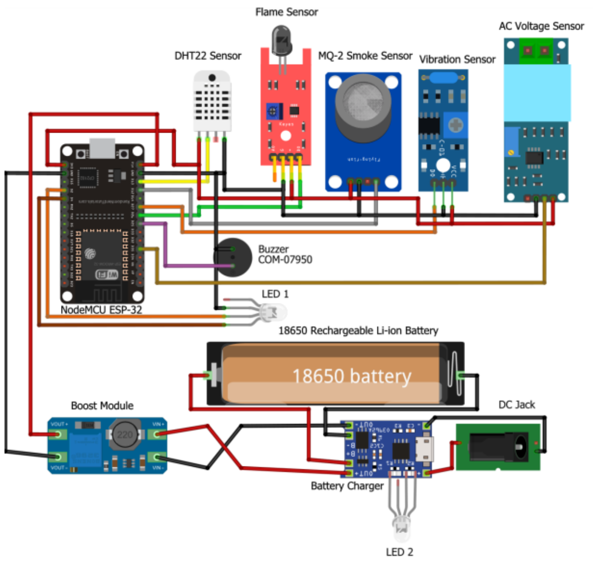


Figure 2 - PCB Architecture Diagram

Here is the pin configuration for the sensor connections:

- The DHT22 sensor's digital pin is connected to D13 of the NodeMCU.
- The MQ2 Smoke sensor's pin is connected to D12.
- The digital output of the Vibration sensor is connected to D27.
- The Fire sensor is connected to D26.
- The Buzzer, serving as an output device, is connected to D25.
- The Green LED light, indicating system operation, is connected to D2.
- The Red LED light, used for alerts, is connected to D4.
- The voltage sensor's input pin is connected to D35 of the NodeMCU.
- A DC Jack is connected to the battery charger, along with an LED light that functions while charging.

To ensure continuous operation during power failures, a battery is incorporated into the system. The battery is connected to the battery charger, where the charger's output (OT) is connected to the IN pin of the Booster Module. The Booster Module then provides a stable 5V output, which is connected to the NodeMCU's Vin pin. However, the NodeMCU operates on 3.3V, so the 5V output is reduced accordingly. Each sensor receives the required 3.3V output from the NodeMCU. For both DC and AC power supply, a two-core wire is directly plugged into the system. A booster module is employed to maintain a stable current and provide the necessary 5V. The LED light used for the battery charger has a red and blue colour.

V. DEVELOPMENT METHODOLOGY

The development methodology used for this project was the waterfall methodology. In the waterfall methodology, the project progresses sequentially through distinct phases, with each phase building upon the previous one. The phases typically include requirements gathering, system design, implementation, testing, and deployment.[9] It's important to note that the waterfall methodology follows a linear approach, where each phase is completed before moving on to the next. This methodology works well for projects with well-defined requirements and stable environments, as changes or modifications to the hardware system are typically challenging to implement once the development progresses beyond a certain point.

A. Hardware Development

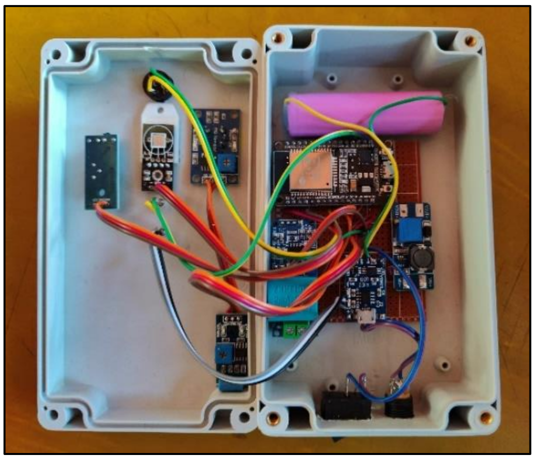


Figure 3 - Internal View

1. Identify the required sensors and components: Determine the specific sensors and hardware components needed for monitoring the server room, such as smoke sensors, temperature sensors, vibration sensors, etc.

2. Select the microcontroller: Choose a suitable microcontroller board, such as ESP-32 NodeMCU, which can handle the data processing and communication tasks required for the system.

3. Design the circuit: Create a circuit diagram that illustrates the connections between the microcontroller, sensors, and other components. Ensure proper power supply and data communication interfaces.

4. Prototype assembly: Assemble the hardware components on a breadboard or a dot board, following the circuit diagram. Use jumper wires and appropriate connectors to establish the necessary connections.

5. Testing and debugging: Verify the functionality of each component and the overall system. Test the sensors to ensure they are providing accurate readings. Debug any issues or errors that arise during testing.

6. Enclosure design: Design or select a suitable plastic enclosure box to house the hardware components securely. Consider factors such as ventilation, accessibility, and protection from external elements.

7. Final assembly: Transfer the prototype hardware to the chosen enclosure. Ensure proper positioning and alignment of the components. Securely mount the microcontroller, sensors, and other hardware elements inside the enclosure.

8. Power supply setup: Integrate the power supply components, such as the 18650 battery, battery charger (e.g., TP4056), and DC jack for external power adapter connection. Verify that the power supply setup is reliable and safe.

9. Documentation: Create documentation that includes the circuit diagram, component list, assembly instructions, and any other relevant details. This documentation will be helpful for future reference and troubleshooting.

10. Quality assurance: Conduct thorough testing and validation of the assembled hardware system. Ensure it meets the required specifications, functionality, and performance criteria.

11. Deployment: Install the assembled hardware system in the server room, considering factors such as optimal sensor placement and secure mounting. Connect the system to the required network or communication infrastructure.

12. Maintenance and monitoring: Regularly monitor the hardware system for any issues or malfunctions. Perform maintenance tasks such as sensor calibration, battery replacement, or firmware updates as needed to ensure the system's continuous operation.

B. System Requirements

Table 1 - Hardware Requirement

Used Hardware	**Requirements**
ESP-32 NodeMCU	Microcontroller board for data processing and communication with the network.
MQ-2 Smoke Sensor	To detect the presence of smoke or combustible gases in the server room.
ZMPT101B Sensor	For measuring the AC voltage in the server room.
DHT22 Sensor	To monitor the temperature and humidity levels in the server room.
IR Flame Detector Module	To detect the presence of flames or fire in the server room.
SW-420 Vibration Sensor	To detect any vibration or movement in the server room.
Mini Buzzer	To generate audible alerts or alarms when specific events occur.
3200mah 3.7V 18650 Battery	To provide power supply to the system.
MT3608 Mini DC-DC Step-Up/Boost Module	To regulate and boost the voltage for specific components.
TP4056 Lithium Battery Charger	For charging the 18650 battery.
Dot Board	For connecting and soldering the electronic components.
Switch	To manually control the system or certain functionalities.
LED Light	To indicate the system status or specific events.
DC Jack	To connect the DC power adapter for the external power supply.
Jumper Wire	For making electrical connections between the components.
Plastic Enclosure Box	To house and protect the hardware components.
DC Power Adapter	To provide external power supply to the system when needed.

Table 2 - Software Development Requirements

Technologies	**Requirements**	
	Platform IDE	**Programming Language**
IoT Development	Arduino IDE	C++
Mobile Application	Visual Studio Code	Framework – Flutter
Web Application		Programming – Dart
Database	Firebase	
Cloud Platform	Google IoT Platform	
Testing	Manual testing in hardware development. Usability testing.	
Version Control	GitHub	
Data Visualization	Syncfusion Flutter Charts	
Notifications Alert	ESP32_MailClient.h	

C. Software Development

1) Arduino Programming

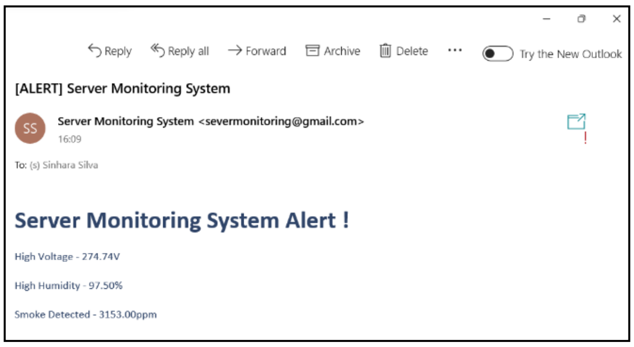
In this system, a mobile application and a web application have been developed alongside an Arduino-based IoT project. The Arduino code is created using the Arduino IDE, starting with the inclusion of necessary libraries and expanding the code from there. The operations of all the sensors used in the project are implemented within this code. The system's functionality is executed from the **void loop()** function. Each specific measurement is treated as an instance variable, and a corresponding function is created for it. To ensure efficient code organization, these functions are called from within the **void loop()** function. Let's take vibration measurement as an example.

To measure vibration, the following code is used inside the **void loop()** function: **long measurement = vibration()**. Within the **vibration()** function, the actual measurement is performed using **long measurement = pulseIn(vs, HIGH)**. This function utilizes the **pulseIn** function to detect the vibration state. Here, **vs** represents the pin used for vibration detection. By using **pulseIn**, the length of the pulse in both high and low states can be determined, as a long value is used to store the result.

The function for fire detection can be represented as follows: **bool measurementF = fire()**. This function utilizes the **digitalRead** function to determine whether a fire has occurred. The value of **measurementF** is not indicative of the fire itself, but rather whether a fire has been detected or not. Since a boolean value can only be either 1 or 0, it provides a binary representation of the fire status.

For the smoke sensor, the **analogRead** function is used. A threshold value is set to determine whether smoke has been detected. The voltage range accepted by the system lies between 195 and 264, and a corresponding function has been provided to check if the measured voltage falls within this acceptable range.

The system employs diverse sensors and sends email notifications upon detection of alarms or alerts, utilizing the **FirebaseESP32** library for data storage and retrieval from the Firebase Realtime Database, alongside the **ESP32_MailClient.h** library for the purpose of email transmission.



[ALERT] Server Monitoring System

Server Monitoring System <severmonitoring@gmail.com>
16:09

To: (s) Sinhara Silva

Server Monitoring System Alert !

High Voltage - 274.74V

High Humidity - 97.50%

Smoke Detected - 3153.00ppm

Figure 4 - Email Notification

1) Third-Party Components and Libraries

DHT and **EmonLib** are the two main libraries used in this project. The DHT library is primarily used to measure temperature and relative humidity. In the Arduino IDE, the code begins by including the DHT library and defining the pin connected to the DHT sensor. The sensor type, in this case, is DHT22. The DHT object is then initialized with the previously defined pin and type. The serial monitor is started at a baud rate of 9600 in the setup() function for debugging purposes.

The DHT sensor is initialized with the begin() function,

which requires a delay for the sensor to provide accurate readings. In this case, the delay for the DHT22 sensor is set to two seconds in the loop(). The temperature and humidity readings are obtained using the readHumidity() and readTemperature() methods, respectively. These values are stored as floats.

To obtain the temperature value in Fahrenheit, the code can use the formula float f = dht.readTemperature(true). The DHT library provides functions for temperature calculations in both Fahrenheit and Celsius. Finally, all the readings are displayed on the serial monitor.

Another library used in this project is **EmonLib**, which is used for energy measurement, specifically for voltage. As DC voltage is not measured in this server room, the **EmonLib** library is chosen for its reliability. The voltage value is given as **Vrms** in the **EmonLib.**

Additionally, the code includes a heat index library to calculate the heat index, which provides an indication of comfort in the environment. The heat index calculation considers humidity and temperature along with the system's measurements.

2) *Application Development*

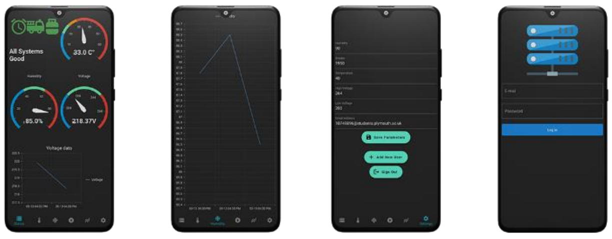


Figure 5 - Mobile Application Development

The **mobile application** and **web application** were developed using the **Flutter** framework and the Dart programming language. Graphical charts were displayed using the **syncfusion_flutter package,** and gauges were imported from the **syncfusion_flutter_gauges package.** These libraries and packages provide developers with the necessary tools and components to create interactive and visually appealing charts and gauges in the applications.**[11]** The Flutter framework, along with Dart, allows for cross-platform development, enabling the applications to run seamlessly on both mobile and web platforms.

VI. SYSTEM TESTING

The IoT-Based Server Monitoring System was subjected to two forms of testing: usability testing and manual testing.**[11]** User experience testing can be considered a subset of usability testing, aimed at evaluating the software application's ease of use and user-friendliness. Targeted usability testing was conducted, considering factors such as user-friendliness, control handling flexibility, and the application's ability to fulfil its objectives. Both a mobile application and a web application were developed for the system, and further testing was conducted to assess their responsiveness and load time.

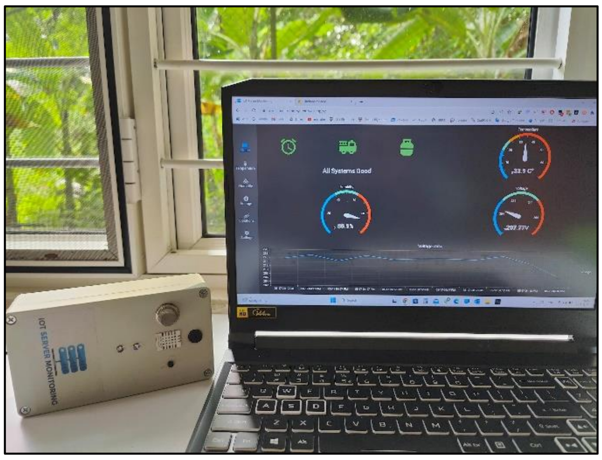


Figure 6 - System Testing

Manual testing, an essential aspect of the overall testing process, involves human testers executing test cases manually. This approach ensured the software's quality, functionality, and usability by comparing observed results with expected behaviour.

To simulate sensor data, a Python script was utilized for testing purposes, generating random numbers and timestamps. By effectively sending this generated data to a Firebase Realtime Database, the script provided a testing framework for simulating an IoT-based server room monitoring system, mimicking its operations in real-world conditions.

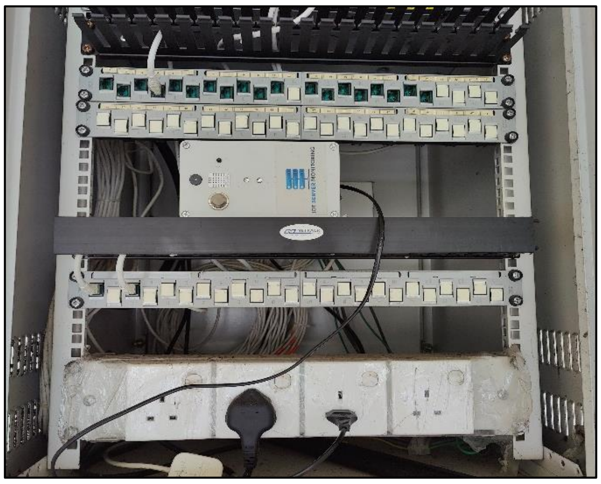
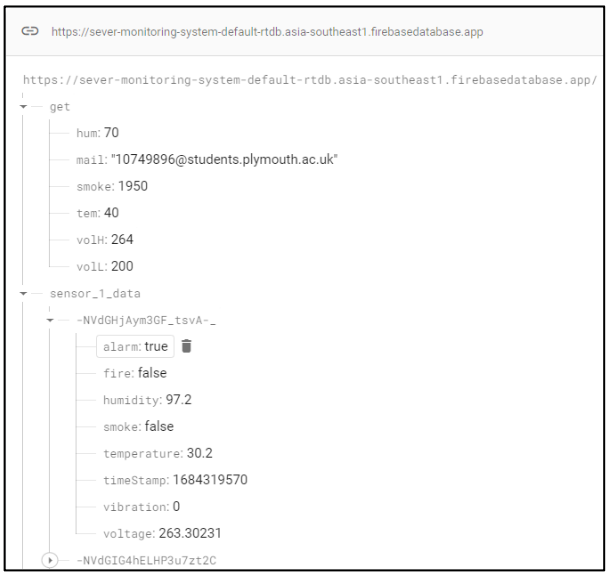


Figure 7 - Inside the Server Box

VII. RESULT AND DISCUSSION



```
https://sever-monitoring-system-default-rtdb.asia-southeast1.firebasedatabase.app

https://sever-monitoring-system-default-rtdb.asia-southeast1.firebasedatabase.app/
  get
    hum: 70
    mail: "10749896@students.plymouth.ac.uk"
    smoke: 1950
    tem: 40
    volH: 264
    volL: 200
  sensor_1_data
    -NVdGHjAym3GF_tsvA-_
      alarm: true
      fire: false
      humidity: 97.2
      smoke: false
      temperature: 30.2
      timeStamp: 1684319570
      vibration: 0
      voltage: 263.30231
    -NVdGIG4hELHP3u7zt2C
```

Figure 8 - Firebase Data Output

In this sample dataset, we have a get object that includes general information such as humidity **(hum)**, email address **(mail),** smoke level **(smoke)**, temperature **(tem)**, high voltage **(volH),** and low voltage **(volL).**

The **sensor_1_data** object contains data from sensor 1, including two entries with unique identifiers **(ID_1 and ID_2)**. Each entry includes information about the alarm status (alarm), fire detection (fire), humidity (humidity), smoke detection (smoke), temperature (temperature), timestamp (timeStamp), vibration level (vibration), and voltage level (voltage).

The provided dataset is from an IoT-based server room monitoring system. It includes information about various parameters measured in the server room at different time intervals. The dataset contains the following columns:

Table 3 - Data Set

sensor_1_data	Description	Value
ID	The unique identifier for each entry in the sensor data.	NVdGHjAym3GF_tsvA-
Fire	Indicates if a fire has been detected in the server room (true/false).	false
Humidity	The humidity level in the server room. (Typically expressed as a percentage.)	97.2
Smoke	Indicates if smoke has been detected in the server room (true/false).	false
Temperature	The temperature in the server room, usually measured in degrees Celsius	30.2
TimeStamp	The time when the data was recorded.	1684319570

Each row represents a specific measurement recorded at a particular time. The dataset captures the changes in humidity, temperature, and voltage over time, providing insights into the environmental conditions of the server room.

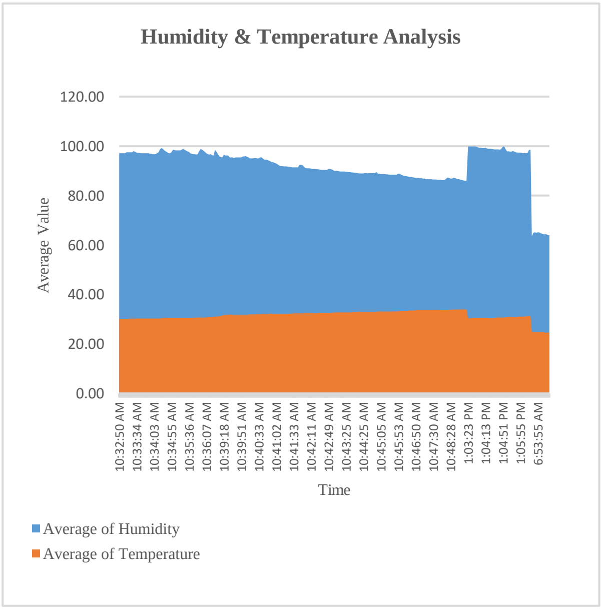


Chart 1 – Humidity and Temperature Analysis

Humidity and Temperature: Throughout the dataset, the humidity remains relatively constant at around **97%** to **99%** with occasional variations, while the temperature remains constant at around **30.2°C to 32.8°C** with minor fluctuations. This suggests a stable environment with consistent humidity and temperature levels.

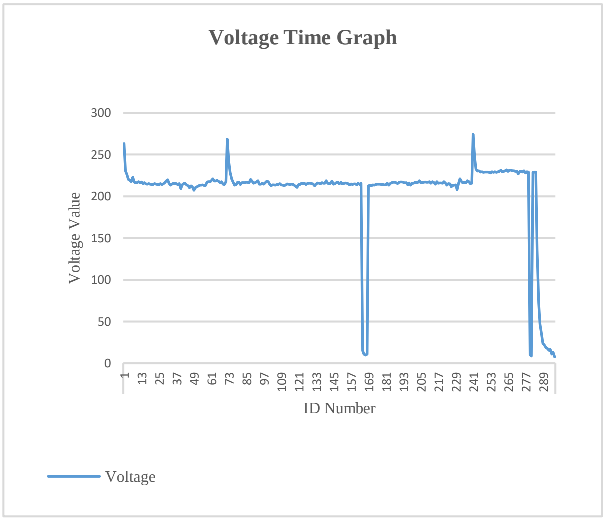


Chart 2 – Voltage Analysis with Time

Voltage: The voltage measurements vary throughout the dataset, ranging from approximately **206.97V** to **268.58V**. There are some instances where the voltage deviates significantly from the average, indicating potential fluctuations in the power supply or electrical system.

A. Recommendation

Based on the analysis of the dataset, I would recommend taking the following actions:

Monitor Voltage Fluctuations: Due to the significant variations in voltage measurements, it is advisable to closely monitor the electrical system. Any abnormal or inconsistent voltage readings should be investigated promptly to identify potential issues and prevent damage to equipment.

Verify Humidity and Temperature Sensors: Since the humidity and temperature readings remain relatively constant throughout the dataset, **it is recommended to verify the accuracy and calibration of the sensors periodically**.**[12]** This will help ensure the reliability of the collected data and the effectiveness of any environmental control systems.

For instance, it is recommended**[12]** to maintain the server room temperature within the range of **20°C to 22°C**. If there are any fluctuations or deviations from this optimal temperature range,**[13]** it should be promptly notified and addressed. Monitoring the environmental temperature closely helps prevent potential issues and ensures the stability of the server room's operating conditions.

VIII. CONCLUSION

In conclusion, the project has effectively fulfilled its objectives by successfully developing a fully functional Internet of Things (IoT)**[15][16]** device that presents considerable value within the industry. The project was completed within the designated time frame, demonstrating proficient project management and execution. An outstanding accomplishment of this work lies in the creation of a user-friendly web application and mobile application, which provide a convenient and intuitive interface for users to monitor and control the various functionalities of the IoT device. The integration of the Flutter framework with the Dart programming language, coupled with the utilization of graphical charts and gauges from the Syncfusion package, has significantly contributed to the development of visually appealing and interactive user interfaces.

By employing a range of sensors, including the MQ-2 Smoke Sensor, ZMPT101B Sensor, DHT22 Sensor, IR Flame Detector Module, and SW-420 Vibration Sensor, and incorporating additional components[14] such as the Mini Buzzer, the IoT device offers comprehensive monitoring capabilities for server rooms. The ESP-32 NodeMCU serves as the central microcontroller, facilitating data collection, processing, and communication functionalities.

Furthermore, the hardware development phase was meticulously executed, ensuring the selection of appropriate components, correct wiring, and efficient power management. The incorporation of a 3200mAh 3.7V 18650 Battery, MT3608 Mini DC-DC Step-Up/Boost Module, and TP4056 Lithium Battery Charger guarantee reliable and uninterrupted operation of the IoT device.

A notable feature of the system is its ability to periodically record data, allowing for the accumulation of valuable measurements over time. This recorded data can be utilized for subsequent analysis and graphing, enabling stakeholders to gain insights into the environmental conditions of the server room and make informed decisions based on the derived findings.

IX. REFERENCES

[1]. maheshyadav216 (2022) IoT Based Server Room Monitoring System, Hackster.io. Available at: https://www.hackster.io/maheshyadav2162/iot-based-server-room-monitoring-system-1ec820

[2]. Vidushi (2021) IoT based server room monitoring system - PsiBorg, PsiBorg. Available at: https://psiborg.in/iot-based-server-room-monitoring-system/.

[3]. Julia Borgin (2022) Common server issues and their effects on operations | TechTarget, Data Center. Available at: https://www.techtarget.com/searchdatacenter/tip/5-common-server-issues-and-their-effects-on-operations

[4]. Botta, A., de Donato, W., Persico, V., & Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56, 684-700. Available at: https://www.journals.elsevier.com/future-generation-computer-systems [Accessed 17 May 2023].

[5]. Hassan, Q. F. (2018). *Internet of Things: Challenges, advances, and applications.* CRC Press. Available at: https://www.crcpress.com/Internet-of-Things-Challenges-Advances-and-Applications/Hassan/p/book/9781498785869 [Accessed 18 May 2023].

[6]. Johnson, P. (2019). An Introduction to the Internet of Things (IoT). *Liaison Technologies*. Available at: https://www.liaison.com/blog/2019/11/14/introduction-internet-of-things/ [Accessed 19 May 2023].

[7]. Smith, M. (2020). IoT for Beginners: A Complete Guide. *Medium*. Available at: https://medium.com/iotforall/iot-for-beginners-a-complete-guide-3c0b36fbb6aa [Accessed 20 May 2023].

[8]. DHT11/DHT22 Sensor with Arduino | Random Nerd Tutorials (2019) Random Nerd Tutorials. Available at: https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/.

[9]. Joharji, G., 2016. *Linkedin.* [Online] Available at: https://www.linkedin.com/pulse/common-problems-server-rooms-ghareed-joharji/?articleId=6215116366253424640

[10]. Latif, M. U. M., Hossain, M. M., & Ali, M. A. (2019). Design and implementation of an IoT-based server room temperature monitoring system. In 2019 2nd International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (pp. 23-26). IEEE. https://doi.org/10.1109/ICASERT.2019.8889227

[11]. Ullah, R., Khan, M. A., Khan, A. U. R., & Ali, I. (2020). IoT-based server room environment monitoring and alert system. In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 727-731). IEEE. https://doi.org/10.1109/ICICCS48112.2020.9122479

[12]. Bélanger, F. and Carter, L., 2008. Trust and risk in e-government adoption. The Journal of Strategic Information Systems, 17(2), pp.165-176. Available at: https://www.sciencedirect.com/science/article/pii/S0963868708000217

[13]. Alvan Prastoyo Utomo, M. *et al.* (2019) ‘Server Room Temperature & Humidity Monitoring Based on Internet of Thing (IoT)’, in *Journal of Physics: Conference Series*. Institute of Physics Publishing. Available at: https://doi.org/10.1088/1742-6596/1306/1/012030.

[14]. Abouzeid, A., Rashed, A., El-Kassas, S., & El-Sayed, H. (2019). IoT-based smart server room monitoring and control system. In 2019 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 1207-1212). IEEE. https://doi.org/10.1109/ICMA.2019.8816745

[15]. Chen, S., Zhang, H., Shang, Q., & Hu, Z. (2020). IoT-based server room monitoring and control system for energy efficiency. In 2020 2nd International Conference on Frontiers of Artificial Intelligence and Statistics (pp. 54-59). ACM. https://doi.org/10.1145/3417316.3417331

[16]. Alzahrani, M., Alamri, A., Alsalih, W., & Alshahrani, M. (2018). IoT-based smart temperature monitoring system for server rooms. In 2018 IEEE International Conference on Communication and Signal Processing (ICCSP) (pp. 721-725). IEEE. https://doi.org/10.1109/ICCSP.2018.8460098