Deggendorf Institute of Technology

Faculty of Electrical Engineering and Media Technology

*“Measurement and Transfer of Atmospheric Data via MQTT Protocole ”*

Master’s thesis in fulfillment of the requirements for the degree of:

Master of Science (M.sc.) Deggendorf Institute

of Technology

Submitted by: Examiner:

*Sabir Shah Prof. Dr.-Ing. Werner Bogner*

*12201169*

On:28 May 2024



Author’s Declaration

Name of student:

|  |  |
| --- | --- |
|  | |
|  | Prof. Dr. Werner Bogner | |
|  | |

Professor/Supervisor at DIT:

If external: Name of supervisor:

Master’s thesis topic:

Measurement and transfer of atmospheric data via MQTT protocol

1. I hereby declare that this Master’s thesis is my own work, has not been submitted for any other degree at any other university or institution, does not contain or use any sources or resources other than those referenced, and that all direct and paraphrased quotes have been duly cited as such.

Deggendorf

S

, (Date) (Signature student)

1. I hereby consent to my completed Master’s thesis being made available to a broader public through the library of the Deggendorf Institute of Technology.

Yes No

X

#### If yes:

I certify and warrant that I am the sole owner and holder of all rights to the Master’s thesis, including the right to use the tables, figures, illustrations, plans, or similar, submitted therein and that no third-party rights or claims, nor any statutory provisions, will be infringed as a result of its publication.

Deggendorf

S

(Date) (Signature student)

Only to be completed by the examiner if the author consents to the Master’s thesis being published.

Collection of a copy of the Master’s thesis by the library, including the lending of the copy, is hereby

endorsed

not endorsed.

Deggendorf

(Date) (Signature examiner)

Table of Contents

**[Abstract](#_Toc167282119)** [4](#_Toc167282119)

**[List of Abbreviations](#_Toc167282120)** [5](#_Toc167282120)

**[Acknowledgments](#_Toc167282121)** [5](#_Toc167282121)

**[Introduction](#_Toc167282122)** [6](#_Toc167282122)

**[History of IoT:](#_Toc167282123)** [7](#_Toc167282123)

**[Background](#_Toc167282124)** [8](#_Toc167282124)

**[Problem statement](#_Toc167282125)** [9](#_Toc167282125)

**[Objectives of your project](#_Toc167282126)** [9](#_Toc167282126)

**[System Integration:](#_Toc167282127)** [10](#_Toc167282127)

**[Node-RED Implementation:](#_Toc167282128)** [10](#_Toc167282128)

**[Ease of Programming:](#_Toc167282129)** [10](#_Toc167282129)

**[Data Visualization:](#_Toc167282130)** [10](#_Toc167282130)

[Literature Review 11](#_Toc167282131)

**[Methodology](#_Toc167282132)** [22](#_Toc167282132)

**[Hardware and Software](#_Toc167282133)** [22](#_Toc167282133)

**[Microcontroller( Arduino Nano 33 IoT](#_Toc167282134)** [22](#_Toc167282134)

**[Sensor(BME 280)](#_Toc167282135)** [24](#_Toc167282135)

[Features and Specifications of BME280 25](#_Toc167282136)

[Features 25](#_Toc167282137)

[Specifications 25](#_Toc167282138)

**[Measuring Data/MQTT protocol implementation](#_Toc167282139)** [27](#_Toc167282139)

[To install Node-RED for programming and receiving data, you can follow these steps: 28](#_Toc167282140)

[To install the Mosquitto MQTT broker on the Raspberry Pi, follow these steps: 28](#_Toc167282141)

[Step 1: Activating Mosquitto Repository 28](#_Toc167282142)

[Step 2: Installing Packages 29](#_Toc167282143)

[Step 3: Configuration 29](#_Toc167282144)

**[MQTT In Block:](#_Toc167282145)** [29](#_Toc167282145)

**[Gauge 1](#_Toc167282146)** [29](#_Toc167282146)

**[Gauge 2:](#_Toc167282147)** [29](#_Toc167282147)

[Gauge 3: 30](#_Toc167282148)

**[Results](#_Toc167282149)** [30](#_Toc167282149)

**[Data visulization:](#_Toc167282150)** [31](#_Toc167282150)

**[Comparison of different graph:](#_Toc167282151)** [32](#_Toc167282151)

**[Data Storage:](#_Toc167282152)** [33](#_Toc167282152)

**[Power Consumption of Arduino Nano 33 IOT:](#_Toc167282153)** [33](#_Toc167282153)

**[Using the formula](#_Toc167282154)** [35](#_Toc167282154)

**[Security Consideration:](#_Toc167282155)** [36](#_Toc167282155)

[Future Work and Improvements: 36](#_Toc167282156)

**[Cloud Integration:](#_Toc167282157)** [36](#_Toc167282157)

**[Security Enhancements](#_Toc167282158)** [36](#_Toc167282158)

**[Enhanced Encryption for Energy Systems:](#_Toc167282159)** [36](#_Toc167282159)

**[Energy Efficiency Considerations:](#_Toc167282160)** [36](#_Toc167282160)

**[Appendix](#_Toc167282161)** [37](#_Toc167282161)

[Bibliography 40](#_Toc167282162)

**Table of Figure**

[Figure 1 : A brief description for the the Arduino pins 23](#_Toc167282416)

[Figure 2 : It is Picture of the Sensor 24](#_Toc167282417)

[Figure 3: This is how it is connected 30](#_Toc167282418)

[Figure 4: Result 30](#_Toc167282419)

[Figure 5: Show the pressure temperature humidity value 31](#_Toc167282420)

[Figure 6 : Result Graph 32](#_Toc167282421)

[Figure 7 : Recorded Data 33](#_Toc167282422)

[Figure 8: Recorded Data 34](#_Toc167282423)

# **Abstract**

This thesis presents the development and implementation of an IoT-based environmental monitoring system using Arduino Nano 33 IoT for measuring atmospheric data, including temperature, humidity, and pressure. The collected sensor data is securely transmitted via MQTT protocol over a WiFi network to a Node-RED dashboard hosted on a Raspberry Pi. Real-time visualization and local storage of atmospheric data in .csv format on an SD card are achieved, allowing remote access to the data through an IP address without the need for an external monitor.Security and reliability are fundamental aspects of this system. The MQTT protocol is utilized with encryption mechanisms to ensure the secure transmission of data, maintaining its confidentiality and integrity during transfer. Additionally, system reliability is emphasized to ensure consistent and accurate data collection and transmission. In addressing energy considerations, the thesis includes an analysis of power consumption patterns associated with the IoT devices used in the monitoring system. This analysis aims to optimize energy efficiency and sustainability in IoT deployments. The study showcases the practical implementation of IoT technologies for environmental monitoring, demonstrating a comprehensive approach to data collection, transmission, visualization, and storage. By integrating security, reliability, and energy efficiency considerations, this research contributes to advancing IoT applications in environmental science and underscores the potential for scalable and sustainable IoT solutions in various monitoring contexts. This thesis contributes valuable insights into the design and deployment of IoT-based environmental monitoring systems, highlighting the importance of secure and reliable data transmission, user-friendly data visualization, and energy-conscious IoT device management. The findings presented offer practical implications for researchers and practitioners interested in leveraging IoT technologies for environmental data collection and analysis.

# 

# **List of Abbreviation**

Application Programming Interface

(API) 18

Arduino UNO development board

(AUDB) 14

Artificial Neural Network

(ANN) 18

Data Acquisition Systems

(DAQ) 14

Hybrid Shunt Active Harmonic Power Filter

(HSAHPF) 18

indoor environmental quality

(IEQ) 13

information technology

(IT) 8

Internet of Things

(IoT) 7, 12, 20, 21

Internet Protocol

IP 7

machine-to-machine

(M2M) 8, 16

Message Queuing Telemetry Transport

MQTT 17, 27

microelectromechanical systems

(MEMS) 8

operational technology

(OT) 8

Pacific Northwest National Laboratory

(PNNL) 14

Quality of Service

(QoS) 17

sick building syndrome

(SBS) 14

signal processing unit

(SPU) 14

Transport Layer Security

(TLS) 17

unique identifiers

(UID) 7

volatile organic compounds

(VOC) 13

WebSockets

(WSS) 16

# **Acknowledgments**

I extend my deepest gratitude to my esteemed professors for their unwavering support, guidance, and encouragement during my master's program. Their expertise and commitment have played an instrumental role in shaping both my research endeavors and academic development. Special appreciation is due to Professor Dr.-Ing. Werner Bogner and my supervisor, Michael Benisch, for their invaluable feedback and constructive criticism, which have significantly enhanced the quality of my work. Their mentorship and motivation have been pivotal to my achievements.I am profoundly thankful to my mother for her boundless love, unwavering support, and for instilling in me the values of perseverance and diligence. Without her sacrifices and encouragement, reaching this milestone would not have been possible.Lastly, heartfelt thanks go to my siblings for their constant presence, support, and belief in me throughout this journey. Their unwavering confidence has served as a constant source of strength and motivation. I am deeply appreciative of the contributions each of you has made to my personal and academic growth.

# 

# **Introduction**

The Internet of Things (IoT) refers to a network of interconnected devices, machines, objects, and even living beings that possess unique identifiers (UIDs). All have the capability to exchange data over a network without the need for direct human or computer interaction. These devices can range from computing devices to mechanical systems, and enable seamless communication and data transfer, leading to increased automation and efficiency in various domains. By leveraging IoT technology, the world becomes more interconnected, allowing for enhanced monitoring, control, and communication between diverse entities, ultimately shaping a more interconnected and intelligent ecosystem. In the Internet of Things (IoT), a “thing” can refer to various entities, such as a person wearing a smartwatch, livestock fitted with RFID transponders, or vehicles such as cars, trucks or motorbikes with tracking devices. Essentially, any object, whether natural or man-made, can be assigned an Internet Protocol (IP) address and possess the capability to transmit data over a network. This interconnectedness enables seamless communication and data exchange, enhancing monitoring, control, and functionality across a wide range of applications. The IoT empowers objects and beings with the ability to contribute to and benefit from the vast network of connected devices, creating a more intelligent and interconnected world. (1)

## **History of IoT:**

In 1999, the co-founder of the Auto-ID Center at MIT, Kevin Ashton introduced the concept of the Internet of Things (IoT) during a presentation to Procter & Gamble (P&G).His intention was to highlight the potential of radio frequency identification (RFID) technology to P&G’s senior management. Ashton cleverly named his presentation “Internet of Things” to align with the trending concept of the internet during that time. Additionally, MIT professor Neil Gerstenfeld’s book, titled “When Things Start to Think,” published in the same year, provided a clear vision of the direction in which the IoT was heading, even though it didn’t explicitly use the term.These early contributions laid the foundation for the development and recognition of the IoT as a transformative concept.While Kevin Ashton is credited with popularising the term “Internet of Things” in his presentation in 1999, the concept of connected devices has been present since the 1970s. Back then, it was referred to as the “embedded internet” or “pervasive computing.” These early ideas laid the foundation for the interconnected world we see today, where devices are seamlessly integrated into our daily lives, sharing data and enabling new possibilities. Ashton’s contribution brought attention to this evolving concept and helped shape the narrative around the Internet of Things. Today, the Internet of Things (IoT) has emerged through the convergence of wireless technologies, microelectromechanical systems (MEMS),microservices, and the internet. This convergence has played a crucial role in breaking down the barriers between operational technology (OT) and information technology (IT). As result, unstructured data generated by machines can now be analyzed to extract valuable insights and drive continuous improvements.The integration of these technologies has paved the way for a more interconnected and data-driven ecosystem. In the early 1980s, a notable example of an internet appliance was a Coke machine located at Carnegie Mellon University.This machine was connected to the web, allowing programmers to remotely check its status. They could easily determine if the machine was stocked with cold drinks, saving them a trip if they were not available. This early instance showcased the potential of connecting everyday objects to the internet and accessing information remotely, foreshadowing the broader concept of the Internet of Things. IoT emerged from the concept of machine-to-machine (M2M) communication, which involves the interconnection of devices through a network without human intervention. M2M technology enables devices to connect to the cloud, enabling remote management and data collection. It forms the foundation of IoT by facilitating the seamless exchange of information and enabling devices to operate intelligently and autonomously represents the advancement of M2M technology, forming a vast network of intelligent devices that connect individuals, systems, and various applications to gather and exchange data.M2M serves as the underlying connectivity framework that enables the seamless integration and communication between these devices, forming the foundation of IoT. (2)

## **Background**

Information on IoT and its applications in environmental monitoring: The integration of the Internet of Things (IoT) has revolutionized the way we interact with and monitor our environment. IoT facilitates the monitoring of crucial environmental phenomena through the deployment of sensor-equipped devices. These devices collect data, which is then wirelessly transmitted to storage servers, typically cloud-based, where it is analyzed and presented in a meaningful format. This data can be accessed via various user interfaces, such as mobile and web applications, enabling informed decision-making based on real-time environmental insights.Traditional notions of end users and servers in the Internet landscape are evolving with the emergence of IoT. In this new paradigm, devices and objects themselves become hosts, giving rise to the term "Internet of Things." These devices, equipped with sensors, capture and transmit data on various environmental parameters such as temperature, pressure, humidity, sound, emissions, and even vital signs. Environmental monitoring systems built on IoT principles encompass data collection through sensor networks and subsequent analysis for both short-term interventions, like remote heating or cooling control, and long-term data interpretation and trend analysis.The proliferation of IoT has led to numerous applications across different sectors, including smart power stations, smart cities, wearable devices, and home automation. These applications heavily rely on the collection of sensor data, its processing by microcontrollers, and its transmission via wired or wireless means to storage platforms like the cloud. Recent research efforts have focused on developing cost-effective IoT solutions using platforms such as Raspberry Pi and Arduino. These endeavors have resulted in practical implementations ranging from home automation to environmental monitoring systems.One significant aspect of IoT-enabled environmental monitoring is the utilization of wireless sensor networks. However, challenges such as energy consumption, network congestion, data loss, and reliability issues need to be addressed. Different wireless technologies like Bluetooth, Zigbee, Wi-Fi, GSM, MQTT and LoRa offer varying trade-offs in terms of energy efficiency, data transfer rates, and range, catering to diverse application requirements.In agriculture, IoT-based solutions have shown promise in water conservation and crop management. Soil moisture sensors, combined with camera and sensor networks, enable farmers to monitor and control irrigation systems remotely, optimizing water usage and improving crop yield. The integration of IoT technologies in agriculture holds potential for addressing challenges related to water scarcity and sustainable farming practices. (1) Cunin, McGrath, and MacNamee showcase a comprehensive approach to environmental monitoring using IoT technology, emphasizing the importance of sensor modules, wireless communication, data management, and user interface design. By deploying sensor modules across the University of Limerick campus, the team demonstrates the feasibility of real-time monitoring of various environmental parameters such as pollution, humidity, temperature, and luminosity. This initiative not only serves the immediate goal of environmental surveillance but also serves as a valuable educational platform for students to delve into IoT application development, requiring interdisciplinary skills in software development, data analysis, and hardware systems. The authors' emphasis on energy efficiency, durability, and scalability underscores the project's potential for broader applications beyond the university campus, laying the groundwork for future environmental monitoring initiatives on a larger scale. (2)

## **Problem statement**

In the realm of atmospheric data collection and analysis, there persists a demand for a comprehensive system that seamlessly integrates measurement, wireless transmission, visualization, and storage while ensuring reliability, security, and energy efficiency. Although IoT technologies offer promising solutions, existing implementations often face challenges in terms of integration, coding complexity, visualization accessibility, and real-time data storage.Traditional approaches to IoT system development typically involve extensive programming, potentially leading to complexity and errors. However, leveraging Node-RED—an intuitive visual programming tool—provides a solution to this challenge. With its graphical interface and a wide array of pre-built functions, Node-RED simplifies the development process, allowing users to create intricate IoT workflows through drag-and-drop operations. This eliminates the need for manual coding, significantly reducing development time and potential errors while enhancing system flexibility and scalability. Moreover, Node-RED offers a diverse set of functions tailored for IoT applications, including data parsing, protocol conversion, and integration with various services and platforms. This versatility enables developers to tailor their solutions to specific requirements without extensive customization or external libraries, further streamlining the development process and enhancing system robustness.Additionally, Node-RED facilitates effortless data visualization through customization dashboards, which can be accessed via a web browser using the system's IP address. This accessibility ensures that stakeholders can easily monitor and interpret atmospheric data in real-time, fostering informed decision-making and enabling timely responses to environmental changes.The ability to store data in real-time enhances the system's utility and analytical capabilities. By continuously capturing and archiving atmospheric data to a secure location, stakeholders can conduct comprehensive analyses, identify trends, and derive actionable insights to support various applications, including environmental monitoring, research, and decision support.However, despite the advancements facilitated by Node-RED and IoT technologies, challenges persist in ensuring the secure and reliable transmission of data, optimizing energy consumption to prolong system lifespan, and addressing potential scalability issues to accommodate future expansion and integration with emerging technologies.Therefore, this thesis aims to address these challenges by designing and implementing an integrated IoT system for atmospheric data collection, transmission, visualization, and storage. Through the utilization of Arduino Nano 33 IoT and Raspberry Pi, coupled with the power of Node-RED, the research endeavors to develop a robust, efficient, and user-friendly solution that meets the diverse needs of stakeholders while contributing to the advancement of IoT-based environmental monitoring and analysis.

## **Objectives of your project**

The overarching goal of this project is to design, develop, and implement an integrated IoT system for atmospheric data collection, transmission, visualization, and storage. Leveraging Arduino Nano 33 IoT, Raspberry Pi, Node-RED, and MQTT communication, the project aims to address key challenges in existing solutions while enhancing reliability, security, and energy efficiency. The specific objectives include:

### **System Integration:**

* Integrate Arduino Nano 33 IoT and Raspberry Pi into a cohesive system for seamless data flow.
* Establish MQTT communication protocol between devices to enable reliable and secure data transmission.

### **Node-RED Implementation:**

* Utilize Node-RED as the primary development platform for creating IoT workflows.
* Explore the vast library of pre-built functions in Node-RED to streamline development and minimize coding requirements.

### **Ease of Programming:**

* Demonstrate the ease of programming with Node-RED through graphical interface and drag-and-drop functionality.
* Eliminate the need for extensive manual coding, reducing development time and potential errors.

### **Data Visualization:**

* Develop customizable dashboards using Node-RED for intuitive visualization of atmospheric data (temperature, pressure, and humidity).
* Enable stakeholders to access real-time data visualization via a web browser using the system's IP address.

**Real-Time Data Storage:**

* Implement mechanisms for real-time data storage, ensuring continuous capture and archiving of temperature, pressure, and humidity data.
* Store data locally in a .csv file on the Raspberry Pi to facilitate easy access and analysis.

**Operational Autonomy:**

* Design the system to operate without the need for an external monitor or display, ensuring autonomous functionality.
* Enable remote access to system functionality and data visualization through the web-based interface.

**Energy Considerations:**

* Investigate energy-efficient strategies to optimize power consumption and prolong system lifespan.
* Implement measures to minimize energy usage during data collection, transmission, and storage processes.

**Security and Reliability:**

* Ensure secure and reliable transmission of data over wireless networks, utilizing encryption protocols and authentication mechanisms.
* Implement robust error handling and data integrity checks to mitigate potential risks and ensure data reliability.

**Scalability and Future Integration:**

* Design the system with scalability in mind to accommodate future expansion and integration with emerging technologies.
* Explore possibilities for integrating additional sensors or functionalities to enhance the system's capabilities over time.

By achieving these objectives, the project aims to deliver a robust, efficient, and user-friendly IoT solution for atmospheric data monitoring and analysis, specifically focusing on temperature, pressure, and humidity measurements. This contribution will advance environmental monitoring, research, and decision support efforts.

# Literature Review

IoT environmental monitoring leverages IoT technology to collect real-time data on environmental parameters such as air and water quality. This data aids businesses in making informed decisions to mitigate their environmental impact and adhere to regulations. The process involves four key steps: observation, analysis, storage, and action. Various devices like sensors, data loggers, GIS, and remote monitoring systems are employed for monitoring purposes. Benefits include enhanced understanding of the environment, improved efficiency, sustainability, and business growth opportunities. Use cases encompass air and water quality monitoring, energy management, commercial farming optimization, toxic gas detection, and animal conservation efforts. Caburn Telecom provides IoT solutions tailored for environmental monitoring, facilitating businesses in harnessing IoT's potential for sustainability initiatives. They offer a range of IoT services to ensure optimal coverage and security for connected sensors. Through IoT-based environmental monitoring, businesses can proactively address environmental challenges, reduce their ecological footprint, and contribute to conservation efforts. Caburn Telecom stands as a partner in enabling businesses to deploy effective and efficient IoT solutions for environmental monitoring needs.(3) IoT-based environmental monitoring involves the continuous collection of data using sensors and connected devices to measure various physical properties like temperature, moisture, and water levels. These intelligent devices process data using edge computing technology and transmit critical information to the cloud for further analysis or action. By detecting abnormalities or specific conditions, these systems can trigger alerts and automated processes, thus acting as a proactive watchdog for environmental conditions. This trend towards green technology is supported by IoT, aiding applications across energy systems, agriculture, water management, and environmental remediation. With real-time data insights, businesses can proactively monitor equipment, reduce waste, increase sustainability, and prevent disasters. The four essential components of IoT-based environmental monitoring include monitoring the environment, measuring data, cataloging data, and providing actionable insights. Use cases range from water quality monitoring and air quality monitoring to energy management and toxic gas detection, demonstrating IoT's potential for environmental sustainability and conservation efforts. Digi offers IoT solutions for environmental monitoring, providing tools for analysis, contaminant detection, and energy conservation to reduce carbon footprint. (4) Introduces an integrated information system (IIS) that merges Internet of Things (IoT), Cloud Computing, Geoinformatics, and e-Science for environmental monitoring and management, with a focus on climate change and its ecological impacts. It employs multi-sensors and web services for data collection, utilizing both public and private networks for data transmission. Key technologies include real-time operational database (RODB), extraction–transformation–loading (ETL), on-line analytical processing (OLAP), and representational state transfer/Java database connectivity (RESTful/JDBC). The middleware layer incorporates Application Program Interfaces (APIs), facilitating data sharing and processing. The study presents a case study on regional climate change in Xinjiang, showing trends in air temperature and precipitation over the last 50 years and their correlation with ecological indicators. The paper underscores the effectiveness of the IIS in improving monitoring processes and decision-making, providing a prototype for environmental management in the era of big data and IoT. (6) A study on environmental monitoring, highlighting the significance of this field due to climate changes. It discusses the use of remote sensing and wireless sensor networks, along with recent advancements like the Internet of Things (IoT), cloud computing, and cyber-physical systems, for data transmission and management. The study introduces three IoT-based wireless sensor systems for environmental monitoring: one using UDP-based Wi-Fi communication, another employing Wi-Fi and HTTP, and a third using Bluetooth Smart. These systems allow data recording at remote locations and visualization from any Internet-connected device, facilitating monitoring over large geographical areas. The paper describes the development details, differences, and similarities between these systems, analyzing their feasibility for environmental monitoring applications based on factors like energy autonomy, ease of use, solution complexity, and Internet connectivity. The abstract also lists index terms related to the study's focus areas, including Bluetooth, energy harvesting, IEEE 802.11 standards, IoT, and low-power electronics.(7) An IoT-enabled Environmental Monitoring System tailored for smart cities, focusing on temperature, humidity, and CO2 levels. It comprises transmitter and receiver nodes, along with a LabVIEW-based GUI for data visualization and logging. Additionally, an Android app enables remote monitoring. By utilizing low-power sensors, microcontrollers, and wireless transceivers, the system ensures efficient resource management. Testing in Gandhinagar, India, demonstrated its effectiveness across various environmental conditions. The system's reliability and power consumption were evaluated, indicating areas for future improvement. Overall, this comprehensive approach contributes to advancing IoT applications in urban environments, enhancing resource utilization and citizen well-being. (8) A comprehensive data acquisition system designed to monitor and control greenhouse environmental conditions to optimize plant growth. Utilizing an Arduino Mega 2560 board, the system integrates multiple sensors for temperature, humidity, soil moisture, CO2 concentration, and lighting, which are critical for maintaining optimal growing conditions. The hardware setup, including DHT11 for temperature and humidity, soil moisture sensors, photosensitive light sensors, and MG-811 for CO2, feeds data into the Arduino, which processes and transmits it to a computer via USB. The software component leverages LabVIEW for its graphical user interface, allowing real-time data visualization and control. LabVIEW's Interface for Arduino Toolkit facilitates seamless communication, enabling precise adjustments to environmental parameters. By continuously monitoring these factors and employing actuators to modify conditions, the system aims to enhance greenhouse productivity. This integration of Arduino's flexible hardware and LabVIEW's powerful data processing offers an effective and user-friendly solution for advanced greenhouse management. (9) The development of a cost-effective, Arduino-Based Data Acquisition System (ABDAS) designed for experimental aerodynamics research. Created with an Arduino Mega 2560, ABDAS provides an affordable alternative to high-priced commercial systems, like the National Instruments NI USB-6210 Data Acquisition System (NIDAS), commonly utilized at the Instituto Universitario de Microgravedad “Ignacio Da Riva” (IDR/UPM) of the Universidad Politécnica de Madrid. ABDAS aims to deliver reliable, multi-channel voltage measurements ranging from 0-6 V, with a minimum accuracy of 10 mV and a sampling rate of at least 500 Hz. Supporting five measurement channels, the system is versatile for various experiments, from thermocouple temperature readings to dynamic pressure recordings in wind tunnels. Designed for simplicity and ease of use, ABDAS features intuitive hardware and software interfaces. Voltage dividers extend the input range to 7.5 V, with calibration ensuring precise measurements. The system is notably cost-efficient, with a total budget of around, potentially reducible to €82 without the offset feature. The software, written in the open-source Arduino language, is easily modifiable for future updates. ABDAS’s performance was validated through two testing campaigns: a direct comparison with NIDAS measuring sine waves at various frequencies generated by a Hewlett Packard 33120A waveform generator, and a calibration of THIES First Class cup anemometers in the S4 wind tunnel at IDR/UPM, following MEASNET standards. This calibration established the transfer function between the anemometers' output frequency and wind speed, demonstrating ABDAS’s precise measurement capabilities. The results highlight ABDAS’s potential for diverse research and educational applications, particularly within engineering programs where cost-effective, accurate measurement systems are essential. The study concludes by emphasizing ABDAS’s benefits, including its low cost, user-friendliness, and high accuracy, making it an excellent tool for both academic and research settings. (10). This paper introduces a portable, continuous measurement toolbox for indoor environmental quality (IEQ)monitoring and performance assessment, addressing the high costs and inflexibility associated with traditional data acquisition systems. The toolbox incorporates various sensors to measure temperature, relative humidity, illuminance, CO2, volatile organic compounds (VOCs), PM2.5, and occupancy. Arduino Uno boards serve as the data acquisition hardware, interfacing with the sensors. Communication between the Arduino boards and a central computer is facilitated by ZigBee protocol, with each Arduino board connected to an XBee device for wireless data transmission to an XBee receiver. The toolbox leverages the open-source, agent-based software platform VOLTTRON for data communication and analysis. Developed by the Pacific Northwest National Laboratory (PNNL), VOLTTRON enables real-time monitoring and cloud-based data storage, enhancing the system's functionality. The toolbox's data collection capabilities were calibrated against a precise data acquisition card to ensure measurement accuracy. Experiments conducted in an open computer lab within a commercial building validated the system's effectiveness in assessing thermal comfort, indoor air quality (IAQ), and lighting performance. IEQ significantly impacts occupants' comfort, health, productivity, and overall quality of life. Poor thermal comfort and insufficient indoor air quality are primary sources of dissatisfaction and health issues in commercial buildings, contributing to sick building syndrome (SBS) and reduced worker productivity. Previous studies have developed IEQ monitoring devices with varying degrees of success and cost-efficiency. For example, continuous IEQ monitoring has been employed in senior centers, commercial office buildings, and academic institutions, utilizing diverse sensors and data loggers. However, these systems often face challenges related to high costs, limited flexibility, and complex installation processes. The presented toolbox addresses these challenges by utilizing cost-effective Arduino-based hardware and the flexible, open-source VOLTTRON software. The system's wireless capability, enabled through a ZigBee mesh network, allows for easy installation and scalability, eliminating the need for extensive wiring. This setup not only reduces installation time and labor costs but also enhances the system's adaptability to various monitoring scenarios. Wireless data transmission ensures that data from multiple sensors can be collected and analyzed in real time, providing a comprehensive assessment of IEQ. The paper emphasizes the advantages of the newly developed toolbox over existing systems. By replacing commercial data loggers with Arduino devices and integrating multiple sensors wirelessly, the toolbox reduces costs and increases flexibility. Additionally, the use of VOLTTRON for online monitoring and cloud-based data storage offers significant improvements in data accessibility and analysis capabilities. The study demonstrates the toolbox's reliability and robustness through a case study, highlighting its potential for widespread application in building performance monitoring. The toolbox's architecture, calibration process, and application in a real-world scenario are detailed in the paper. Calibration involved addressing errors in the Arduino-based data logger, primarily focusing on the analog to digital converter (ADC). The case study involved continuous monitoring of IEQ parameters in a commercial building's open computer lab over ten days, assessing thermal comfort, IAQ, and lighting performance. The results confirm the toolbox's effectiveness in providing accurate, real-time IEQ data. The IEQ toolbox presents a cost-effective, flexible solution for continuous indoor environmental monitoring. Its open-source hardware and software components, combined with wireless data transmission and real-time analysis capabilities, make it a valuable tool for both research and practical applications in building performance assessment. The study highlights the potential for further development and integration of additional sensors, reinforcing the toolbox's versatility and utility in enhancing indoor environmental quality. (11) In today’s world, Data Acquisition Systems (DAQs) serve as essential interfaces between the analog and digital realms, enabling the collection, processing, and analysis of various physical phenomena. Widely utilized in research labs and industrial automation, DAQs measure signals like current and voltage, with data acquired by sensors needing conditioning before digital processing. This complexity requires multiple components—sensors, communication links, signal processors, computers, databases, and software—working in unison. Literature shows diverse DAQ architectures, such as embedded systems, web-based, and Arduino-based DAQs, each tailored for specific applications like environmental studies and automotive monitoring. This work presents a real-time DAQ system using an Arduino UNO development board (AUDB) and Python programming, comprising a sensing unit, signal processing unit (SPU), AUDB, and PC interface. The system employs a fiber loop sensor to measure liquid volume changes, processed by an AD620 instrumentation amplifier, and digitized by the AUDB for real-time monitoring via Python. Experimental validation with various water-glycerol mixtures demonstrated the system's capability, showing increasing output voltage with liquid volume and density. This Arduino-based DAQ system, compared to commercial data loggers, offers cost-efficiency, power efficiency, and reduced measurement errors, with future work focusing on calibration for enhanced accuracy. (12) Open-source microelectronics, particularly microcontroller boards like Arduino Uno, have become invaluable in analytical chemistry for their affordability and integrated development interfaces. These devices are versatile, supporting both in-house and portable instrumentation for user control, data acquisition, and analysis in research labs and undergraduate teaching settings. They allow students to gain practical experience with building and programming instrumentation, exemplified by applications such as photometers, automated burets, and PCR thermocyclers. Additionally, Arduinos enhance classroom demonstrations by coupling sensors like pH meters or thermometers to large LCD displays or real-time data graphs on digital projectors. This report details an in-house developed Arduino-based circuit for electronic data acquisition, utilizing open-source Python software to enable cost-effective and versatile data acquisition compatible with any PC or Mac. The described setup, comprising an Arduino Uno and a 16-bit ADC (ADS1115), costs under $50, significantly less than commercial USB data acquisition devices, and offers comparable performance for many laboratory experiments. The setup involves mounting the ADC on a breadboard connected to the Arduino via jumper cables, with power supplied through a USB connection. The open-source software developed in Python simplifies the user interface for data acquisition, allowing users to select serial ports, input channels, acquisition rates, file lengths, and paths. Data is output as tab-delimited text files, easily analyzed with spreadsheet programs. Although the maximum data acquisition rate is capped at 500 Hz for stability, this suffices for most classroom and laboratory experiments. Demonstrations included chromatographic separations and data acquisition from commercial capillary electrophoresis instruments, showing the Arduino-based system as a viable, low-cost alternative to commercial setups. This system supports a wide range of time-based experiments, making it ideal for teaching environments where high-resolution data acquisition is needed without incurring high costs. The report underscores the advantages of integrating microcontrollers into chemical instrumentation, enhancing educational curricula by enabling hands-on experience with instrument design and data acquisition. The device’s simplicity and affordability make it accessible for broad deployment in teaching and research labs, promoting a deeper understanding of electronics in analytical chemistry. Additionally, the flexibility of open-source software facilitates customization and integration with various instruments, further extending the utility of microcontrollers in experimental settings. The project is supported by NIH grants and contributions from the University of Michigan, highlighting its significance in advancing educational and research methodologies in analytical chemistry. (13) MQTT, a standards-based messaging protocol, is integral for machine-to-machine communication, particularly within the Internet of Things (IoT) ecosystem, where devices like smart sensors and wearables operate over resource-constrained networks with limited bandwidth. It facilitates efficient data transmission due to its simplicity and minimal resource requirements, making it suitable for small microcontrollers and ensuring optimized network bandwidth with minimal control messages. The protocol's scalability, requiring minimal code and power consumption, supports communication with vast numbers of IoT devices, potentially millions. Reliability is enhanced through built-in features that reduce reconnection time and ensure message delivery with three quality-of-service levels—at most once, at least once, and exactly once. Security is maintained through message encryption and modern authentication protocols, such as OAuth and TLS1.3, while extensive support across programming languages like Python allows for quick and minimal coding implementations in various applications. Historically, MQTT was developed in 1999 for the oil and gas industry to monitor pipelines via satellite, designed for minimal bandwidth and battery usage. Initially tied to IBM's MQ Series product, it became an open protocol in 2010 and was later standardized by OASIS in 2013, with version 5 released in 2019. MQTT operates on the publish/subscribe model, which decouples message senders (publishers) from receivers (subscribers) via a message broker. This model provides space decoupling (publishers and subscribers don’t share network location info), time decoupling (they don’t need simultaneous connectivity), and synchronization decoupling (messages can be sent or received independently).

Key MQTT components include clients and brokers. An MQTT client, which can be any networked device running an MQTT library, acts as a publisher when sending messages and a subscriber when receiving them. The MQTT broker, coordinating message distribution, handles client authorization, authentication, message filtering, and missed message management. Communication begins when a client sends a CONNECT message to the broker, which responds with a CONNACK message, establishing a TCP/IP-based connection. MQTT messages, organized by topics, allow the broker to filter and distribute them to subscribed clients. Topics follow a hierarchical structure, like file directories, enabling organized message management (e.g., "ourhome/groundfloor/livingroom/light"). Clients publish messages containing topic and data in byte format, which can be various data types, while subscribers receive messages on topics of interest by sending a SUBSCRIBE message to the broker. For web integration, MQTT over WebSockets (WSS) wraps MQTT payloads in WSS headers, allowing data reception in web browsers. MQTT ensures security through SSL protocol, supporting encryption and authentication using SSL certificates and passwords. Clients are authenticated by the broker via unique identifiers and passwords, and clients authenticate servers using certificates or DNS lookups. While MQTT is not RESTful due to its publish/subscribe model and reliance on a standing TCP connection, MQTT version 5 introduces a request/response feature resembling REST's request-response pattern. AWS IoT Core supports MQTT by enabling secure, scalable IoT device management and message routing to AWS services, offering multiple communication protocols, mutual authentication, end-to-end encryption, and real-time data processing based on business rules. AWS's managed service simplifies IoT fleet management without server provisioning, ensuring secure and reliable connections for billions of devices. (14) MQTT, or Message Queuing Telemetry Transport, is a lightweight messaging protocol designed for restricted low-bandwidth networks and high-latency IoT devices. It operates on the publish/subscribe principle, where data sources publish information to a central broker, and interested recipients subscribe to specific topics to receive the data. This protocol is particularly suitable for machine-to-machine (M2M) communication in environments with limited bandwidth, high latency, or unreliable connections, making it ideal for IoT applications. A fundamental concept in MQTT is the topic, which serves as a hierarchical identifier for messages. Clients subscribe to topics of interest, and the broker ensures that published messages are delivered to the appropriate subscribers. The broker, the central component in the MQTT architecture, manages the routing of messages between publishers and subscribers, handles client sessions, and provides authentication and authorization mechanisms. Examples of MQTT brokers include HiveMQ and EMQX, as well as cloud-based solutions like Azure IoT Hub and AWS IoT Core. Messages exchanged via MQTT consist of a topic and a payload, with the payload containing the actual data to be transmitted. The payload can be structured in various formats such as JSON, XML, or OPC UA, allowing for flexible communication between devices and software. MQTT clients, which encompass all devices and software connected to the broker, can both publish messages to topics and subscribe to topics to receive messages. Quality of Service (QoS) levels determine the reliability of message delivery, with options ranging from ensuring delivery exactly once to at most once. In IoT applications, MQTT facilitates the transmission of data from numerous devices to a centralized cloud environment for analysis and processing. By organizing data into topics and employing a publish/subscribe model, MQTT enables seamless communication between machines and people, enabling various stakeholders to access and act upon IoT data. This approach enhances efficiency, reduces costs, and enables predictive maintenance and planning in IoT deployments. Getting started with MQTT is straightforward, with MQTT brokers like HiveMQ offering easy setup and configuration. The protocol's simplicity and efficiency make it well-suited for IoT and cloud applications, enabling fast, reliable, and scalable messaging between connected devices and servers. By leveraging MQTT, organizations can harness the power of IoT to drive innovation, improve decision-making, and unlock new opportunities for automation and optimization in various industries. (15) The rise of interconnected devices in the Internet of Things (IoT) ecosystem has accentuated the urgency of developing robust security mechanisms to safeguard communication channels against escalating cyber threats. While conventional solutions like Transport Layer Security (TLS) and symmetric encryption provide effective protection, they often introduce performance overhead, particularly on resource-constrained IoT devices and networks. In response, the proposed Value-to-Keyed-Hash Message Authentication Code (Value-to-HMAC) mapping method presents a novel approach, emphasizing message integrity and confidentiality without the computational burden associated with encryption. By leveraging HMAC signatures generated from data using secret keys, the Value-to-HMAC method offers streamlined security, outperforming traditional symmetric encryption algorithms, especially on devices with limited resources like the Onion Omega2+. Its reliance on efficient cryptographic hash functions like SHA3-224, SHA3-256, or Blake2s ensures both speed and security, making it a promising solution for securing IoT communication channels while addressing emerging challenges in the evolving IoT landscape. (16) (17) The rapid proliferation of Internet of Things (IoT) devices has sparked a demand for efficient and lightweight communication protocols, leading to the widespread adoption of MQTT (Message Queuing Telemetry Transport). MQTT, a publish/subscribe Push protocol developed by IBM in 1999, has emerged as a cornerstone in IoT communication due to its lightweight nature and bandwidth efficiency. It facilitates communication between devices in constrained environments with low bandwidth and processing capabilities, making it ideal for IoT applications. MQTT operates on a publish/subscribe model, where publishers disseminate messages to topics, and subscribers receive messages based on their topic subscriptions. Quality of Service (QoS) levels ensure the reliability of message delivery, ranging from at most once to exactly once. Additionally, MQTT supports retained messages, clean sessions, and wills, enhancing its reliability and resilience in IoT deployments. However, MQTT does have limitations, including the absence of message expiry, challenges in maintaining message ordering, and the lack of support for message priority. Security concerns also arise, with authentication mechanisms varying among different broker implementations. Despite these drawbacks, MQTT remains integral to IoT ecosystems across diverse domains, including healthcare, energy, utilities, and social networking. Various brokers, such as Mosquitto, RSMB, MQTT.js, HiveMQ, and VerneMQ, offer different features and limitations, catering to specific IoT use cases. Moving forward, addressing the limitations of MQTT, such as message expiry, security enhancements, and support for message priority, presents opportunities for further innovation. Future work may focus on developing MQTT brokers capable of providing advanced functionalities like priority messaging, message ordering, and enhanced security based on policy-Attribute based encryption using lightweight elliptic curve cryptography, thereby advancing the capabilities of MQTT in meeting the evolving needs of the IoT landscape. (18)

Node-RED data visualization is an emerging tool gaining traction in the era of big data, offering simplified integration of IoT devices and intuitive data processing through visual programming. Leveraging Node-RED's low-code environment, users can easily construct data processes by dragging and dropping nodes, while Node-RED data visualization enhances this capability by presenting data in various charts and display components for clearer understanding. The advantages include visual programming efficiency, diverse chart options, real-time data updates, and robust scalability, enabling access to third-party data sources. Its application spans IoT device monitoring, data analysis reporting, and operational monitoring, empowering enterprises to monitor, analyze, and make informed decisions more effectively. With its user-friendly features and versatile chart displays, Node-RED data visualization is poised to become a valuable asset for businesses and developers in navigating the complexities of data analysis and decision-making. (19) Node-RED is a visual programming tool designed to connect hardware devices, APIs, and online services through a flow-based interface accessible via any browser. Utilizing Node.js as its runtime environment, Node-RED excels in event-driven applications, allowing users to automate tasks without coding. The Node-RED Dashboard, a downloadable plugin, enhances web visualization and dashboard customization. Hosted on the IQ Home gateway, it offers users a clean, customizable platform to display sensor data in various formats like line charts, bar charts, pie charts, or gauges. This remote-accessible solution supports secure development and debugging of Node-RED flows through the LinkIt! application, providing high customizability and low time to market for automation needs. (20) The proliferation of smartphones, smart devices, and affordable sensors has sparked considerable interest in the Internet of Things (IoT). This expansion has resulted in a surge of IoT-related products, spanning from devices to applications. Sensors deployed within an IoT network furnish data regarding their environment, engaging with other devices and applications within the broader IoT ecosystem. However, IoT components often diverge from standardization, rendering environments heterogeneous. This heterogeneity poses challenges in real-time data collection and monitoring, including overseeing the infrastructure. Such data collection serves not only to analyze the environment but also to monitor device health and operational elements like software. This paper proposes an architecture aimed at gathering, visualizing, and monitoring streams of sensor and infrastructure data. The ThingsBoard IoT platform facilitates data collection and visualization, while Node-Red organizes data based on sensor names. Experiments employing sensor datasets demonstrate the approach, processing, and visualization methods. (21) Ensuring high-quality electrical power consumption and distribution is crucial for industries, aligning with the goal of minimizing carbon footprint and preserving costly electrical components. This study presents the implementation of a Hybrid Shunt Active Harmonic Power Filter (HSAHPF) aimed at mitigating harmonic pollution. An Artificial Neural Network (ANN)-based control algorithm is deployed within a Hardware in the Loop (HIL) configuration, trained on the pq0 theory model. Integrating a physical processor with the designed filter, the HIL setup employs an external microprocessor (Raspberry PI 3B+) as the primary data server, generating reference current signals for the HSAHPF based on inputs including 3-phase source voltages, 3-phase applied load currents, and compensated voltage across the DC-link capacitors. The ANN model utilizes backpropagation and gradient descent to predict outputs, ensuring effective harmonic reduction. Real-time data visualization is facilitated through a Java script Application Programming Interface (API) called Node-RED, enabling seamless data transmission between SIMULINK and external processors via serial socket TCP/IP communication for real-time data exchange. Additionally, a real-time Supervisory Control and Data Acquisition (SCADA) system is demonstrated for HSAHPF testing, leveraging a HIL topology enabling control algorithms to execute on an embedded microprocessor for physical system control. (22) The proliferation of IoT (Internet of Things) strategies has ushered in a significant wave of digitization across various industrial sectors, leading to the integration of IoT trends into industrial developments, thereby transitioning from conventional IoT to IIoT (Industrial IoT). This evolution extends beyond mere industrial processes to encompass the well-established Industry 4.0 paradigm, characterized by automation, big data, IoT, and cloud computing. This paper aims to underscore the significance of IoT integration within the industry and elucidate the utilization of the vast amount of collected data. Traditionally bound by stringent norms and regulations, the industrial landscape is now embracing Industry 4.0 by deploying industrial iterations of familiar IoT devices; for instance, Raspberry Pi is being replaced by RevolutionPi. By illustrating through a generic machine example, we will showcase contemporary technological capabilities in data storage, utilization of various IIoT communication protocols such as MQTT, and the benefits of widely adopted IBM-developed Node-RED. The paper endeavors to demonstrate the simplicity of storing, analyzing, and visualizing data from industrial systems leveraging innovative IT advancements. Furthermore, it presents the developed system, the gleaned valuable data, and the requisite insights for future developmental endeavors. (23)

IoT security is paramount in safeguarding internet-connected devices and networks against cyber threats, encompassing practices aimed at identifying, monitoring, and addressing potential vulnerabilities. As the IoT landscape expands to include a myriad of interconnected devices, ranging from smart homes to industrial systems, the sheer volume and diversity of these devices present ample opportunities for cybercriminals to exploit security loopholes. The consequences of IoT security breaches can be severe, affecting both virtual and physical systems, with potential implications for safety, privacy, and data integrity. Key challenges in IoT security include the lack of robust testing and development practices, default passwords susceptible to brute-forcing, the proliferation of IoT malware and ransomware, data privacy concerns, insecure interfaces, and the rise of remote working, exacerbating vulnerabilities in home networks. High-profile examples of IoT security breaches, such as the Mirai botnet attack and the VPNFilter malware incident, underscore the urgency of implementing robust security measures. Best practices for IoT security include keeping devices and software updated, changing default passwords, using strong passwords and encryption methods, setting up guest networks, reviewing privacy settings, disabling unused features, enabling multi-factor authentication, understanding networked devices, and exercising caution when using public Wi-Fi. By adhering to these best practices, users can mitigate IoT security risks and ensure the integrity and security of their connected devices and networks. (24) Securing an IoT deployment involves addressing risks across three key layers: the device, communication, and application layers. At the device layer, organizations need to safeguard both physical and software properties, employing measures like secure booting, user authentication, and regularly updated firmware. In the communication layer, infrastructure and data-centric solutions such as encryption methods and VPNs are crucial to protect data in transit. Finally, at the application layer, securing web, mobile, and cloud components is paramount, with practices like code analysis, automated updates, and key exchange solutions being essential. Organizations must also implement threat management and monitoring systems to detect and respond to anomalies effectively. Solutions like KORE's SecurityProTM offer comprehensive network and security monitoring to help protect IoT solutions. (25) (26) In recent years, the proliferation of Internet-connected devices, spanning from basic sensors to sophisticated cloud servers, has propelled the expansion of the Internet of Things (IoT). This study delves into the profound impact of cloud computing on IoT, recognizing the necessity for a robust system to manage the vast volumes of data generated by IoT devices. Across diverse sectors such as industry, healthcare, automotive, and beyond, IoT technologies have been instrumental in streamlining various aspects of human life. However, with the widespread adoption of IoT comes an escalation in security threats. To ensure secure communication within IoT networks, various security measures, including encryption techniques, are imperative. This research focuses on exploring data protection methodologies, particularly emphasizing the evolution and significance of lightweight encryption techniques tailored for the IoT environment. Furthermore, it underscores the critical need for End-to-End Encryption in IoT systems, advocating for the implementation of security protocols like the Signal protocol, which offers robust end-to-end encryption for applications. The paper concludes by outlining potential avenues for future research, emphasizing the ongoing necessity for innovative security solutions to address evolving threats in the IoT landscape. (27) Security stands as a paramount concern within the Internet of Things (IoT), a sentiment echoed by various stakeholders. Unlike previous instances where security breaches mainly resulted in financial losses or intellectual property theft, IoT introduces a new realm of vulnerability. With IoT devices undertaking critical functions, their compromise could lead to far-reaching consequences, impacting public safety, environmental integrity, productivity, and more. While existing IP Security protocols address several concerns, IoT's unique architecture and device characteristics present distinct challenges to ensuring comprehensive security across every IoT solution. To navigate these challenges, innovative security propositions tailored to IoT's constraints have emerged. However, implementing robust security measures must strike a balance between effectiveness and associated costs, alongside aligning with specific business requirements. Consequently, not every IoT solution necessitates the entire spectrum of IoT security protocols. This paper delves into the intricacies of IoT security challenges, offering insights into potential security options and approaches tailored to address the diverse needs of IoT solutions. (28)

The rapid advancement of the Internet of Things (IoT) is fundamentally reshaping our interaction with technology in our daily lives. This paradigm facilitates seamless communication among low-resource devices, with the data they generate being utilized for crucial decision-making across various domains such as traffic management, healthcare, and home security. Given the constrained resources of IoT devices, accurately assessing their reliability is paramount. This report offers a comprehensive examination of the evolution of reliability measurement, followed by a thorough review of the current state-of-the-art methods for quantifying reliability in IoT systems. The analysis underscores the myriad challenges inherent in this endeavor. Importantly, this review identifies several key research directions for enhancing IoT reliability. Despite the critical importance of this research area, the current study stands as the first detailed review in assessing IoT reliability. It serves as a foundation for further exploration and development in this field, addressing the pressing need for reliable IoT systems in our increasingly interconnected world. (29) (30) A model for Internet of Things (IoT) systems that considers probabilistic functional dependence (PFD), where the failure of one component (the trigger) can lead to other components (dependent components) becoming isolated or inaccessible with certain probabilities. This phenomenon is common in IoT systems involving relayed wireless communications, such as body sensor systems and smart homes. Unlike existing works that assume single-level PFD and zero failure propagation time, this study acknowledges cascading PFD in IoT systems with multi-level configurations. In such systems, a component may serve as both a trigger and a dependent component simultaneously, leading to correlations among different PFD groups. Moreover, failures may not instantly propagate but could take some random time to become effective. To address these complexities, the paper proposes a combinatorial hierarchical methodology for reliability analysis of IoT systems subject to cascading PFD and random failure propagation time. This methodology is versatile, accommodating various types of failure and propagation time distributions. To illustrate its applicability, the study includes an analysis of a smart home sensor system using the proposed methodology. (31) The Internet of Things (IoT) holds the promise of revolutionizing human society, making it more intelligent, convenient, and efficient, with significant economic and environmental benefits. However, ensuring reliability is a critical challenge that must be overcome to realize this transformative potential. By examining the layered architecture of IoT and identifies reliability challenges associated with the specific enabling technologies at each layer. It then conducts a comprehensive synthesis and review of existing literature on IoT reliability. The review categorizes and analyzes reliability models and solutions across four layers: perception, communication, support, and application.Despite the considerable volume of research in this area, the field of IoT reliability is still in its nascent stages. The article highlights several challenging research problems and opportunities, particularly focusing on the current underexplored behaviors and future aspects of the evolving complexity and dynamics of IoT systems. This discussion aims to guide future research efforts toward addressing critical reliability concerns and harnessing the full potential of IoT technology. (32)

The integration of IoT technology into business processes is on the rise, with a focus on reducing power consumption to extend device lifespan and improve environmental impact. Selecting the right radio technology, such as BLE, Wi-Fi, or cellular, is crucial for optimizing power usage based on application needs. Strategies such as minimizing data transmission, utilizing power-saving modes, and leveraging cloud services can greatly enhance the efficiency and longevity of IoT devices, ultimately leading to more cost-effective and sustainable products. (33) The content delves into the intricate details of power management in IoT devices, focusing on minimizing power loss and optimizing energy efficiency. Various topics such as power dissipation in electrical devices, power consumption in digital circuits, and the importance of selecting suitable processors are covered. The discussion extends to power-saving features, clock speed management, and the calculation of battery capacity for IoT devices running on solar power. The content emphasizes the critical role of proper power management in reducing energy consumption and enhancing the overall performance of electronic devices. (34)The content highlights the challenges of energy consumption in IoT devices and offers practical solutions to improve energy efficiency. It emphasizes the importance of selecting the right microcontroller and wireless protocol, such as Bluetooth Low Energy or ZigBee, to reduce power consumption and extend battery life in IoT devices. By partnering with Nabto, businesses can optimize their IoT ecosystem and implement energy-efficient solutions to address these challenges. Additionally, the article suggests exploring additional resources and reaching out for a consultation with Nabto to further enhance device power management in IoT. (35)In the era of the Internet of Things (IoT), low-energy technologies still struggle to provide the reliability required by industries, particularly concerning wireless operations necessary for widespread deployments. Although industrial wireless communication performance has reached an acceptable level, efficiently dimensioning the energy requirements of devices to meet application demands remains challenging. This challenge is exacerbated by the inherent uncertainty associated with energy harvesting.Therefore, it is crucial to model and dimension the energy consumption of IoT applications during the pre-deployment or pre-production stages, taking into account critical factors such as cost reduction, device lifetime, and available energy. This paper introduces a comprehensive model for the power consumption of wireless sensor nodes. The model adopts a system-level perspective, considering all energy expenditures including communications, data acquisition, and processing. Moreover, it relies solely on parameters that can be empirically quantified once the platform (i.e., technology) and the application (i.e., operating conditions) are defined.This framework provides a new approach for studying and analyzing energy life cycles in applications, enabling the determination of the specific influence of application parameters in advance. It also facilitates understanding tolerance margins and trade-offs within the system, aiding in the optimization of energy consumption for IoT deployments. (36)The study in this paper focuses on the importance of utilizing low power wireless technologies in IoT applications to extend the lifespan of network sensors. By comparing various communication techniques such as ZigBee, Low Power Wi-Fi, 6LoWPAN, and LPWA, the study categorizes protocols based on connectivity range and evaluates their power consumption. The findings underscore the critical role of module selection in determining battery life, emphasizing the need to carefully assess protocols based on the specific module employed. (37)In the forthcoming decade, it's anticipated that there will be an excess of 50 billion smart objects interconnected within the Internet of Things (IoT). These smart objects serve as the bridge between the physical world and computing infrastructure, poised to permeate various facets of our daily lives and transform numerous application domains such as healthcare, energy conservation, and transportation. This paper offers an overview of the challenges associated with designing energy-efficient IoT edge devices and highlights recent research endeavors proposing promising solutions to tackle these challenges. Initially, the paper delineates the hurdles in efficiently providing power to an IoT device. Subsequently, it delves into the significance of emerging memory technologies in enhancing the energy efficiency of IoT devices. Finally, the paper explores the potential impact of approximate computing in augmenting the energy efficiency of wearables and other computationally intensive IoT devices. (38)

# **Methodology**

## **Hardware and Software**

**Microcontroller( Arduino Nano 33 IoT)**

In order to connect hardware to the PC, it requires both a programming language and some software, so the Arduino nano comes into play first. The Arduino Nano 33 IoT is Arduino's smallest board designed specifically for Internet of Things (IoT) applications. Powered by the Arm® Cortex®-M0 32-bit SAMD21 processor, it boasts a range of advanced features including the u-blox NINA-W102 Wi-Fi module and the ECC608A crypto-chip for enhanced security. (39) Here are a few of the Arduino Nano’s salient characteristics:

**Wi-Fi Connectivity:** Equipped with the u-blox NINA-W102 Wi-Fi module, the Nano 33 IoT provides seamless connectivity to Wi-Fi networks, enabling IoT projects to connect to the internet and interact with online services. This feature facilitates remote monitoring, control, and data exchange, enhancing the versatility of IoT applications (Arduino, n.d.).

**Arduino Cloud Compatibility:** The Nano 33 IoT is fully compatible with the Arduino Cloud platform, allowing users to build IoT projects effortlessly in just minutes. With Arduino Cloud, developers can remotely manage and monitor their devices, access data in real-time, and create custom IoT applications with ease (Arduino, n.d.).

**Bluetooth® Connectivity:** Additionally, the Nano 33 IoT is Bluetooth® enabled, providing support for Bluetooth® Low Energy (BLE) applications. This feature enables users to control peripheral devices via Bluetooth® and implement a wide range of IoT applications that leverage BLE technology (Arduino, n.d.).

**Integrated IMU:** The Nano 33 IoT comes equipped with an integrated Inertial Measurement Unit (IMU) featuring the LSM6DS3 accelerometer and gyroscope. This IMU enables developers to create motion tracking devices and incorporate motion sensing capabilities into their projects, opening up opportunities for innovative applications in areas such as wearable technology and motion-controlled devices (Arduino, n.d.).

**Programming:** The Arduino Nano 33 IoT can be programmed using the Arduino IDE, a free, open-source program available for Windows, Mac, and Linux. This familiar development environment simplifies the process of writing, uploading, and debugging code for IoT projects (Arduino, n.d.).

**Size:** Similar to the Arduino Nano, the Nano 33 IoT boasts a compact form factor, measuring just 0.7 by 1.7 inches. This small size makes it easy to use on a breadboard or in tight spaces, facilitating rapid prototyping and experimentation.

With its shared features and additional capabilities, the Arduino Nano 33 IoT builds upon the legacy of the Arduino Nano, offering enhanced connectivity and functionality for modern IoT projects.

A computer chip with many different colors

Description automatically generated with medium confidence

Figure 1 : A brief description for the the Arduino pins

Using Arduino Nano 33 IOT instead of Arduino/Arduino UNO because of the following Reason:

1: One of the most significant advantages of the Nano 33 IoT is its built-in Wi-Fi and Bluetooth connectivity. Unlike the Arduino Uno, which requires additional shields or modules for wireless communication, the Nano 33 IoT simplifies IoT development by providing native support for Wi-Fi and Bluetooth connectivity

2: The Nano 33 IoT features a smaller form factor compared to the Arduino Uno, making it more suitable for projects with space constraints or those requiring a compact design. This smaller size allows for easier integration into portable devices, wearables, and other space-limited application

3: The Nano 33 IoT is equipped with a more powerful microcontroller compared to the Arduino Uno. Its ARM Cortex-M0 32-bit SAMD21 processor offers increased processing power and efficiency, allowing for faster execution of tasks and more complex computations. This makes the Nano 33 IoT better suited for applications requiring higher computational performance

4: Nano 33 IoT has two additional analog inputs compared to the Uno, providing slightly more flexibility for sensor interfacing.

However, There are no soldered pins on the Nano, so it must have its pins soldered in order to be used on a breadboard. When soldering, extreme caution must be taken to ensure that the circuit is not shorted. If there are two pins at the same point, then it will be short.

## **Sensor(BME 280)**

It has now received the sensor component that it will use. Utilizing the most basic temperature,Humidity and pressure sensor BME280, test the circuit first.

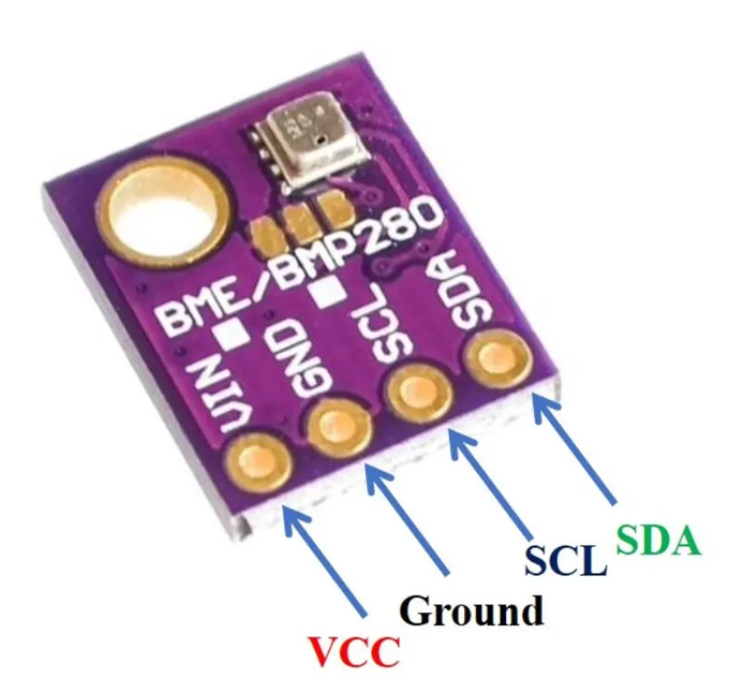


Figure 2 : It is Picture of the Sensor

# Features and Specifications of BME280

## Features

Compatible with 3.3V/5V microcontrollers

**Environmental monitoring:** temperature, humidity, and barometer

Gravity I2C interface and reserve XH2.54 SPI interface

Small size, convenient to install

## Specifications

**Supply voltage:** 3.3 V

**Size:** 15 x 10 mm

**Weight:** 1 g

**Diameter of mounting holes:** 3 mm

**Operation range (full accuracy):**Pressure: 300…1100 hPa

**Temperature:** -40…85°C

**Interface:** I2C and SPI

Average current consumption (1Hz data refresh rate):1.8 μA @ 1 Hz (H, T)

2.8 μA @ 1 Hz (P, T)

3.6 μA @ 1 Hz (H, P, T)

**Average current consumption in sleep mode:** 0.1 μA

**Humidity sensor:** Response time (τ63%): 1 s

**Accuracy tolerance**: ± 3 % relative humidity

**Hysteresis:** ≤ 2 % relative humidity

**Pressure sensor: RMS Noise**: 0.2 Pa (equiv. to 1.7 cm)

**Sensitivity Error**: ± 0.25 % (Equiv. to 1 m at 400 m height change)

**Temperature coefficient offset:** ±1.5 Pa/K (Equiv. to ±12.6 cm at 1°C temperature change)

The BME280 sensor is a versatile environmental sensor capable of measuring temperature, humidity, and barometric pressure. Here are some of its key features:

**Temperature Measurement:** The BME280 sensor accurately measures temperature with high precision. It provides temperature data in degrees Celsius (°C) with a resolution of up to 0.01°C.

**Humidity Sensing:** In addition to temperature, the BME280 sensor also measures relative humidity. It provides humidity data as a percentage (%), allowing for precise monitoring of ambient humidity levels.

**Barometric Pressure Sensing:** The BME280 sensor is equipped to measure barometric pressure, enabling users to monitor changes in atmospheric pressure over time. Barometric pressure data is typically provided in hectopascals (hPa) or millibars (mbar).

**High Accuracy:** With its advanced sensing elements and calibration algorithms, the BME280 sensor offers high accuracy in temperature, humidity, and pressure measurements. This ensures reliable data for various applications, including weather monitoring and indoor climate control.

**Compact Size:** Despite its advanced capabilities, the BME280 sensor maintains a compact form factor, making it suitable for integration into small electronic devices and IoT projects. Its small size allows for easy placement and installation in a variety of environments.

**Low Power Consumption:** The BME280 sensor is designed with energy efficiency in mind, consuming minimal power during operation. This makes it well-suited for battery-powered applications where power consumption is a critical consideration.

**I2C and SPI Interfaces:** The BME280 sensor supports both I2C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface) communication protocols, providing flexibility in interfacing with microcontrollers and other devices. This allows for seamless integration into a wide range of projects and platforms.

**Wide Operating Range:** The BME280 sensor operates over a wide temperature and humidity range, making it suitable for both indoor and outdoor applications. It can withstand environmental conditions ranging from -40°C to 85°C and 0% to 100% relative humidity.

Overall, the BME280 sensor offers a comprehensive solution for environmental monitoring, combining temperature, humidity, and barometric pressure sensing capabilities in a compact and energy-efficient package. Its high accuracy, low power consumption, and versatile interface options make it an ideal choice for a variety of IoT, weather monitoring, and indoor climate control applications.

# **Measuring Data/MQTT protocol implementation**

To facilitate the measurement of atmospheric data,Some Libraries are installed as described below:

**Wire.h:** This library enables communication over the I2C bus, facilitating communication with devices such as sensors or displays.

**Adafruit\_Sensor.h:** Adafruit\_Sensor.h provides an abstract base class for sensor drivers, ensuring consistent sensor data handling across different sensor types.

**Adafruit\_BME280.h:** Adafruit\_BME280.h is a library specifically designed for the BME280 sensor, offering functions to easily interface with and read data from the sensor.

**WiFiNINA.h:** WiFiNINA.h is a library for the Arduino WiFi module, allowing the Arduino to connect to Wi-Fi networks and communicate over the internet.

**PubSubClient.h:** PubSubClient.h is a library for MQTT (Message Queuing Telemetry Transport) communication, enabling the Arduino to publish and subscribe to messages on MQTT topics.

**ArduinoJson.h:** ArduinoJson.h is a library for parsing and generating JSON (JavaScript Object Notation) data, commonly used for exchanging data between devices and services in IoT applications.

After writing the complete code (refer to the Appendix), the Arduino is connected to the PC via a USB communication/power cable. Upon uploading and burning the code, the corresponding atmospheric data is displayed in the serial monitor. Subsequently, the Arduino will establish a connection to the local network via Wi-Fi. It will then transmit this data through the MQTT protocol. A specific topic named "sensor data" has been created, and all three values (temperature, humidity, and pressure) will be published to this topic on the server. This process ensures that the collected atmospheric data is efficiently transmitted and made available for further analysis or utilization. Configuration of Node-RED on Raspberry Pi for data reception and visualization: To enable data reception via the MQTT protocol, a Raspberry Pi was utilized. Prior to receiving data and making it accessible, the Raspberry Pi needed to be set up. This involved the installation of necessary components. To begin, the Raspberry Pi required the following essentials:

* A power supply
* Boot media, typically a microSD card with adequate storage and speed

The Raspberry Pi can be configured either as an interactive computer with a desktop or as a headless computer accessible solely over the network. For a headless setup, additional peripherals such as a display, keyboard, and mouse are not required. Instead, preconfiguring a hostname, user account, network connection, and SSH during the operating system installation suffices. However, for direct usage of the Raspberry Pi, the following accessories are essential:

* A display
* A cable to connect the Raspberry Pi to the display
* A keyboard
* A mouse
* A power supply

Regarding the power supply, different Raspberry Pi models have varying power requirements. For instance, the Raspberry Pi 4 Model B necessitates a 5V/3A power supply, while older models such as the Raspberry Pi 3 require a 5V/2.5A supply. It's crucial to use the appropriate power supply to ensure proper functioning of the Raspberry Pi. Boot media is essential for Raspberry Pi models, as they lack onboard storage. Typically, microSD cards are used for this purpose. The recommended SD card capacity varies depending on the intended usage, with at least 32GB recommended for Raspberry Pi OS installations. Additionally, the use of a wired keyboard and mouse, connected via USB ports, facilitates interaction with the Raspberry Pi. For display connectivity, the Raspberry Pi 4 Model B features two micro HDMI ports. This allows for easy connection to displays for visualization purposes. By adhering to these setup requirements and recommendations, the Raspberry Pi can effectively serve as a platform for receiving and processing data via the MQTT protocol. (40)

### To install Node-RED for programming and receiving data, you can follow these steps:

1. Open the terminal window on your Raspberry Pi.
2. Use the following command to install Node-RED from the Raspberry Pi OS repositories: sudo apt-get install nodered
3. Press Enter to execute the command.
4. Follow any prompts or confirmations that appear during the installation process.
5. Once the installation is complete, Node-RED will be installed on your Raspberry Pi and ready to use.

Installing Node-RED using the package manager (apt-get) ensures a smooth installation process and allows you to easily manage Node-RED updates in the future. (41).To obtain the IP address or link for accessing the Node-RED page online, the command node-red-start can be executed in the terminal. Upon execution, the corresponding IP address, along with the port number (typically 1880), will be displayed. This information can then be used to access the Node-RED editor interface via a web browser. (41)

# To install the Mosquitto MQTT broker on the Raspberry Pi, follow these steps:

## Step 1: Activating Mosquitto Repository

* Open the LXTerminal and execute the following commands:
* curl -O http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
* sudo apt-key add mosquitto-repo.gpg.key
* rm mosquitto-repo.gpg.key
* cd /etc/apt/sources.list.d/
* sudo curl -O http://repo.mosquitto.org/debian/mosquitto-wheezy.list
* sudo apt-get update

## Step 2: Installing Packages

* Install the Mosquitto packages using the command:
* sudo apt-get install mosquitto mosquitto-clients python-mosquitto
* After installation, stop the Mosquitto broker using
* sudo /etc/init.d/mosquitto stop

## Step 3: Configuration

* Create a configuration file and add the following lines to restrict anonymous clients:
* cd /etc/mosquitto/conf.d/
* sudo nano mosquitto.conf
* allow\_anonymous false
* password\_file /etc/mosquitto/conf.d/passwd
* require\_certificate false
* Save the file and exit the editor. Then create an empty password file
* sudo touch passwd
* Use the mosquitto\_passwd tool to create a password hash for the user 'pi':
* sudo mosquitto\_passwd -c /etc/mosquitto/conf.d/passwd pi

You will be prompted to enter the password twice. Enter the password according to your preference. (42) With all the necessary libraries installed and everything prepared, it's time to start programming in Node-RED. Below, we outline the required block codes:

**MQTT In Block:** This block serves as the MQTT input node in Node-RED, facilitating data reception from an MQTT broker.

* **Function 1:** This function node is utilized to extract only the temperature data from the incoming data stream.

**Gauge 1:** Representing a gauge node, this block is responsible for displaying data on the Node-RED dashboard. Users can adjust the minimum and maximum values according to their requirements, such as setting a temperature range of 0 to 60 degrees Celsius. Additionally, customization options for the name, unit, and appearance of the gauge are accessible by double-clicking on the block.

* **Function 2:** This function node extracts the pressure value from the incoming data stream.

**Gauge 2:** Similar to the previous gauge, this block displays the pressure value on the Node-RED dashboard, typically in hectopascals (hPa). Users have the flexibility to customize the appearance and range of the gauge as needed.

* **Function 3:** Responsible for extracting the relative humidity value, this function node typically operates within a range of 0 to 100 percent.

Gauge 3: This gauge node visualizes the relative humidity value on the Node-RED dashboard, often within a range of 0 to 100 percent.

* **Function 4:** Designed to create a .csv file on the desktop and write real-time data into it, this function node enables data storage on external devices like USB drives or SD cards without requiring a display screen. Users can tailor the file path to suit their preferences.

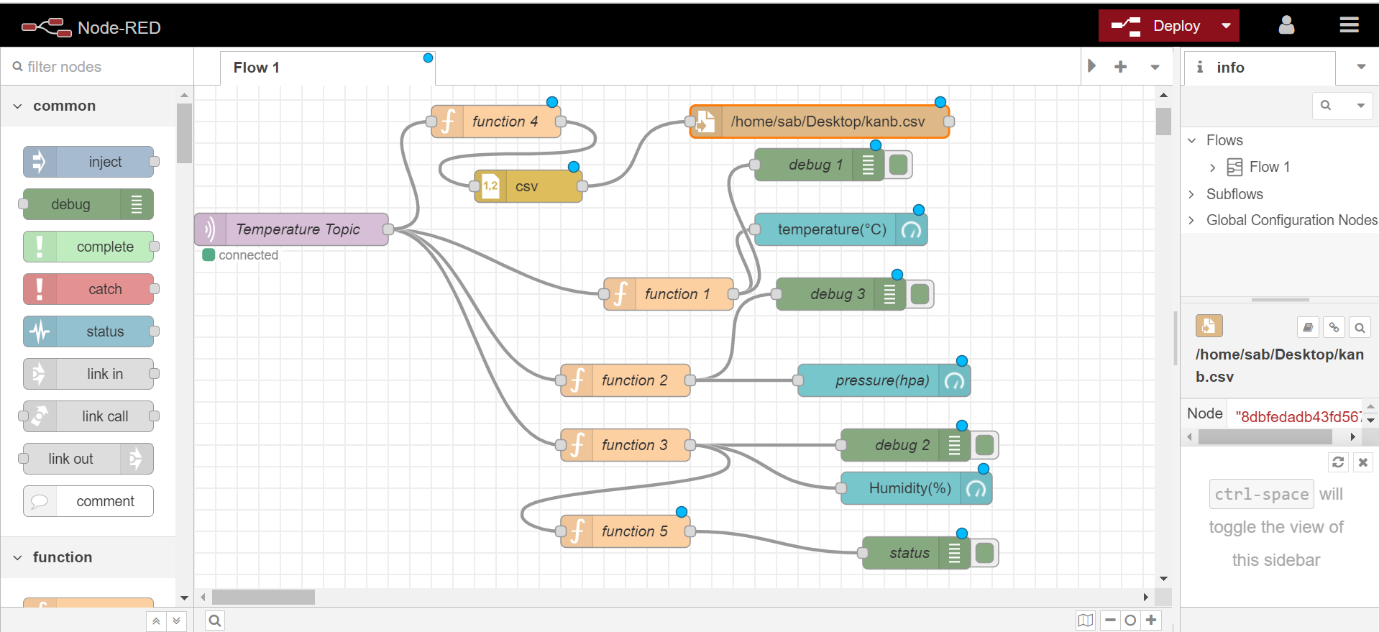


Figure 3: This is how it is connected

# **Results**

Once all components are set up and the program is executed, including the debugging and uploading processes, the MQTT communication is initiated. Subsequently, the temperature, pressure, and humidity values are displayed in the serial monitor, as depicted in the illustration below.

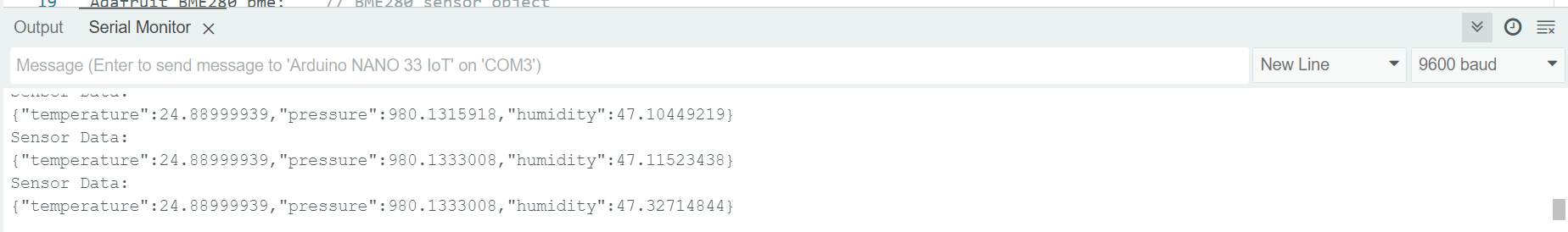


Figure 4: Result

Similarly, upon deploying our Node-RED program, we successfully received the data on the Raspberry Pi side. This signifies that the communication setup established between the sensor nodes and the Raspberry Pi was effective. As a result, the Raspberry Pi was able to receive and process the incoming data, allowing for further analysis or visualization through the Node-RED platform. This successful data reception on the receiving end validates the robustness of our deployed solution and ensures that our system is functioning as intended.

**Data visulization:** The received data can be observed in the Node-RED debug window. However, for more comprehensive analysis and visualization, we utilize the Node-RED dashboard. This dashboard can be accessed using the same IP address followed by "/ui" in our case, such as 192.168.1.102:1880/ui. Here, "1880" represents the port through which the Node-RED dashboard is accessible. By accessing this dashboard, we can effectively visualize and analyze the received data in a user-friendly interface, facilitating better insights and decision-making.

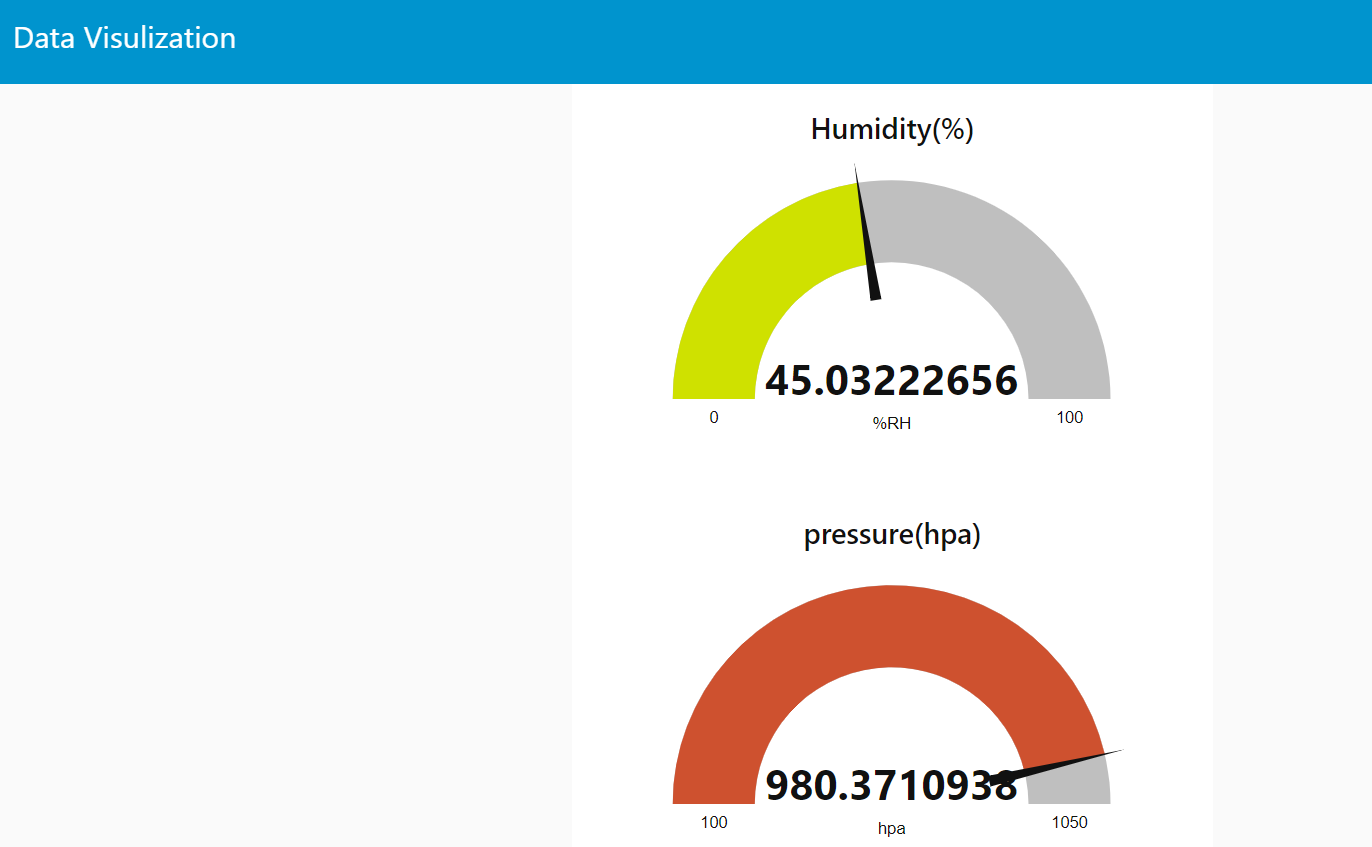
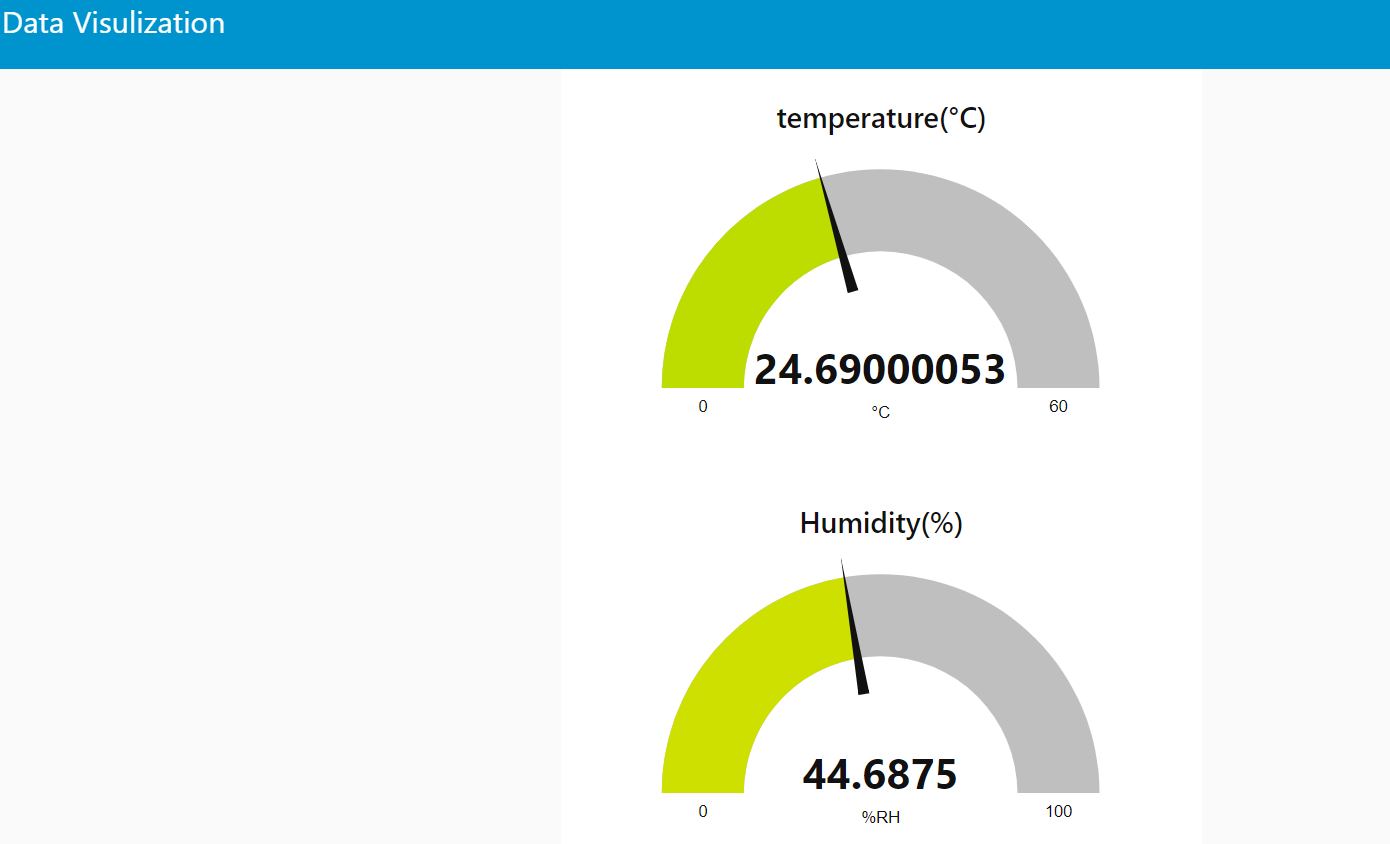
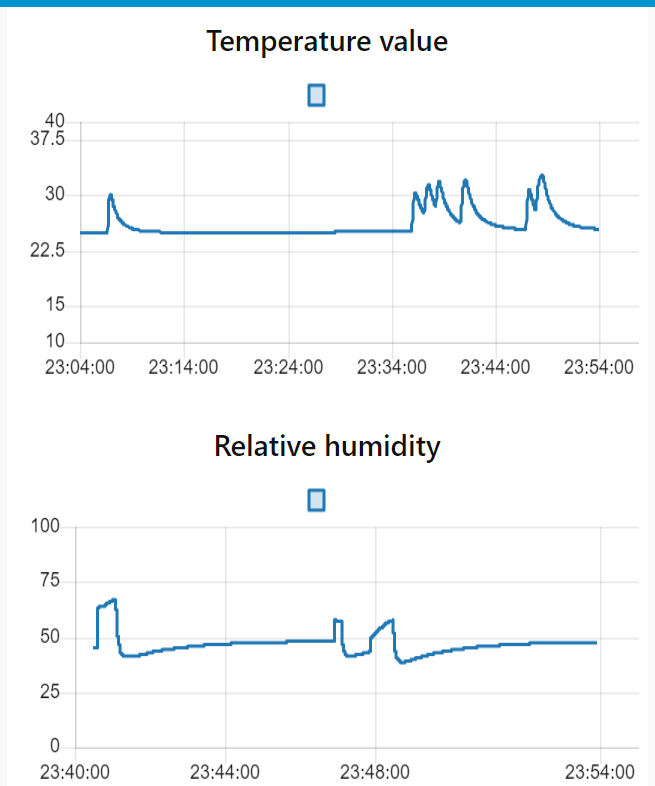


Figure 5: Show the pressure temperature humidity value

The figure above depicts real-time temperature, pressure, and humidity values. The temperature is indicated as 24.5 degrees Celsius, while the pressure is measured in hPa (hectopascals), and the relative humidity is represented as a percentage ranging from 0 to 100. In this context, 0 signifies completely dry air. To enhance visualization, we have configured the temperature range from 0 to 60 degrees Celsius, the pressure range from 100 to 1050 hPa, and the humidity range from 0 to 100. Adjustments to these ranges can be made by double-clicking on the corresponding gauge block within the Node-RED interface, allowing flexibility to tailor the visualization according to specific requirements and sensor characteristics.

**Comparison of different graph:** By connecting the chart block in Node-RED, we can visualize our data in various formats such as line graphs, pie charts, and more. Unlike displaying only the current value, these graphs provide a comprehensive view of the data over a specified time interval. This feature enables us to analyze trends and patterns over time, allowing for deeper insights into the data. Whether examining data on a weekly, monthly, or custom timeframe, the visualization capabilities provided by Node-RED empower users to gain valuable insights and make informed decisions based on historical data tree.



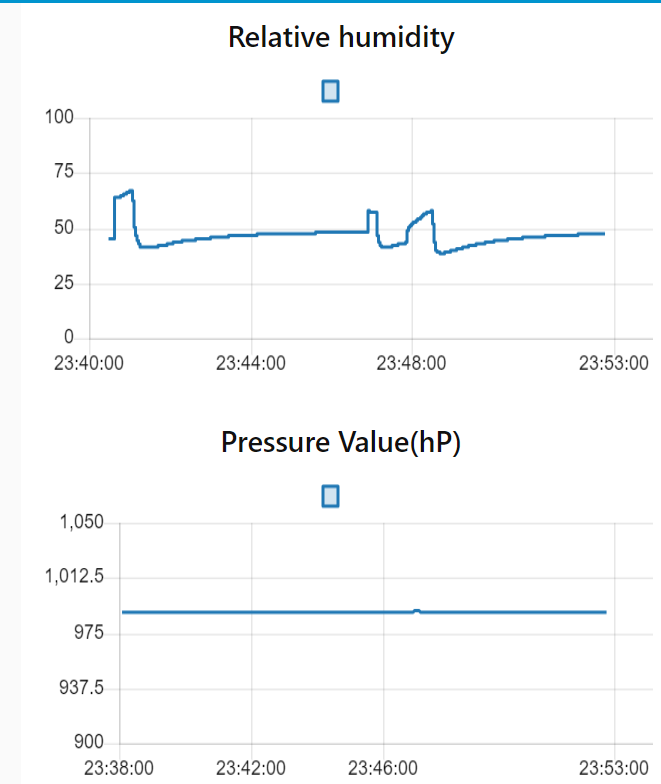


Figure 6 : Result Graph

The line graph in the above figure depicts the variation of temperature, pressure, and humidity values over a specified period, offering a dynamic representation of these parameters. Notably, the humidity and pressure values fluctuate, as indicated by the varying peaks and troughs in the graph. This fluctuation occurs due to experimental manipulation, such as touching the sensor to the body, altering the immediate environment's humidity and pressure conditions. In contrast, the pressure remains relatively constant, reflecting the stable conditions within the room environment, where positional changes do not significantly impact atmospheric pressure. By analyzing trends and previous values, we can forecast future data points using various algorithms such as linear regression, moving averages, exponential smoothing, and more. These algorithms utilize historical data patterns to predict future trends and provide valuable insights for decision-making. Whether it's anticipating future demand, predicting equipment failures, or forecasting weather patterns, leveraging these algorithms allows us to make proactive decisions and mitigate risks. Additionally, by integrating machine learning techniques, we can enhance the accuracy of our forecasts and adapt to changing conditions more effectively.

**Data Storage:** In order to facilitate efficient data analysis for subsequent research endeavors, the collected data is systematically stored in a CSV (Comma Separated Values) file format. The implementation involves four function blocks dedicated to data management, each designed to write specific data streams into the CSV file. These streams include pressure, temperature, and sensor readings, with each datum accompanied by a timestamp for temporal context. The CSV file is created on the desktop and named 'kanb' for ease of access and organization.

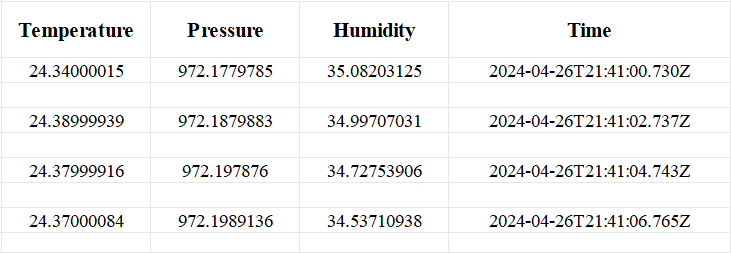


Figure 7 : Recorded Data

This structured approach ensures the integrity and accessibility of the collected data for comprehensive analysis and interpretation.To enhance flexibility and accessibility, the data storage location can be modified to an external USB drive connected to the Raspberry Pi. By adapting the file path accordingly, the data remains securely stored and can be conveniently analysed at any given time. This configuration ensures portability and ease of access, enabling seamless data analysis processes while leveraging the Raspberry Pi's capabilities for efficient data management.

**Power Consumption of Arduino Nano 33 IOT:** One of the objective of this section is to calculate the power consumption of the Arduino Nano 33 module. This is a crucial consideration in assessing the device's operational efficiency. The Arduino Nano 33 typically consumes 3.279 V and 1.35 mA of power as shown in the figure.

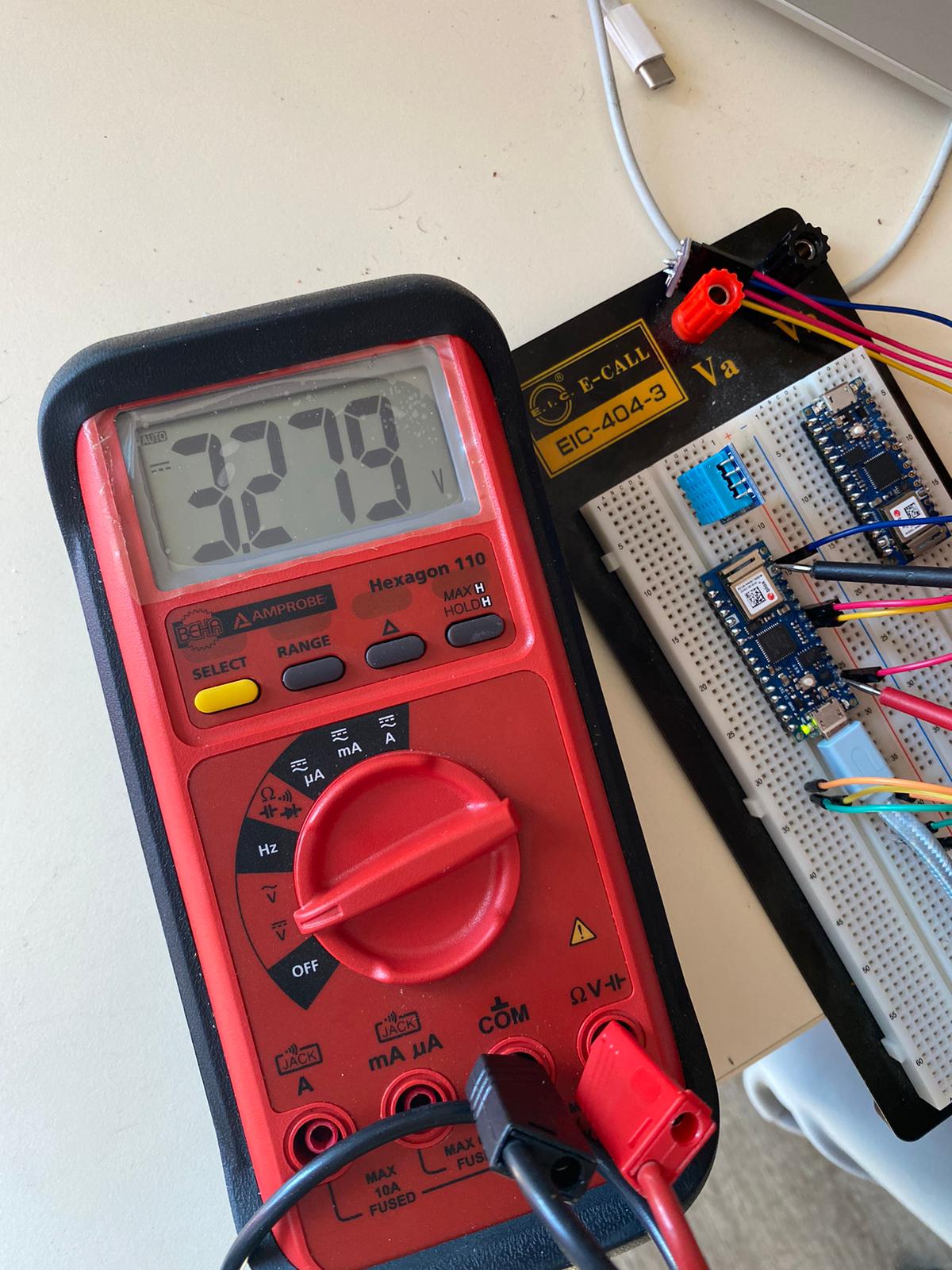


Figure 8: Recorded Data

# **Using the formula**

P = V \* I (where P represents power, V represents voltage, and I represents current), we calculate:

P=3.279 V×1.35 mA

P=4.4265 mW

P=3.279V×1.35mA

P=4.4265mW

This indicates that the Arduino Nano 33 consumes approximately 4.4265 milliwatts of power during operation.

To evaluate the annual power consumption, we apply the formula:

E=P×365×24

Substituting the calculated power consumption value:

E=4.4265 mW×365×24

E=38.7666 mWh

E=4.4265mW×365×24

E=38.7666mWh

Thus, the Arduino Nano 33 consumes approximately 38.7666 milliwatt-hours of energy over the course of a year. For scenarios where the Arduino Nano 33 operates outdoors on battery power, considerations must be made regarding battery selection and efficiency. Utilizing a lithium-ion battery with a capacity of 2000mAh and an operating voltage of 4.5V as a reference, the expected runtime can be determined using the formula:

T=2000 mAh/1.35 mA

T​=1481.48hours

This indicates that the Arduino Nano 33 can operate continuously for approximately 1481.48 hours, or around 61.7 days, on a single charge. However, practical considerations such as battery discharge curves, intermittent power spikes during data transmission, and battery aging effects must be taken into account to optimize operational efficiency.The power rating obtained for the BME280 sensor, as measured in our experiment, aligns with the thermal characteristics reported in the referenced paper. Specifically, our measured power consumption falls within the range of values provided for the BME280 sensor's sleep mode (0.17–0.36 μW) and active mode (595–1260 μW). This consistency reinforces the validity of our experimental setup and measurements, indicating that our findings accurately reflect the expected performance of the BME280 sensor under different operating conditions (43)Top of Form.

**Security Consideration:** The data being transmitted via the MQTT protocol is securely configured, utilizing the local WiFi network for communication. This configuration requires the use of both a SSID (Service Set Identifier) and a password, ensuring that only authorized devices can access and exchange data within the network. To receive this data, one must first connect to the same local WiFi network by providing the correct SSID and password. Once connected, access to the data becomes available through the network's assigned IP address. This IP address can be used not only for data reception but also for accessing other network services, such as browsing the programming interface of tools like Node-RED and visualizing the transmitted data through dedicated visualization platforms. The secure configuration of SSID and password ensures the integrity and privacy of the transmitted data, while access to the local WiFi network enables seamless interaction with the data and related services, enhancing the overall functionality and utility of the system. Accessing the data doesn't necessitate an external display monitor; instead, it can be conveniently done using a smartphone or any device with network capabilities. Once connected to the local network by entering the correct SSID and password, users can simply input the corresponding IP address into their smartphone's browser or a dedicated application. This streamlined process enables seamless access to the data without the need for additional hardware. It enhances flexibility by allowing users to monitor and interact with the data remotely, using their preferred devices. Whether at home, in the office, or on the go, users can effortlessly tap into the network and access the valuable data insights provided by the system.

# Future Work and Improvements:

**Cloud Integration:** Consider integrating the data into cloud platforms like AWS to enable remote accessibility from any location on Earth. This would enhance the scalability and flexibility of the system.

**Security Enhancements:** Implement a username and password system within the Node-RED platform to add an extra layer of authentication. Additionally, explore options for role-based access control to manage different levels of access for users.

**Enhanced Encryption for Energy Systems:** Investigate the implementation of advanced encryption techniques tailored specifically for energy-related data. This could involve exploring options such as end-to-end encryption, homomorphic encryption, or quantum encryption to ensure the security and privacy of sensitive data.

**Energy Efficiency Considerations:** While implementing these improvements, it is essential to consider energy efficiency. Optimize data transmission protocols and minimize computational overhead to ensure minimal impact on energy consumption.

These future enhancements aim to further enhance the reliability, accessibility, and security of the system, paving the way for more robust and efficient energy management solutions.

**Appendix:**

#include <Wire.h> // Include the Wire library for I2C communication

#include <Adafruit\_Sensor.h> // Include the Adafruit sensor library

#include <Adafruit\_BME280.h> // Include the Adafruit BME280 sensor library

#include <WiFiNINA.h> // Include the WiFiNINA library for Wi-Fi communication

#include <PubSubClient.h> // Include the PubSubClient library for MQTT communication

#include <ArduinoJson.h> // Include the ArduinoJson library for JSON data handling

#define SEALEVELPRESSURE\_HPA (1013.25) // Define the local sea level pressure for altitude calculations

#define SECRET\_SSID "TP-LINK\_AC2C06" // Wi-Fi network SSID

#define SECRET\_PASS "38951107" // Wi-Fi network password

const char\* mqtt\_server = "192.168.1.102"; // IP address of the MQTT broker

const char\* mqtt\_topic = "sensor\_data"; // MQTT topic to publish sensor data

WiFiClient wifiClient; // Instance of WiFiClient used for Wi-Fi communication

PubSubClient client(wifiClient); // Instance of PubSubClient for MQTT communication

Adafruit\_BME280 bme; // BME280 sensor object

void setup() {

Serial.begin(9600); // Initialize serial communication at 9600 baud

connectWiFi(); // Connect to Wi-Fi network

client.setServer(mqtt\_server, 1883);// Set MQTT server and port

if (!bme.begin(0x76)) { // Initialize BME280 sensor with I2C address 0x76

Serial.println("Could not find a valid BME280 sensor, check wiring!");

while (1); // Halt program if sensor is not found

}

Serial.println("BME280 sensor found.");

// Set BME280 sensor sampling settings

bme.setSampling(Adafruit\_BME280::MODE\_NORMAL, // Operating mode

Adafruit\_BME280::SAMPLING\_X2, // Temperature oversampling

Adafruit\_BME280::SAMPLING\_X16, // Pressure oversampling

Adafruit\_BME280::SAMPLING\_X16, // Humidity oversampling

Adafruit\_BME280::FILTER\_X16, // Filtering

Adafruit\_BME280::STANDBY\_MS\_500); // Standby time

}

void loop() {

if (!client.connected()) { // Check if MQTT client is connected

reconnect(); // Reconnect if not connected

}

client.loop(); // Maintain MQTT connection

// Read sensor data from BME280 sensor

float temperature = bme.readTemperature(); // Read temperature in Celsius

float pressure = bme.readPressure() / 100.0F; // Read pressure in hPa (Pa to hPa conversion)

float humidity = bme.readHumidity(); // Read humidity in percentage

// Create JSON object to store sensor data

StaticJsonDocument<200> jsonDoc; // Define JSON document with buffer size 200

// Populate JSON object with sensor data

jsonDoc["temperature"] = temperature; // Add temperature data to JSON object

jsonDoc["pressure"] = pressure; // Add pressure data to JSON object

jsonDoc["humidity"] = humidity; // Add humidity data to JSON object

// Serialize JSON object to a string

String jsonString; // Define a string to store JSON data

serializeJson(jsonDoc, jsonString); // Serialize JSON object to string

// Print JSON string to Serial Monitor

Serial.println("Sensor Data:"); // Print header for sensor data

Serial.println(jsonString); // Print JSON string

// Publish JSON string via MQTT

client.publish(mqtt\_topic, jsonString.c\_str()); // Publish JSON string to MQTT topic

delay(2000); // Delay between sensor readings

}

void connectWiFi() {

Serial.println("Connecting to WiFi"); // Print status message for Wi-Fi connection attempt

WiFi.begin(SECRET\_SSID, SECRET\_PASS); // Connect to Wi-Fi network with SSID and password

while (WiFi.status() != WL\_CONNECTED) { // Wait until Wi-Fi connection is established

delay(500);

Serial.print("."); // Print dot to indicate connection attempt

}

Serial.println(""); // Print new line

Serial.println("WiFi connected"); // Print status message for successful Wi-Fi connection

Serial.println("IP address: "); // Print header for IP address

Serial.println(WiFi.localIP()); // Print local IP address

}

void reconnect() {

while (!client.connected()) { // Loop until MQTT connection is established

Serial.print("Attempting MQTT connection..."); // Print status message for MQTT connection attempt

if (client.connect("arduinoClient")) { // Attempt to connect to MQTT broker with client ID "arduinoClient"

Serial.println("connected"); // Print status message for successful MQTT connection

} else {

Serial.print("failed, rc="); // Print status message for failed MQTT connection

Serial.print(client.state()); // Print MQTT connection state

Serial.println(" try again in 5 seconds"); // Print suggestion for reconnection delay

delay(5000); // Delay for 5 seconds before retrying

}

}

}

# Bibliography

1. **telecomcaburn.** caburn telecom. IoT-Based Environmental Monitoring: How IoT Technology Can Help Businesses Go Green.

2. **BurgeSimon.** ISJ. What is IoT & Why IoT is Important. 2023 July 1. https://internationalsecurityjournal.com/why-iot-is-important/.

3. **DJames.** Ijareeiepapublished.

4. **CuninThomas, McGrathSean MacNameeCiaran.** Environmental monitoring based on Internet of Things Technology. 2018 December 04-06.

5. IoT-Based Environmental Monitoring: Types and Use Cases. Connect with confidence . 2022 April 15. https://www.digi.com/blog/post/iot-based-environmental-monitoring.

6. **FangShifeng.** An Integrated System for Regional Environmental Monitoring and Management Based on Internet of Things. 2014 January 27. https://ieeexplore.ieee.org/abstract/document/6725615.

7. **MoisGeorge, FoleaSilviu 和 SanislavTeodora.** Analysis of Three IoT-Based Wireless Sensors for Environmental Monitoring. 2017 March 27. https://ieeexplore.ieee.org/abstract/document/7887698.

8. **ShahJalpa 和 MishraBiswajit.** IoT enabled environmental monitoring system for smart cities. 2016 January 22-24. https://ieeexplore.ieee.org/abstract/document/7562757.

9. **MahmoodMehdi Faris Mahmood BasilDiaa.** Data Acquisition of Greenhouse Using Arduino. 2014. https://www.iasj.net/iasj/download/e746b4694f0676b1.

10. **Vidal-PardoAntonio.** Design and Development of a 5-Channel Arduino-Based Data Acquisition System (ABDAS) for Experimental Aerodynamics Research. 2018 Juy 22. https://www.mdpi.com/1424-8220/18/7/2382.

11. **Majid KaramiViola McMorrow, Liping WangGabrielle.** Continuous monitoring of indoor environmental quality using an Arduino-based data acquisition system. 2018 September. https://www.sciencedirect.com/science/article/abs/pii/S2352710218301025.

12. **BezboruahTulshi.** A Real-Time Data Acquisition System for Monitoring Sensor Data. 2018 June .(https://www.researchgate.net/profile/Pratiksha-Sarma-2/publication/326398953\_A\_Real-Time\_Data\_Acquisition\_System\_for\_Monitoring\_Sensor\_Data/links/5b4a05b7aca272c609409d43/A-Real-Time-Data-Acquisition-System-for-Monitoring-Sensor-Data.pdf.

13. **James P. Grinias†T. Whitfield, Erik D. Guetschow, and Robert T. KennedyJason.** An Inexpensive, Open-Source USB Arduino Data Acquisition Device for Chemical Instrumentation. 2016 June 22 . https://pubs.acs.org/doi/full/10.1021/acs.jchemed.6b00262.

14. What is MQTT? https://aws.amazon.com/what-is/mqtt/).

15. What is MQTT? A practical introduction. https://www.opc-router.com/what-is-mqtt/#:~:text=MQTT%20(Message%20Queuing%20Telemetry%20Transport,%2Dmachine%20(M2M)%20communication.

16. **ChengDinculeană andXiaochunDan.** Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices. 2019 Febuary 27. https://www.mdpi.com/2076-3417/9/5/848.

17. **R A Atmoko1Riantini1 and M K Hasin1R.** IoT real time data acquisition using MQTT protocol. Journal of Physics: Conference Series. 2017 Octuber 27 . https://iopscience.iop.org/article/10.1088/1742-6596/853/1/012003/meta.

18. **Dipa SoniMakwanaAshwin.** A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT). 2017 April. https://www.researchgate.net/profile/Dipa-Soni/publication/316018571\_A\_SURVEY\_ON\_MQTT\_A\_PROTOCOL\_OF\_INTERNET\_OF\_THINGSIOT/links/58edafd4aca2724f0a26e0bf/A-SURVEY-ON-MQTT-A-PROTOCOL-OF-INTERNET-OF-THINGSIOT.pdf.

19. What is Node-RED data visualization? What are the advantages and application scenarios? 2023. https://www.iotrouter.com/what-is-node-red-data-visualization-what-are-the-advantages-and-application-scenarios/.

20. Node-RED Data Visualization. https://www.iqhome.org/solutions/node-red.

21. **OkhovatElham BauerMichael.** Monitoring the Smart City Sensor Data Using Thingsboard and Node-Red. 2022 November 21 . https://ieeexplore.ieee.org/abstract/document/9604421.

22. **RizwanRaffay.** Implementation of ANN-Based Embedded Hybrid Power Filter Using HIL-Topology with Real-Time Data Visualization through Node-RED. 2021 November 1. https://www.mdpi.com/1996-1073/14/21/7127.

23. **Katalin FerenczDomokosJozsef.** Using Node-RED platform in an industrial environment. 2020 Feb. https://www.researchgate.net/profile/Katalin-Ferencz/publication/339596157\_Using\_Node-RED\_platform\_in\_an\_industrial\_environment/links/5e5ab48c4585152ce8fc6a6c/Using-Node-RED-platform-in-an-industrial-environment.pdf.

24. Internet of Things security challenges and best practices. 2022. https://www.kaspersky.com/resource-center/preemptive-safety/best-practices-for-iot-security.

25. KORE. https://www.korewireless.com/news/key-security-considerations-for-iot-implementation.

26. IoT Security Challenges and Problems. BALBIX. 2023. https://www.balbix.com/insights/addressing-iot-security-challenges/.

27. **SarmilaK.B. 和 ManisekaranS.V.** A Study on Security Considerations in IoT Environment and Data Protection Methodologies for Communication in Cloud Computing. 2019 Oct 31. https://ieeexplore.ieee.org/abstract/document/8888414.

28. A Study on Security Considerations in IoT Environment and Data Protection Methodologies for Communication in Cloud Computing. 2019 oct 31. https://www.ltts.com/sites/default/files/whitepapers/2017-07/whitepaper\_security-considerations-for-internet-of-things.pdf.

29. **MooreSamuel.** IoT reliability: a review leading to 5 key research directions. ResearchGate. 2020 Aug. https://www.researchgate.net/publication/343512706\_IoT\_reliability\_a\_review\_leading\_to\_5\_key\_research\_directions.

30. **VAAMICHAEL.** IoT and reliability. Force Technology. 2016 Oct . https://forcetechnology.com/en/articles/iot-reliability.

31. **Guilin ZhaoXingLiudong.** Reliability analysis of IoT systems with competitions from cascading probabilistic function dependence. ScienceDirect. 2020 June. https://www.sciencedirect.com/science/article/abs/pii/S0951832019307744.

32. **XingLiudong.** Reliability in Internet of Things: Current Status and Future Perspectives. IEEE. 2020 May 07. https://ieeexplore.ieee.org/abstract/document/9089244.

33. How to reduce energy consumption when using an IoT connection. embeddedcomputing. 2018 April 18. https://embeddedcomputing.com/technology/iot/how-to-reduce-energy-consumption-when-using-an-iot-connection?gad\_source=1&gclid=Cj0KCQjwjLGyBhCYARIsAPqTz1-WQtZi9IHom8hxcxUC3WX8J06oj9OuNUuO0WN8wmgDm\_Ywm3xwuNIaAii0EALw\_wcB.

34. **OnatAhmet.** Energy Considerations. 2019. http://wireless.ictp.it/school\_2019/slides/Energy\_IoT.pdf.

35. The Ultimate Guide to Power Consumption & Energy Efficiency in IoT. Nabto. 2022. https://www.nabto.com/ultimate-guide-power-consumption-energy-efficiency-iot/.

36. The Power of Models: Modeling Power Consumption for IoT Devices. IEEE. 2015 July 12. https://ieeexplore.ieee.org/abstract/document/7122861.

37. **MahmoudS.Mahmoud.** A Study of Efficient Power Consumption Wireless Communication Techniques/ Modules for Internet of Things (IoT) Applications. 2016 April. http://repository.embuni.ac.ke/handle/123456789/825.

38. **JayakumarHrishikesh.** Energy-efficient system design for IoT devices. IEEE. 2016 March 10. ://ieeexplore.ieee.org/abstract/document/7428027.

39. Nano 33 IoT. https://docs.arduino.cc/hardware/nano-33-iot/#features.

40. Raspberry Pi Documentation. RaspBerry Pi. https://www.raspberrypi.com/documentation/computers/getting-started.html.

41. Running on Raspberry Pi. Node Red. https://nodered.org/docs/getting-started/raspberrypi.

42. Installing Mosquitto on to the PI. iotasmarterplanet. https://iotasmarterplanet.wordpress.com/mqtt/installing-mosquitto-on-to-the-pi/.

43. Integreted Enviournmental Unit. BME280. https://www.bosch-sensortec.com/media/boschsensortec/downloads/product\_flyer/bst-bme280-fl000.pdf.