PROGRESS IN ENGINEERING APPLICATION AND TECHNOLOGY
e-ISSN: 2773-5303

PEAT

Vol. 6 No. 1 (2025) p. 478-488
https://publisher.uthm.edu.my/periodicals/index.php/peat

Development of Independent Industrial IoT (IIoT) Device for Closed Building Application

Loo Jun Xian[1], Azlina Bahari[1]*, Amirul Syafiq Sadun[1]*

[1] *Department of Electrical Engineering Technology, Faculty of Electronic Engineering Universiti Tun Hussein Onn Malaysia, Pagoh, Johor, 84600, MALAYSIA*

*Corresponding Author: lina@uthm.edu.my
DOI: https://doi.org/10.30880/peat.2025.06.01.050

Article Info

Received: 17 January 2025
Accepted: 05 February 2025
Available online: 30 April 2025

Keywords

ESP32,4G Module, GMS Module, Linux, Debian12, AWS Cloud, Elastic Cloud Computing(EC2), Relational Database Services(RDS), FS-MCore-F800,DHT22

Abstract

This thesis, titled "Development of Independent Industrial IoT (IIoT) Device for Closed Building Applications," explores the creation of a self-sustaining IoT device designed to monitor temperature and humidity in closed building environments. The primary goal is to develop an IoT device that operates independently of the building's Wi-Fi and power sources, ensuring uninterrupted data collection and transmission, particularly in environments with limited or unreliable local networks and power supplies.The device, constructed using an ESP32 microcontroller, a 3000mAh LiPo battery, an FS-MCore-F800 Series GSM module, and a DHT22 temperature and humidity sensor, leverages the GSM module to transmit data to the Node-Red platform and MySQL database. This eliminates the need for a local Wi-Fi network while enabling real-time monitoring. Testing demonstrated reliable, continuous sensing and transmission of temperature and humidity data, validating the device's effectiveness and consistent server communication.The successful implementation provides a proof of concept for autonomous sensors in industrial applications, particularly in scenarios requiring continuous monitoring without access to traditional power or network infrastructure. Insights gained from this research offer a foundation for advancing industrial IoT solutions.

1. Introduction

The rapid evolution of the Internet of Things (IoT) has brought about significant changes across various industries, enabling the interconnection of devices and facilitating seamless data exchange. IoT technology allows for real-time monitoring, automation, and data-driven decision-making, which are particularly valuable in industrial settings. The deployment of IoT in industrial environments, including warehouses and manufacturing plants, has been widely studied, demonstrating its effectiveness in improving operational efficiency and maintaining environmental conditions (Zare & Iqbal, 2020).

In industrial environments such as warehouses, factories, and storage facilities, maintaining optimal temperature and humidity levels is crucial for several reasons. In warehouses storing perishable goods, incorrect temperature and humidity can lead to spoilage, resulting in substantial financial losses. In manufacturing plants, specific environmental conditions are necessary to ensure the quality and safety of both products and personnel. Therefore, reliable environmental monitoring systems are essential to maintain these conditions.

Progress in Engineering Application and Technology Vol. 6 No. 1 (2025) p. 478-488

Traditional environmental monitoring systems in closed buildings typically rely on the building's existing infrastructure, such as Wi-Fi networks and wired power supplies. While effective in many scenarios, these systems have several limitations. Firstly, dependence on Wi-Fi networks can be problematic in areas with poor connectivity or where network stability is an issue. Network failures can lead to data gaps, making it difficult to maintain continuous monitoring. Secondly, relying on wired power sources limits the flexibility of deploying these systems in various locations within the building. Power outages or interruptions can compromise the functionality of monitoring devices.

To overcome these limitations, there is a growing interest in developing self-sustaining IoT devices that operate independently of the building's infrastructure. Such devices would be particularly useful in settings where traditional power and network access are unavailable or impractical. By using alternative communication methods and independent power sources, these devices can offer more reliable and flexible monitoring solutions.

The goal of the article is to is to design and develop an Independent Industrial IoT (IIoT) device specifically tailored for closed building applications. This device aims to monitor environmental parameters using multiple sensors and operates autonomously without reliance on conventional infrastructure such as Wi-Fi and wired power sources. Additionally, the project seeks to demonstrate the device's capability for continuous 24-hour monitoring and data collection. Furthermore, the gathered data will be analyzed using platforms like LabVIEW to explore correlations between monitored parameters, contributing to deeper insights into environmental management within closed industrial settings.

2. Methodology

The methodology involves designing and integrating hardware and software components to develop a self-sustaining IoT device. The proposed IoT device utilizes an ESP32 microcontroller, a DHT22 temperature and humidity sensor, an FS-MCore-F800 Series GSM module, and a LiPo battery, ensuring independent operation without reliance on existing building infrastructure. Studies have demonstrated the effectiveness of ESP32 in IoT applications due to its low power consumption and wireless connectivity capabilities (Babiuch, Foltynek, & Smutny, 2019). Additionally, GSM-based IoT communication has been proven effective for remote data transmission in environments with unreliable Wi-Fi networks (Murugan et al., 2021). To optimize energy consumption, the device implements sleep mode functionalities for both the ESP32 and the GSM module. Prior research has explored energy-efficient designs for IoT devices, highlighting the impact of deep sleep intervals on battery performance (Macheso et al., 2021). The device further utilizes AWS EC2 and RDS for cloud-based processing and storage, a method previously validated for scalability and efficiency in industrial IoT systems (Kodali & Valdas, 2018).Two operational modes, Sentry and Live, were implemented to support periodic and real-time data transmission, respectively. The software configuration employs AWS EC2 and RDS for cloud-based processing and storage, with Node-RED handling data flow and visualization. Incremental testing ensured reliable system functionality, from sensor integration to cloud communication, while dynamic DNS setup facilitated seamless remote access.

2.1 System Block Diagram (Hardware)

Fig. 1 shows a hardware block diagram that illustrates the hardware architecture of the final year project, demonstrating the integration of numerous components for the seamless collection, processing, and transmission of data. The system is fundamentally powered by a Li-Po battery equipped with a TP4056 charging module. The TP4056 module facilitates secure battery charging, while a VBAT to 5V power module transforms the Li-Po battery's output voltage (3.3V - 4.2V) into a consistent 5V supply. This regulated power is crucial for operating the ESP32 microprocessor and other associated hardware components.

The ESP32 microcontroller serves as the system's central processing unit. It interfaces with various sensors via its analogue and digital pins, gathers data, and oversees communication. The microcontroller interfaces with three principal sensors: the DHT22 Temperature and Humidity Sensor, the Sharp 2Y0A21 Distance Detection Sensor, and a Potentiometer. The DHT22 sensor is employed in Sentry Mode to intermittently record ambient temperature and humidity data for processing and analysis. The Sharp 2Y0A21 distance sensor is employed in Live Mode to measure distances to objects and showcase the system's capacity to relay real-time data. Likewise, the potentiometer functions as an analogue input device in Live Mode, so affirming the system's capacity to transmit real-time data.

A 4G Cellular Module is employed to enable distant data transmission by communicating with the ESP32 over a serial interface. The 4G module is essential for linking the hardware to the cloud. It acquires data

processed by the ESP32 and transfers it to a cloud computing service via WebSockets (TCP). This facilitates real-time surveillance and evaluation of sensor data, guaranteeing accessibility from distant locations.

The system functions in two separate modes: Sentry Mode and Live Mode. In Sentry Mode, the system emphasises frequent environmental surveillance utilising the DHT22 sensor to record temperature and humidity metrics. Live Mode showcases real-time data transfer utilising the Sharp 2Y0A21 distance sensor and the potentiometer. Both modes demonstrate the system's versatility and reliability in managing data collection and communication.

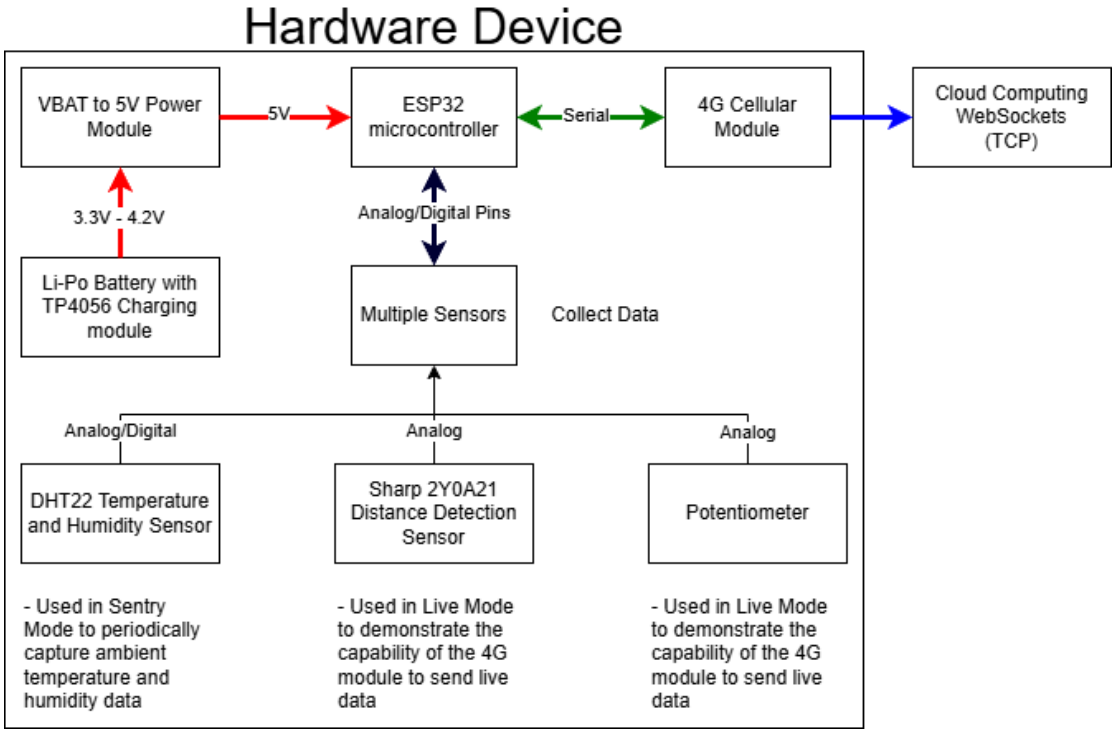

***Fig. 1*:** Hardware System Block Diagram

2.2 System Block Diagram (Software)

Fig. 2 shows the software block diagram depicts the cloud-based architecture utilised in the final year project, outlining the integration of diverse components and services for data handling, processing, and display. The architecture is fundamentally based on an AWS EC2 Virtual Server operating Debian 12 (free tier), which functions as the principal processing environment. AWS Relational Database Service (RDS) with MySQL (free tier) is deployed within the same Virtual Private Cloud (VPC) to manage and store the acquired data. A LAN connection within the same VPC enables uninterrupted read and write (R/W) activities between the EC2 instance and the RDS MySQL database.

The system interfaces with external hardware using WebSockets (TCP), facilitating real-time data transmission to the EC2 server. The hardware devices transmit data that is subsequently read and processed by Node-RED, a flow-based programming tool installed on the EC2 instance. Node-RED functions as the data intermediary, connecting the hardware, database, and dashboard components. The processed data can be stored in or retrieved from the MySQL database hosted on AWS RDS, facilitating dependable data storage and query operations.

The Node-RED Dashboard is utilised for presenting real-time or archived data. This dashboard functions as a visual interface that displays diverse information gathered by the system, encompassing real-time data streaming and records kept in the database. The graphical flow-based features of Node-RED facilitate seamless integration among data gathering, processing, and visualisation.

Supporting services are amalgamated to augment system functioning and administration. CasaOS offers a graphical user interface (GUI) for Linux operating systems, facilitating server configuration and administration. Crontab is employed to sustain a dynamic domain name service (DDNS), guaranteeing system accessibility despite fluctuations in IP addresses. Crontab also manages scheduled tasks, including regular database updates and system maintenance. Furthermore, Adminer, a streamlined MySQL database editor, enables database management activities such as query execution, table alteration, and data visualisation.

Progress in Engineering Application and Technology Vol. 6 No. 1 (2025) p. 478-488

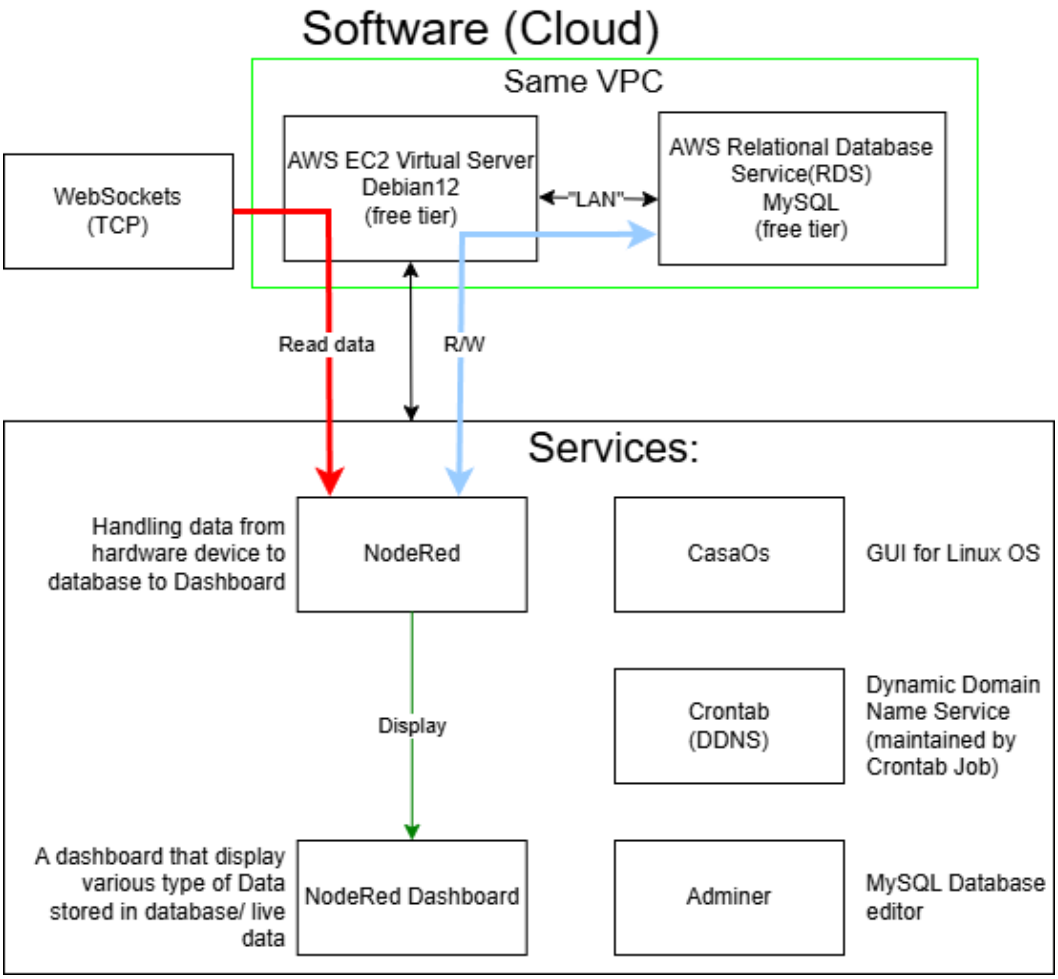

***Fig. 2*:** Software Block Diagram

2.3 Flowchart

The flowchart in **Fig. 3** illustrates the operating cycle of the hardware system, which functions in two unique modes: Sentry Mode and Live Mode. These modes govern the system's management of data gathering, processing, and transmission, while enhancing performance and energy efficiency.

The procedure commences with the system's initialisation. At starting, all sensors, including the DHT22 temperature and humidity sensor, the distance sensor, and the potentiometer, are initialised. The 4G Cellular Module initiates upon the application of power to the device. Upon initialisation, the ESP32 microcontroller assesses the operational mode to ascertain whether it will engage Sentry Mode or Live Mode, contingent upon established criteria.

In Sentry Mode, the system emphasises intermittent data surveillance while prioritising energy efficiency. Data is initially gathered from all accessible sensors, including temperature, humidity, distance, and potentiometer measurements. The gathered data is then organised into a structured string format, such as JSON, to facilitate transfer to the cloud server. The system subsequently confirms the connectivity status of the 4G Cellular Module to the cloud server using TCP. Should the connection fail, the system does ongoing verifications until the module establishes a successful connection. Upon connection, the prepared data is conveyed to the server via a hardware serial interface utilising the 4G module. To improve energy economy, both the ESP32 microcontroller and the 4G Cellular Module are transitioned into a low-power sleep mode for a specified length, thus minimising overall power usage during idle intervals.

Conversely, Live Mode prioritises ongoing, real-time data acquisition and transmission to facilitate applications necessitating instantaneous updates. Analogous to Sentry Mode, sensor data is gathered and retained in variables for analysis. The gathered data is subsequently processed into a string, conforming to a structured format like JSON, to ensure consistency throughout transmission. The system delivers formatted data in real-time to the cloud server over the 4G Cellular Module utilising TCP communication. This ongoing data transmission guarantees that real-time information is available for prompt observation and evaluation.

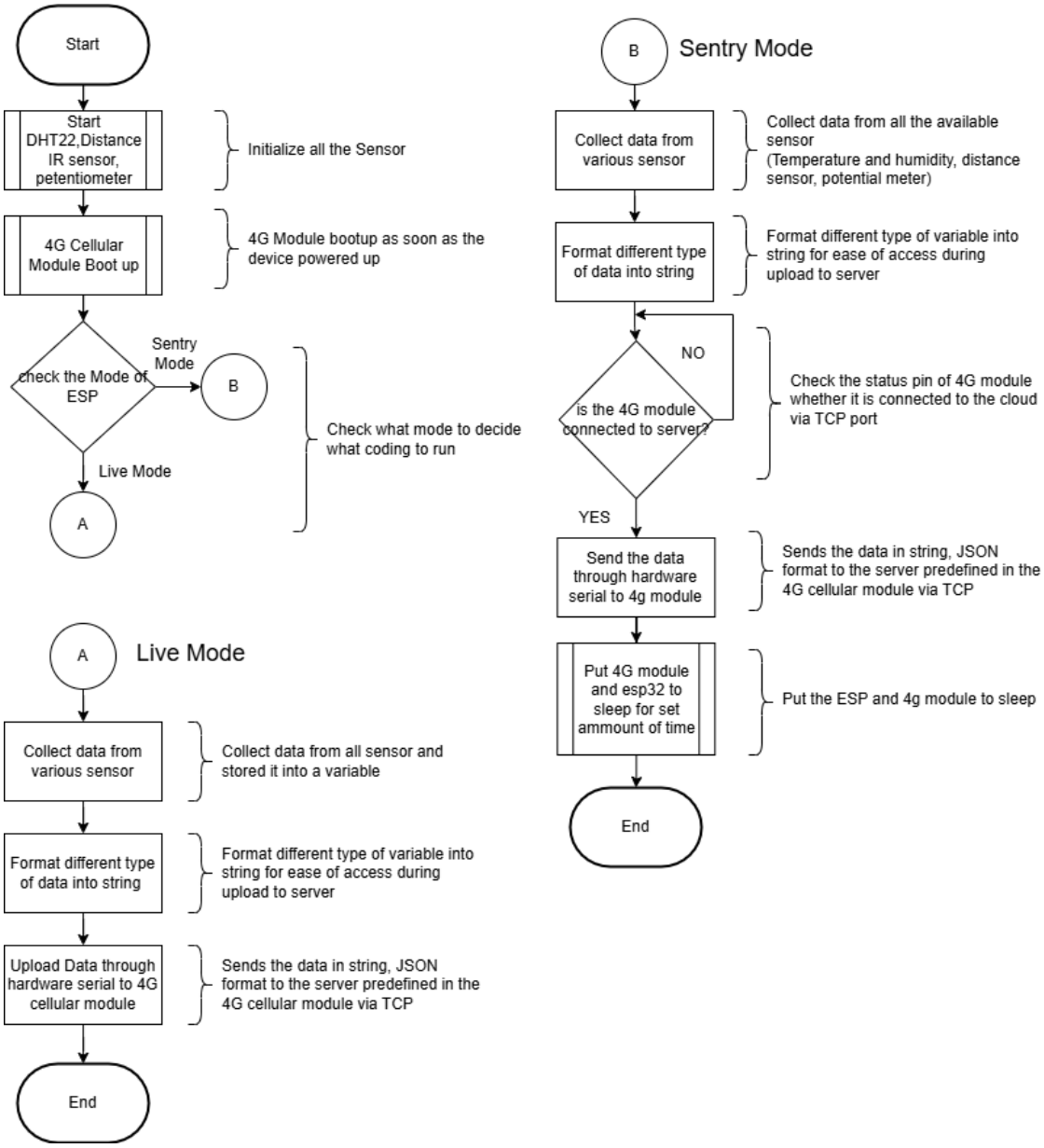

***Fig. 3*:** Flowchart

2.4 Schematic Diagram

Fig. 4 shows the schematic diagram that illustrates the whole hardware configuration of the project, emphasizing different modules that perform various functions. Each module has been carefully designed to guarantee the system functions effectively, prioritizing data collecting, processing, and transmission through the 4G module.

The Power Supply Mechanism provides the fundamental basis for powering the system. A Li-Po battery serves as the main power source, connected to a TP4056 module that manages battery charging and protection. The output from the TP4056 module is directed into a 5V power module, which regulates the voltage to 5V for the connected components. A 30 µF capacitor (C1) is integrated into the circuit to maintain consistent power delivery and reduce voltage fluctuations.

The ESP32 Microcontroller serves as the system's central processing unit, managing sensor data collecting, processing, and communication with the 4G module. It is powered by the 5V output from the power module and connects with multiple peripherals, including sensors and the 4G module. Certain GPIO pins on the ESP32 are designated for certain functions, including sensor input, mode regulation, and the management of the 4G module's sleep mode.

The 4G Module is a crucial component that enables wireless communication, allowing data transmission to the cloud server. The module functions on a 5V power source and interfaces with the ESP32 through serial communication using the TX2 and RX2 pins. A 470 µF capacitor (C2) is placed near the power input of the 4G

Progress in Engineering Application and Technology Vol. 6 No. 1 (2025) p. 478-488

module to maintain stability and avoid voltage fluctuations during operation. The module includes a sleep mode feature to enhance energy savings in specified operation modes.

A Sleep Control Mechanism is implemented to manage the sleep mode of the 4G module. This mechanism uses an optocoupler (U1, PC817) to ensure electrical separation between the ESP32 and the 4G module, thereby protecting the ESP32's GPIO pins, which are limited to a maximum voltage of 3.6V. The optocoupler is stimulated by a GPIO pin of the ESP32, grounding the sleep control pin of the 4G module and activating sleep mode. A 200 Ω resistor (R1) is incorporated to control the current passing through the optocoupler.

The Mode Switch Mechanism enables users to alternate between two operational modes: "Sentry Mode" and "Live Mode." A physical switch is linked to a GPIO pin (D2) on the ESP32, allowing the microcontroller to detect mode changes and operate accordingly. This feature allows users to alter the system's functionality according to specific requirements.

The system has many sensors to enable data collection. The DHT22 sensor detects temperature and humidity and is interfaced with GPIO pin D14. The SHARP 2Y0A21 distance sensor detects proximity and is linked to GPIO pin D27. Furthermore, a potentiometer functions as an analog input device, enabling users to manually modify particular parameters. The potentiometer is linked to GPIO pin D26 via a voltage divider circuit including a 10 kΩ resistor (R2) and a 1 kΩ variable resistor (RV1). All sensors function at 5V, and their outputs are connected to the ESP32 for processing afterwards.

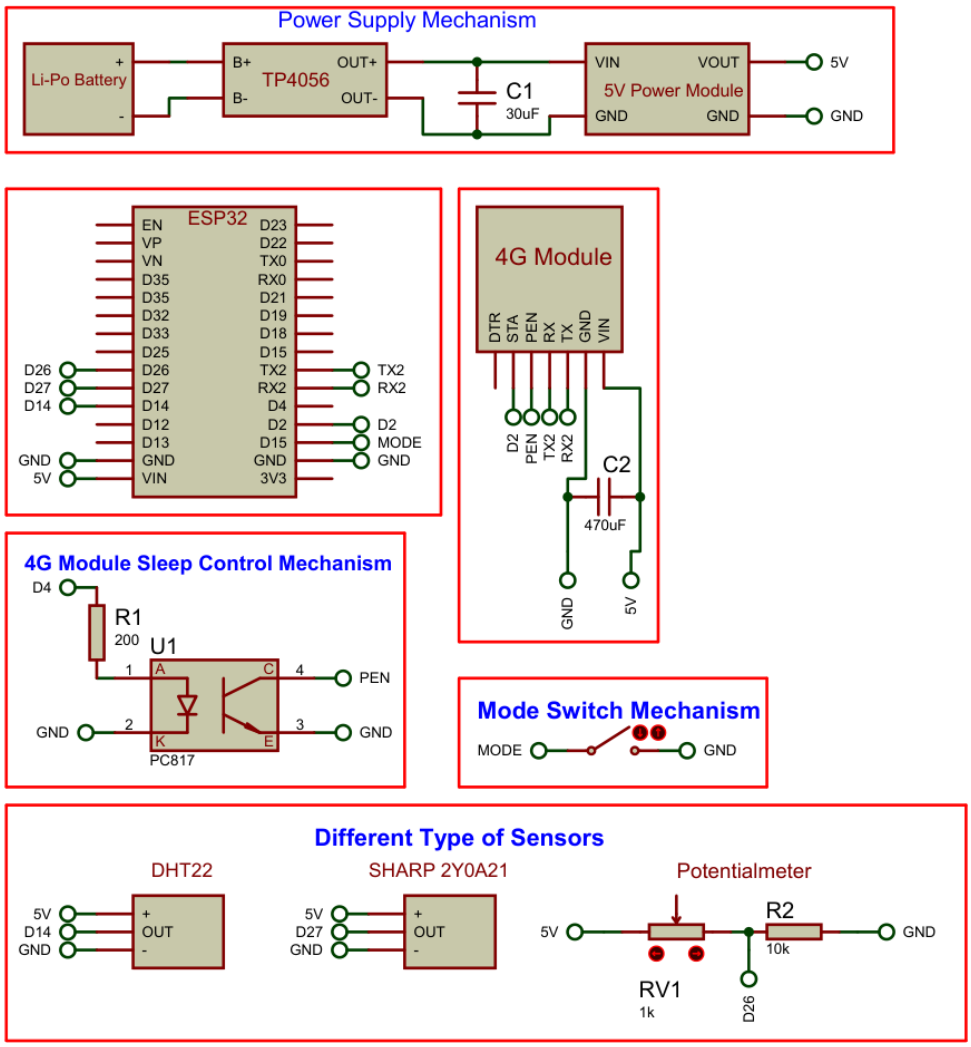

Fig. 4: Schematic Diagram

3. Result and Discussion

The section presents the results and discussion form implementing and testing the developed IIoT device. This section evaluates the device's performance across various parameters, including hardware functionality, software outputs, and overall system efficiency and performance.

3.1 Accuracy of DHT22

Table. 1&2 presents the accuracy of the DHT22 sensor over three trials under different situations. During the initial trial, executed on November 2 and 3 with the air conditioner set to 26°C in COOL mode, the DHT22 sensor continuously registered elevated temperatures, averaging around 28.0°C. This variation of about +2.0°C above the stated tolerance of ±0.5°C. The humidity readings averaged around 65%, indicating a reliable measurement while still diverging from optimal levels.

During the second trial on November 4 and 5, with the air conditioning temperature adjusted to 25°C, the sensor readings once more exceeded anticipated levels. The measured average temperature was roughly 27.0°C, resulting in a deviation of nearly +2.0°C, which exceeded the allowable tolerance. Humidity levels persisted about 65%, exhibiting uniform patterns across both iterations in COOL mode.

The third run, executed on November 6 and 7 in DRY mode, sustained the AC temperature at 25°C. The sensor documented somewhat reduced deviations, with temperatures averaging around 27.0°C, yielding a +2.0°C offset. The humidity levels decreased to around 59%, consistent with the functioning of the DRY mode.

The collected data demonstrates continuous accuracy trends while showing systematic deviations in the temperature measurements. This inconsistency was attributed to two principal sources. Manufacturing differences in the DHT22 sensor might lead to varying calibration standards, as the sensors are obtained from many manufacturers. Secondly, the degradation of the sensor's components affects its efficiency. The DHT22 employs a humidity-sensitive substrate to measure moisture levels, and the degrading of this substrate over time leads to errors. The manufacturer outlines recalibration protocols that need regulated temperature and humidity settings; however, these parameters were impractical to reproduce within the confines of this investigation. As a result, the sensor's offsets remained, highlighting the necessity for recalibration mechanisms or different sensors to improve accuracy. The accuracy of the DHT22 sensor was assessed under varying conditions, revealing minor deviations in temperature readings, consistent with previous studies on low-cost environmental monitoring solutions (Flores-Cortez et al., 2023). The reliability of the humidity measurements remained within an acceptable margin, aligning with prior findings on sensor performance.

***Table 1*:** DHT22 Result (1)

Run	Time/date	AC Temperature (°C)	AC Mode	DHT Reading Temperature (°C)	DHT Reading Humidity	Difference in Temperature (°C)	Within spec sheet tolerance? Tolerance= +-0.5°C
1_1	2 November 2024 1200-1800	26	COOL	28.03	65.08%	2.03	NO
1_1	2 November 2024 1800-0000	26	COOL	27.99	64.96%	1.99	NO
1_1	3 November 2024 0000-0600	26	COOL	28.01	65.00%	2.01	NO
1_1	3 November 2024 0600-1200	26	COOL	28.02	65.04%	2.02	NO
2_1	4 November 2024 1800-0000	25	COOL	26.99	64.95%	1.99	NO
2_1	5 November 2024 0000-0600	25	COOL	27.02	65.09%	2.02	NO

***Table 2*:** DHT Result (2)

Progress in Engineering Application and Technology Vol. 6 No. 1 (2025) p. 478-488

2_1	5 November 2024 0600-1200	25	COOL	26.98	64.98%	1.98	NO
2_1	5 November 2024 1200-1800	25	COOL	26.97	64.95%	1.97	NO
3_1	6 November 2024 1200-1800	25	DRY	27.02	59.01%	2.02	NO
3_1	6 November 2024 1800-0000	25	DRY	27.02	59.04%	2.02	NO
3_1	7 November 2024 0000-0600	25	DRY	26.96	58.94%	1.96	NO
3_1	7 November 2024 0600-1200	25	DRY	27.00	59.05%	2	NO

3.2 Operating Hour per Battery Charge

Table. 3 shows a result of different configuration of the sleep mode on ESP32 and 4G Module. On run 6_1 where the ESP32 is configured to enter deep sleep mode every 1 minute and the 4G module stays awake all the time, not even disconnect from the internet. On this run it successfully uploaded 2862 data according to the database. The calculation of up time in minuets is:

$$Up\ Time(Minute) = Total\ number\ of\ Successful \times Sample\ Interval$$
$$Up\ Time\ for\ run\ 6_1 = 2826 \times 1 = 2826\ minutes$$
$$2826\ minutes \div 60\ minutes = 47.1\ hours$$
$$47.1\ hours \div 24\ hours = 1.96\ days$$

Using this approach will guaranteed our result on performance of battery is calculated by all of those success intervals. If the result is calculated using this formula:

$$Up\ Time(Day) = End\ datetime - Start\ datetime$$

This approach will include the times where the device was failed to upload the data, which means the hardware is not contributing while consuming the battery. These should be excluded in this analysis.

On run 7_5 where the ESP32 enter deep sleep mode every 5 minutes and the 4G module maintain on all the time. So, the changes were time interval from 1 minute added to 5 minutes. This resulted a reduced sample size of 5 minute per sample, which means 20 sample per hours. Beside the reduced sample size, a uplift in battery performance is observed in Table 4.3 where its up time in days was 2.55 days. We can calculate the percentage of uplifted performance by using this formula:

$$Percentage\ Performance\ Changes = \frac{Final\ Value - Starting\ Value}{|Starting\ Value|} \times 100$$
$$Percentage\ Performance\ Changes = \frac{2.55 - 1.96}{|1.96|} \times 100 = 30.1\%$$

By increasing the deep sleep time of ESP32 gained a 30.1% uplift in up time of the device. On the data accuracy wise, its proven that increasing to 5 minutes does not make the data less accurate compare to 1 minute's interval. This is proven at the section before this on the Accuracy of DHT22

On run 8_1 where the ESP is configures as 1 minute's Deep sleep, 4G Module 1-minute sleep duration. The 4G Module is the most power consumption component in the whole device, hence bringing sleep function to 4G Module should have a significant impact on the device uptime. Based on the result in Table 4.3 we can observe a 1.05 day of effective up time. To maintain as many variables as possible this result should compare with result on run 6_1 since the difference on both of this run is the 4G Module sleep mode. By substituting into the Percentage Performance Changes:

$$Percentage\ Performance\ Changes = \frac{1.05 - 1.96}{|1.96|} \times 100 = -46.42\%$$

A negative percentage is obtained after the calculation which means this run is 46.42% worst that the 6_1 run.

On run 9_5 a reduction in performance was observed as well. The configuration for this run was ESP32 and 4G module have a 5 minutes of sleep interval. Hence this run's result will be comparing with run 7_5. By using the formula:

$$Percentage\ Performance\ Changes = \frac{1.08-2.55}{|2.55|} \times 100 = -57.64\%$$

A reduction performance of 57.64% is observed. Comparison Graph of all of this run will be enclosed on the next section, as well as the explanation of this phenomena.

The results demonstrated a significant improvement in battery performance when utilizing periodic deep sleep intervals, consistent with research on power optimization techniques for ESP32-based IoT devices (Moise, Svasta, & Ionescu, 2020).

Table 3: Result of Operating Hour per Battery Charge

Run	Esp32 sleep mode	4G Module sleep mode	Sleep duration (Minutes)	Total number of samples	Sample interval (Minutes)	Up time (Minutes)	Up time (Hour)	Up time (Day)
6_1	Deep Sleep	NONE	1	2826	1	2826	47.1	1.96
7_5	Deep Sleep	NONE	5	735	5	3675	61.25	2.55
8_1	Deep Sleep	YES	1	1510	1	1510	25.16	1.05
9_5	Deep Sleep	YES	5	312	5	1560	26	1.08

3.3 Data Integrity Analysis

Table. 4 below summarizes the accuracy of data over several experimental runs, each defined by different setups and situations. In Run 1_1, the anticipated number of entries was 5760; however, only 4582 entries were accurately recorded, leading to 1178 missing data and a success percentage of 79.61%. In Run 2_1, expected entries were 2880, with 2288 recorded, resulting in 592 missed records and a success percentage of 79.45%. Run 3_1 had an expected total of 2880 entries, with 2263 documented, leading to 617 missed entries and a success percentage of 78.50%. Run 4_5, executed with a reduced length and an alternative interval configuration, expected 2182 items, of which 2075 were successfully recorded, resulting in 107 missed entries and a success percentage of 95.10%. The table shows differences in data completeness among the runs, demonstrating the reliability of this particular 4G Module.

The methodology of this analysis was to obtain the uptime, expected records, missed records, and percentage of success. The up time was obtained by subtracting the end date-time with the start date-time.

$$UpTime = End\ DateTime - Start\ DateTime$$

For the expected records, it was calculated by converting the uptime into minutes and divided by the sleep interval of the device, since this determined the how often the device collects data.

$$UpTime = End\ DateTime - Start\ DateTime$$
$$Expected\ Records = \frac{3423}{1} = 3423\ expected\ records$$

For the missed records there is the subtraction between expected records and arrived records.

$$Missed\ Record = Expected\ Records - Arrived\ records$$
$$Missed\ Record = 3423 - 2725 = 698\ missed\ records$$

For the percentage of success there is a ratio of expected records and arrived records.

$$Percentage\ of\ success = \frac{Arrived\ Records}{Expected\ Records} \times 100$$
$$Percentage\ of\ success = \frac{2725}{3423} \times 100 = 79.61\%$$

Progress in Engineering Application and Technology Vol. 6 No. 1 (2025) p. 478-488

***Table 4*:** Data Integrity Analysis

Run	Start datetime	End datetime	Up Time	Expected records	Arrived records	Missed records	Percentage of success
1_1	2024-11-23 15:45:48	2024-11-26 00:48:03	2 days, 9 hours, 2 minutes, and 15 seconds.	3423	2725	698	79.61
2_1	2024-11-04 18:26:00	2024-11-06 03:32:24	1 day, 9 hours, 7 minutes, and 24 seconds.	1987	1581	406	79.57
3_1	2024-11-06 13:12:56	2024-11-07 22:08:51	1 day, 8 hours, 55 minutes, and 55 seconds.	1976	1580	396	79.96
4_5	2024-11-08 12:50:57	2024-11-09 14:39:57	1 day, 1 hour, and 49 minutes	310	295	15	95.16
6_1	2024-10-30 19:11:43	2024-11-01 23:31:29	2 days, 4 hours, 19 minutes, and 46 seconds.	3140	2826	314	90.00
7_5	2024-11-13 13:08:43	2024-11-15 01:34:52	2 days, 12 hours, 50 minutes	730	705	25	96.57
8_5	2024-11-27 23:00:51	2024-11-29 03:30:20	1 day, 4 hours, 29 minutes, and 29 seconds.	342	312	30	91.23

4. Conclusion

This project effectively illustrates the integration of cellular networks with IoT to create an autonomous Industrial IoT device for enclosed building applications. The objectives were accomplished: the gadget functions autonomously without reliance on external power or internet, monitors several sensors utilizing accessible microcontroller ports, and facilitates uninterrupted data collecting for over 24 hours. The cloud service conducts lightweight data analysis and presents customized information on a globally accessible Node-RED dashboard. The devices, driven by an ESP32 microcontroller, incorporate a 4G module, DHT22 temperature and humidity sensor, SHARP 2Y0A21 infrared sensor, potentiometer, Li-Po battery, and power management elements. It offers two modes: Sentry Mode for intermittent data collecting with sleep intervals and Live Mode for instantaneous data delivery. Node-RED manages and archives Sentry Mode data in a MySQL database, whereas Live Mode data is presented directly on the dashboard. The DHT22 sensor demonstrated a persistent temperature discrepancy of +2°C, although humidity measurements remained dependable. Data transmission success rates varied between 79.5% and 96.5%, while battery assessments revealed significant power requirements during cellular reconnections. These results confirm the device's efficacy and highlight opportunities for enhancement.

Numerous enhancements are proposed to augment the device's usefulness and scalability. Utilizing a power management circuit, such as the MCP73871 IC, facilitates seamless transitions between battery and external power sources, ensuring continuous operation and effective battery charging. Enhancing the battery capacity is advisable to prolong the device's operational duration, particularly during energy-demanding tasks. Adopting a professionally designed digital PCB will enhance reliability, optimize spatial efficiency, and streamline assembly for mass manufacturing. Furthermore, the incorporation of sophisticated communication technologies such as LoRa could facilitate the establishment of a sensor network, wherein a singular cellular-enabled device functions as a hub while LoRa nodes aggregate data, thereby enhancing network efficiency and broadening sensor deployment adaptability.

Acknowledgement

The author would like to thank the Faculty of Engineering Technology, University Tun Hussein Onn Malaysia for providing support along the way of completing the project paper.

References

[1] Faiz, R., Alam, N., Islam, S. R., Khan, S. N., & Hoque, M. R. (2022). IoT based solar powered automated fish feeding system. Agricultural Engineering International: CIGR Journal, 24(4).

[2] Flores-Cortez, O. O., Jimenez, C. P., Arévalo, F., López, R. L., Martínez, D. P., & Rafael, O. P. (2023). A Low-cost IoT Mobile System for Air Quality Monitoring in Developing Countries, a Study Case in El Salvador. https://doi.org/10.1109/smartnets58706.2023.10215961
[3] Macheso, P., Chisale, S., Daka, C., Dzupire, N., Mlatho, J., & Mukanyirigira, D. (2021). Design of Standalone Asynchronous ESP32 Web-Server for Temperature and Humidity Monitoring. https://doi.org/10.1109/icaccs51430.2021.9441845
[4] Moise, M. V., Svasta, P. M., & Ionescu, L. M. (2020). Implementation of a prototype air-quality detection network system with geolocation. https://doi.org/10.1109/estc48849.2020.9229789
[5] Murugan, K., Murugeswari, S., Parlapalli, V., Teja, C. M., & Triveni, K. (2021). Air pollution alert system using IoT with GPRS. 3C TecnologíA, 99–111. https://doi.org/10.17993/3ctecno.2021.specialissue8.99-111
[6] Elyounsi, A., & Kalashnikov, A. N. (2022). Predictive IoT Temperature Sensor. https://doi.org/10.3390/ecsa-9-13337
[7] Zare, A., & Iqbal, M. T. (2020). Low-Cost ESP32, Raspberry Pi, Node-Red, and MQTT Protocol Based SCADA System. 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). https://doi.org/10.1109/iemtronics51293.2020.9216412
[8] Kodali, R. K., & Valdas, A. (2018). MQTT Based Monitoring System for Urban Farmers Using ESP32 and Raspberry Pi. https://doi.org/10.1109/icgciot.2018.8752995
[9] Babiuch, M., Foltynek, P., & Smutny, P. (2019). Using the ESP32 Microcontroller for Data Processing. https://doi.org/10.1109/carpathiancc.2019.8765944
[10] Kanani, P., & Padole, M. (2020). Real-time Location Tracker for Critical Health Patient using Arduino, GPS Neo6m and GSM Sim800L in Health Care. https://doi.org/10.1109/iciccs48265.2020.9121128