Tampere University of Applied Sciences

IoT-Enabled Data Visualization and Analysis for Small-Scale Wastewater Treatment

Israt Jahan Sumiya

BACHELOR'S THESIS
April 2024

Software Engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Name of the Degree Programme
Software Engineering

AUTHOR:
Israt Jahan Sumiya
IoT-Enabled Data Visualization and Analysis for Small-Scale Wastewater Treatment

Bachelor's thesis 40 pages, appendices 3 pages
April 2024

This thesis proposes an IoT-enabled data visualization and analysis system for enhancing the educational experience in water quality monitoring. The system utilizes modern IoT technologies to collect, store, visualize, and analyze environmental data, with a focus on air quality metrics such as temperature and humidity. Through integration with platforms like ThingsBoard and Grafana, real-time monitoring, predictive analytics, and comprehensive data analysis are made possible. The thesis presents detailed descriptions of the system architecture, hardware components, software components, data collection and storage processes, data visualization and analysis results, and a comparison between existing and proposed solutions. Future work is outlined, suggesting for further enhancement and adaptation of the system for water quality monitoring. Overall, the thesis utilizes IoT technologies in environmental monitoring, with the potential to significantly improve data accuracy, real-time monitoring capabilities, and prepare students for industry practices.

Keywords: Water Quality Monitoring, Internet of Things (IoT), Data Visualization, Data Analysis, ThingsBoard, Grafana, InfluxDB

CONTENTS

ABBREVIATIONS AND TERMS

DO= Dissolved Oxygen
PH= Potential of Hydrogen
IoT=Internet of Things
LoRa=Long Range
LPWAN=Low-Power Wide Area Network
WQ= Water Quality
RPI= Raspberry Pi
Arduino=An Open-source platform for electronics projects, combining hardware and software for easy prototyping and programming.
Arduino Uno= A popular Arduino board featuring a user-friendly design, based on the ATmega328 microcontroller. It is compatible with various sensors and actuators.
NodeMCU = An open-source firmware and development kit tailored for IoT applications. It uses the ESP8266 or ESP32 Wi-Fi module, allowing devices to connect to the internet.

1 Introduction

Effective monitoring and analysis of water quality plays an important role in the field of water engineering, especially in small-scale wastewater treatment systems. At TAMK's Water Engineering Department, a specific course is designed to equip students with practical knowledge and skills for collecting and analysing data related to water temperature, DO, and PH.

Currently, students in the course use Arduino and a Real-Time Clock (RTC), to gather data from various sensors and store it on an SD card. The data is then visualized using Microsoft Excel. While this method is valuable, there is a growing need for a more advanced and reliable approach that incorporates with modern IOT technologies. These technologies would enable continuous data visualization and analysis.

The goal of this thesis is to enhance the course provided by TAMK's Water Engineering Department. Through this study, will propose implementations of an IoT-enabled data visualization and analysis system. Specifically, will review relevant literature and articles to demonstrate how water quality data analysis and visualization can be conducted using modern technologies and platforms. An example case will be presented, showcasing the utilization of popular cloud technologies such as ThingsBoard, InfluxDB, and Grafana for data visualization and analysis. This approach aims to provide students with hands-on experience in monitoring and analysing IoT data.

1.1 Internet of Things (IoT) and Its Applications

IoT in the context of water quality monitoring refers to the integration and utilization of electronic sensors, communication technologies, and data processing capabilities to establish a comprehensive and real-time monitoring system for assessing and managing water quality. (Jiao & Liu, 2018)

The LPWAN technologies, such as LoRa, offer a solution by enhancing monitoring accuracy and efficiency. Traditional IoT technologies face various limitations, including the need for a continuous and stable power source, limited communication ranges that make it challenging to cover expansive areas or remote locations, scalability issues when accommodating many devices or sensors, and higher costs associated with deploying and maintaining IoT systems. However,

LoRa's advantages in terms of low power consumption, long-range communication, and cost-effectiveness make it a promising choice for water quality monitoring. (Jiao & Liu, 2018)

One notable application of IoT technology involves the integration of wireless sensor networks with IoT devices like NodeMCU. This integration enables the continuous collection of data from a range of WQ parameters. In a specific study (Ajith et al., 2020), an IoT-based system was developed, incorporating a network of sensors and wireless communication devices to monitor parameters such as pH, humidity, temperature, and pollutant levels in real-time. These sensors transmitted the collected data to a cloud-based platform using IoT devices like NodeMCU, facilitating remote access and analysis. By using IoT technology, this system ensured the prompt detection of water contamination, offering a cost-effective, eco-friendly, and efficient solution to address environmental concerns and enhance water resource management. This application of IoT in WQ monitoring has the potential to significantly improve environmental practices and prove invaluable in various related fields.

Another WQ system (P. Baskaran et al.,2019) utilized sensors to continuously measure key parameters like temperature, pH, turbidity, and conductivity in real time. The collected data is processed by an RPI B+ model, and cloud computing is employed to make this information accessible via the internet. Machine learning, particularly Naive Bayes classification, is integrated into the system for automated assessment of water quality levels based on pH and turbidity values. This IoT system showcases how the synergy of sensor data, cloud computing, and machine learning can provide high-frequency, automated monitoring of water quality, offering potential solutions for safe water supply challenges.

1.2 Data Visualization Techniques and Tools

1.2.1 Static Visualization

Static visualization is a commonly used technique for presenting data in various fields including scientific research, business analytics, and information technology. It involves creating graphical representations of data that do not allow for user interaction, providing a fixed view of the information. Static visualization holds an author-driven approach, where the presentation of data is

predetermined and does not allow for the reader or viewer to interact with the data. (Mahajan & Gokhale, 2018)

One commonly used technique is a graphical static visualization, which focuses on presenting a single view of the selected data story. This approach provides a concise and clear representation of the data and is well-suited for small-sized datasets. Another popular technique is the use of charts and graphs to represent the data visually. These visual representations can include bar charts, line graphs, pie charts, and scatter plots, among others. These static visualizations allow for a quick and easy understanding of the data, as they provide a visual summary of the information. (Mahajan & Gokhale, 2018)

Static data visualization tools play a crucial role in presenting IoT data collected from sensors and devices in a meaningful and understandable way. Among the prominent tools, Tableau has been widely adopted due to its user-friendly interface and extensive data integration capabilities (Tableau, 2023). Similarly, Power BI by Microsoft is recognized for its robust data visualization features, making it a popular choice for IoT applications (Microsoft, 2023). Another tool is D3.js, a JavaScript library that offers flexibility and customization in creating static visualizations for IoT data (Bostock et al., 2011). These tools allow us to create static visualizations, such as charts, graphs, and dashboards, that provide insights into historical IoT data, facilitating informed decision-making and trend analysis.

1.2.2 Dynamic Visualization

Dynamic data visualization in IoT is essential for gaining insights into rapidly changing data streams (Mahajan & Gokhale, 2018). Grafana, an open-source software, has emerged as a popular choice due to its flexibility and ease of use. Grafana allows users to create dynamic dashboards that connect to various data sources, making it suitable for visualizing real-time sensor data and IoT metrics (Grafana Labs, 2023). Additionally, InfluxDB, often integrated with Grafana, serves as a time-series database optimized for storing and querying time-sensitive IoT data, thus complementing the dynamic visualization process (InfluxData, 2023).

The need for real-time analytics in IoT applications has also led to the adoption of Apache Kafka and Apache Flink. Apache Kafka is an industry-leading platform for real-time data streaming, known for its capabilities in efficiently handling large

volumes of IoT data through distributed event streaming (Abhishek K. S,2023). On the other hand, Apache Flink is a versatile stream processing framework that plays a pivotal role in empowering IoT practitioners to react swiftly to critical events. This robust open-source tool excels at handling complex event processing and offers dynamic visualization capabilities (Gębiś, W.,2022).

Furthermore, advancements in edge computing have had a profound impact on dynamic IoT visualization. Edge devices, equipped with processing capabilities, can preprocess and filter data at the source, reducing the volume of data transferred to central servers. This localized processing enhances the efficiency of dynamic visualization tools and minimizes latency in critical applications (Shi et al., 2016).

Mainly, the choice of tools and techniques depends on the specific requirements and objectives of IoT applications, emphasizing the need for continuous exploration and adaptation in this rapidly evolving field.

1.3 Data Analysis Techniques

The analysis of WQ is reliant on various statistical techniques. Commonly employed descriptive statistics, such as mean, median, and standard deviation, enable the comprehensive presentation of water quality data (Helsel and Hirsch, 2002).

Advanced machine learning algorithms, such as multiple linear regression, gradient boosting, Lasso regression, Support Vector Machines (SVM), decision trees, and Artificial Neural Network (ANN), are adopted to predict and classify water quality based on key parameters including temperature, pH, turbidity, and coliforms. These data analysis techniques are instrumental in not only assessing water quality but also in enhancing water quality management practices, ultimately contributing to more efficient monitoring, and reduced associated costs. (Azrour et al.,2021)

Time series data analysis techniques, including ARIMA, SARIMA, and the Prophet model, are effective tools for forecasting water quality trends over time. These techniques allow for the identification of patterns and prediction of future water quality conditions, contributing to more accurate and timely decision-making processes (A. P. Kogekar et al.,2021)

In addition to these techniques, the integration of Geographic Information Systems (GIS) has been instrumental in the spatial analysis of water quality parameters. By overlaying the results of laboratory analyses onto the study area shapefile, GIS has enabled the creation of spatial variation maps, allowing for a visual representation of the distribution of key parameters such as Total Dissolved Solids (TDS), Electrical Conductivity (EC), and pH across different regions of the reservoirs. This approach has facilitated a more comprehensive understanding of the spatial patterns and has aided in the identification of areas with potential contamination or variations in water quality parameters. (Oseke et al, 2021)

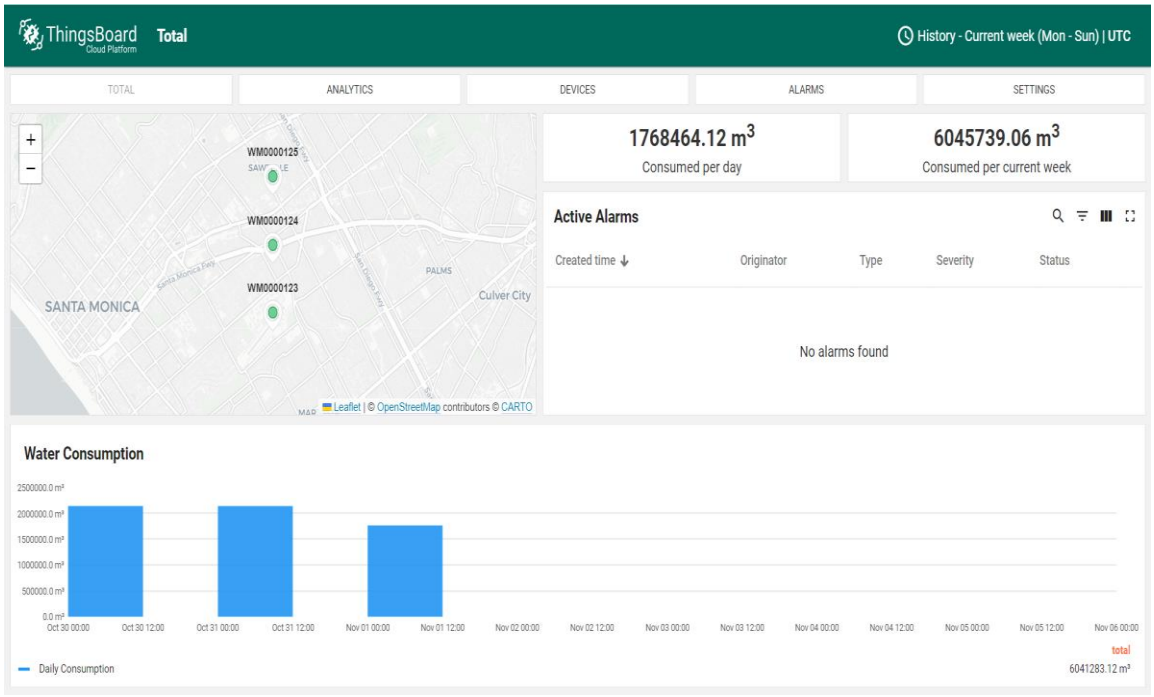
1.4 IoT Data Visualization and Analysis Stack

1.4.1 ThingsBoard

ThingsBoard is an open-source IoT platform that enables users to collect, store, visualize, and analyze data from IoT devices. It is suitable for data visualization and analysis, particularly for water quality data, due to its powerful features and flexibility that cater to various needs within the water metering industry. Its applicability lies in its ability to facilitate real-time data visualization, reliable data collection, and seamless integration with various IoT devices. Specifically, for water quality data, it can provide insights into water consumption patterns, identify anomalies, and facilitate prompt action through its alarm and notification systems. (ThingsBoard, 2023)

In the context of WQ data visualization and analysis, ThingsBoard stands out as a comprehensive and robust platform for numerous reasons. First, it ensures the reliable collection of data from IoT devices and sensors, thus maintaining the accuracy of water quality data. Its powerful rule engine facilitates the processing of data, enabling the generation of alerts and valuable insights for prompt responses to potential water quality issues. Moreover, the platform's advanced visualization capabilities empower users to create interactive dashboards that offer real-time and historical data, aiding in the identification of trends and anomalies. Customizable end-user dashboards further facilitate the easy sharing of monitoring results, tailored to specific requirements. Offering both on-premises and cloud deployment options, ThingsBoard provides users with deployment flexibility that suits their infrastructure and security needs. It also enables remote

control and over-the-air updates for IoT devices, ensuring seamless management and maintenance of water metering solutions. With its scalability and high availability, ThingsBoard efficiently handles large-scale deployments, making it suitable for projects involving numerous water meters. The platform prioritizes data security, employing industry-standard encryption algorithms during data transfer to safeguard sensitive water quality data. Its multi-tenancy support allows for the efficient management of multiple customers, tenants, and devices, catering to the needs of diverse stakeholders within the water quality sector. Additionally, ThingsBoard's compatibility with various connectivity options for IoT sensors, including LoRaWAN, SigFox, and NB-IoT, makes it a versatile solution compatible with a wide range of water metering devices. Collectively, these features make ThingsBoard an ideal solution for effective water quality data visualization and analysis, presenting a comprehensive and versatile tool for research and implementation purposes.(ThingsBoard, 2023)



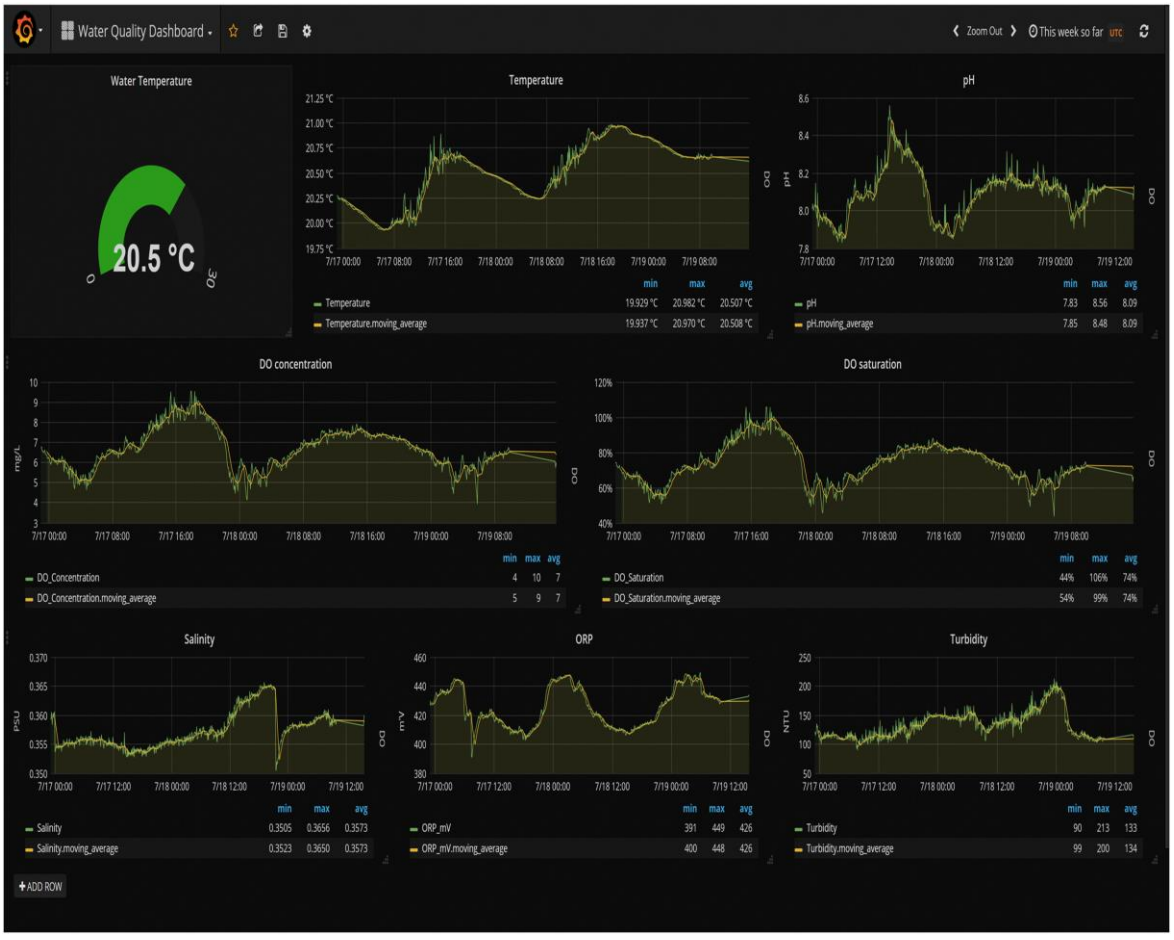
Picture 1: Water Metering Dashboard Tamplate (ThingsBoard,2023)

1.4.2 Grafana

Grafana is a widely used open-source analytics and monitoring software that serves as a user interface (UI) for visualizing and understanding time-series data. With its user-friendly interface and extensive customization options, it has gained

popularity in various fields, including data analysis, system monitoring, and IoT applications. One of the key features of Grafana is its ability to integrate with multiple data sources, such as databases, cloud services, and IoT platforms, allowing users to consolidate and analyze data from diverse sources within a unified dashboard. Grafana primarily retrieves data from connected databases and other sources to generate visualizations and insights (Grafana Labs,2023).

Regarding the integration of data from different sensors and monitoring devices, Grafana does not directly support sensors but rather acts as a dashboard where data gathered from sensors and devices is displayed and analyzed. Users can connect Grafana to databases or other data repositories where sensor data is stored, enabling comprehensive real-time analysis and visualization. The interactive dashboard features of Grafana enable users to monitor trends and anomalies efficiently, facilitating comprehensive analysis and decision-making based on sensor data (Grafana Labs,2023).



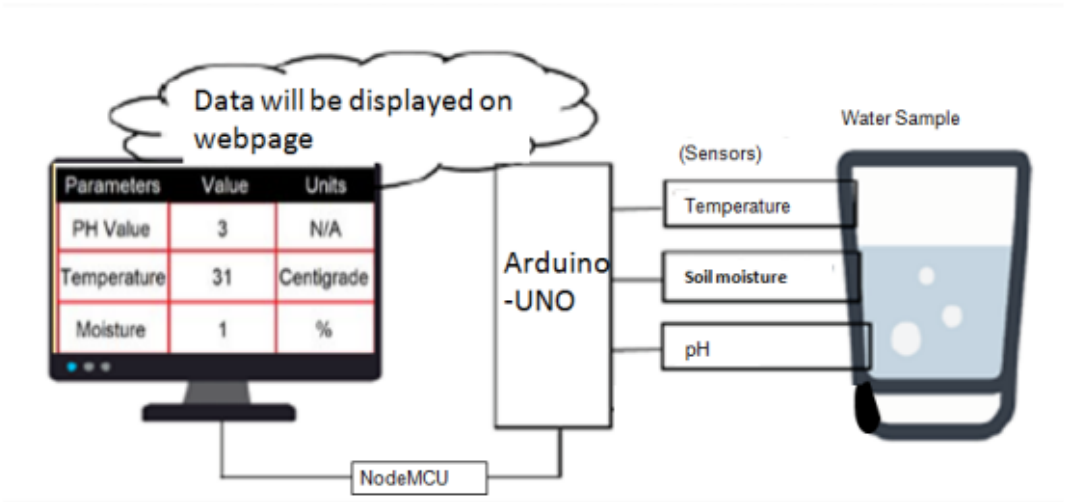
Picture 2: Example Of Using Grafana for Realtime Water Quality Data visulization (Chen et al., 2018)

1.4.3 InfluxDB

InfluxDB is a high-performance, open-source time-series database designed for efficiently storing and retrieving large volumes of time-stamped data. Its robust architecture and optimized storage format make it well-suited for handling water quality data collected from IoT devices. InfluxDB integrates seamlessly with Grafana, enabling users to utilize its querying capabilities for data analysis and visualization.It uses compression techniques and retention policies that allow for long-term storage of historical data while maintaining fast query performance. This makes it suitable for storing water quality data over extended periods, enabling retrospective analysis and trend identification. InfluxDB also provides data aggregation and downsampling options, allowing users to efficiently analyze large datasets and extract meaningful insights (InfluxData, 2023).

1.5 Examples of IoT Devices for Water Quality Monitoring

Arduino stands out as a widely utilized microcontroller enabling the integration of diverse sensors crucial for water quality assessment, such as temperature, pH levels, dissolved oxygen (DO), and conductivity. Keshipeddi's work (2021) employed Arduino Uno and NodeMCU as microcontrollers to interface with sensors for water quality monitoring. The system, encompassing pH, temperature, and soil moisture sensors, processes data through Arduino Uno before transmitting it to NodeMCU, which serves as a core controller for data transfer to a web browser.



Picture 3: Block diagram of IoT Based Smart Water Quality Monitoring System (Keshipeddi et al. ,2021).

In another study, Hamid et al. (2020) introduced a Smart Water Quality Monitoring System (SWQMS) tailored for public swimming pools. The IoT device comprises NodeMCU, integrated with an ESP8266 Wi-Fi module; a pH sensor with built-in LEDs for power indication; a DS18B20 digital temperature sensor; and an IoT platform, Ubidots, for data upload and visualization. This system enables real-time monitoring of pH levels and temperature in swimming pool water, featuring automated pH regulation through water pumps to promptly maintain water quality. The device offers an efficient and cost-effective alternative to conventional manual monitoring methods.

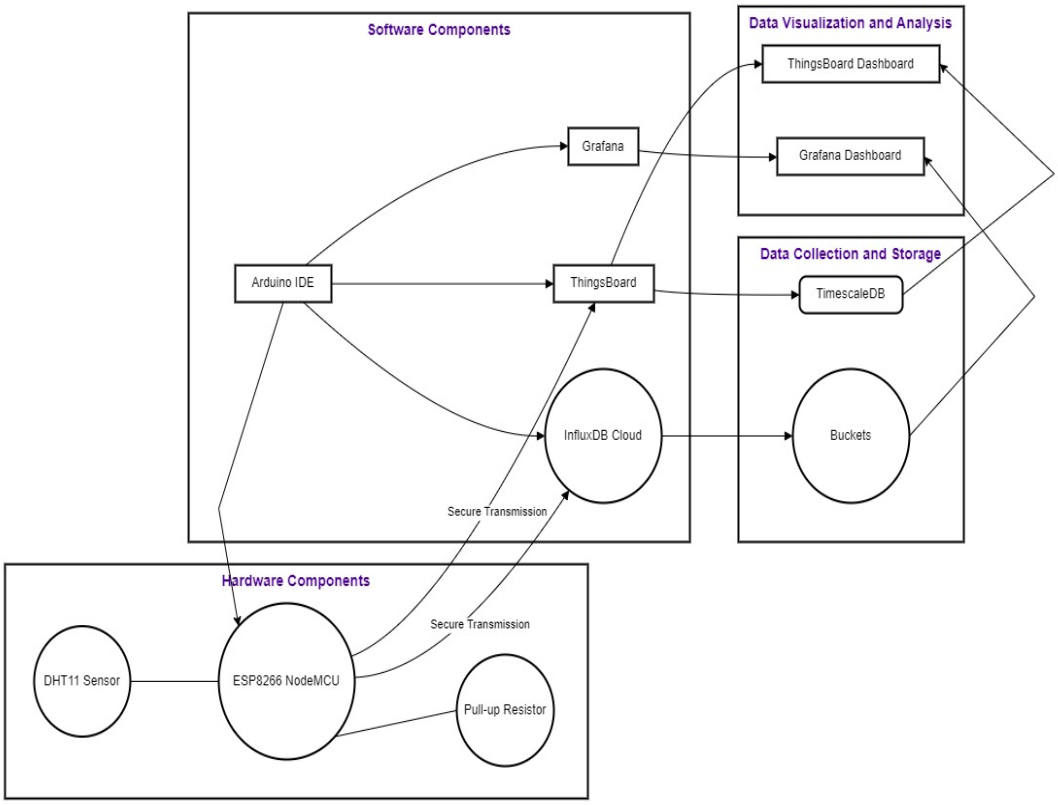
Additionally, Salleh et al.'s research (2022) proposes an IoT-based system addressing water overflow challenges faced by households. The system utilizes an HC-SR04 Ultrasonic Sensor connected to a NodeMCU ESP8266 microcontroller to monitor water levels in tanks in real-time. The Blynk App on a mobile phone serves as the interface, displaying water percentages and eliminating the need for manual checks. Emphasizing the importance of water conservation, the low-cost system contributes to efficient monitoring, with potential future enhancements including the incorporation of turbidity elements for comprehensive water quality assessment.

These examples highlight how microcontrollers are utilized in IoT water quality monitoring, enabling data processing, connectivity, integration, control, and energy efficiency.

2 Methodology

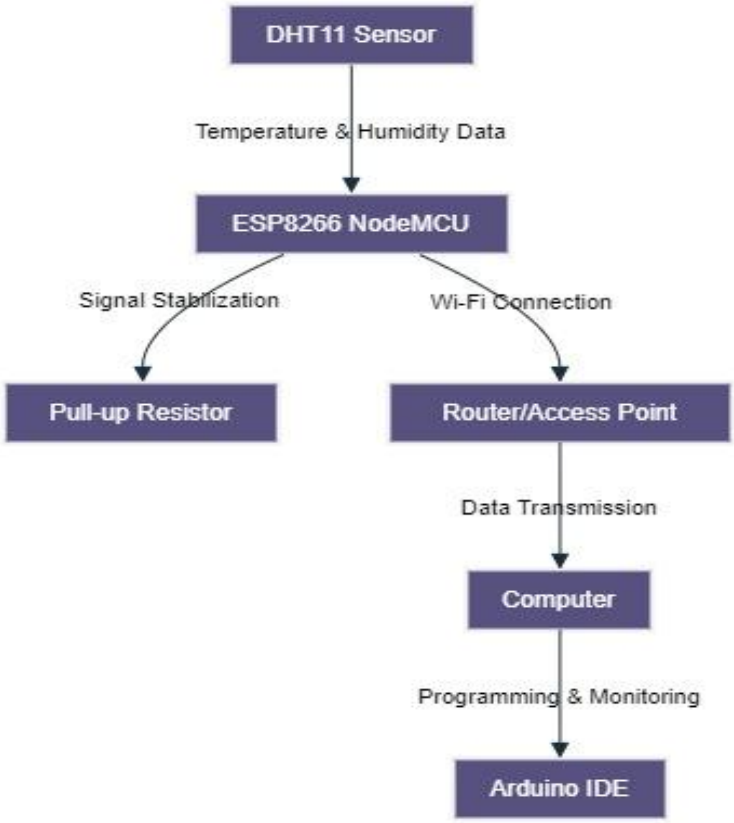
2.1 System Architecture

As an example, a small IoT device is utilized, which measures air quality metrics such as temperature and humidity. In implementation, the main objective was to provide IoT enabled data visualization and analysis.



Picture 4: System Architecture for IoT-Based Air Quality Monitoring and Data Analysis.

2.2 Hardware Components & Block Diagram



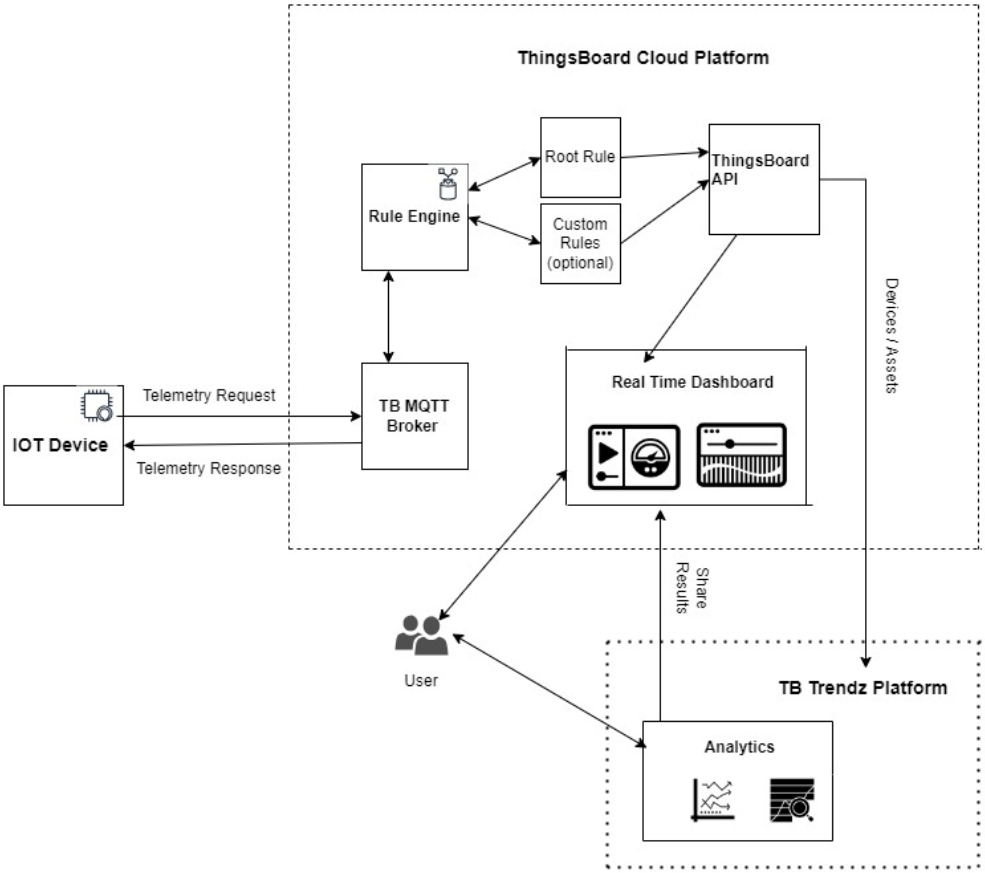
Picture 5: Air Quality Measuring HW Block Diagram.

The DHT11 sensor is utilized for monitoring temperature and humidity. It operates within a temperature range of 0°C to 50°C (32°F to 122°F) and a humidity range of 20% to 80% (Zahid Ali, 2019). The sensor sends air quality data to the ESP8266 NodeMCU microcontroller. Specifically, we utilized the Lolin D1 Mini, a compact Wi-Fi development board based on the ESP8266 microcontroller. The microcontroller processes this data and ensures signal stability with a pull-up resistor. It connects to the internet via a router or access point using Wi-Fi. A computer running the Arduino IDE is used to program the ESP8266 NodeMCU and monitor data transmission, facilitating further processing and visualization of the collected data.

2.3 Software Components

In this section, we will explore the key software components used for visualizing and analyzing data through the proposed cases.

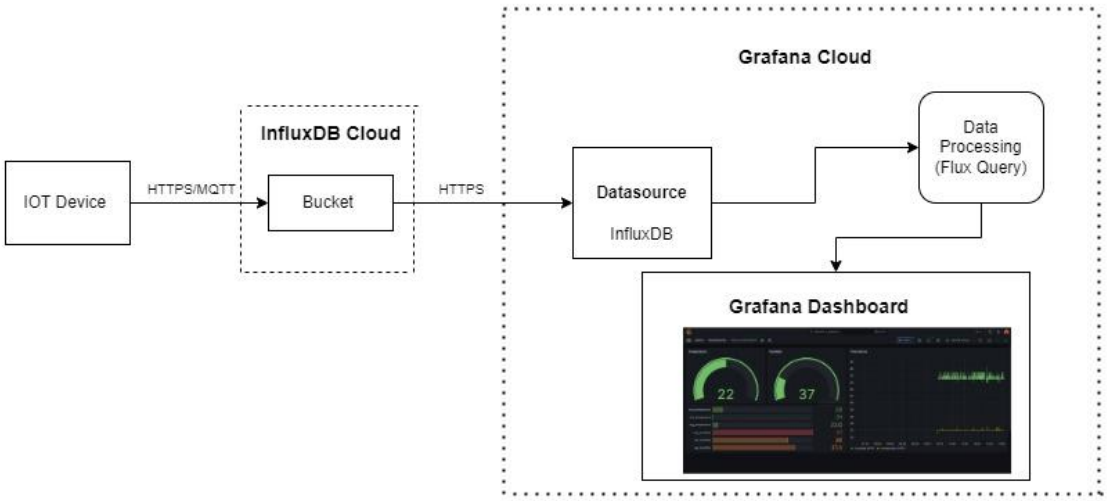
2.3.1 ThingsBoard



Picture 6: Proposed Architecture for IOT Data Visualization and Analysis Utilizing ThingsBoard.

When an IoT device sends out telemetry data, it undergoes a series of steps within the ThingsBoard environment. Initially, the data is dispatched to the ThingsBoard MQTT broker, where it is received and then processed by the ThingsBoard server node. The server node, using the ThingsBoard API, assesses and processes the incoming data according to predefined rules and transformations. After processing, the telemetry data is stored using the ThingsBoard API in the platform's dedicated data storage. Then the data is then accessible through a dashboard, which retrieves and displays the information, providing real-time insights and visualizations.

2.3.2 InfluxDB and Grafana

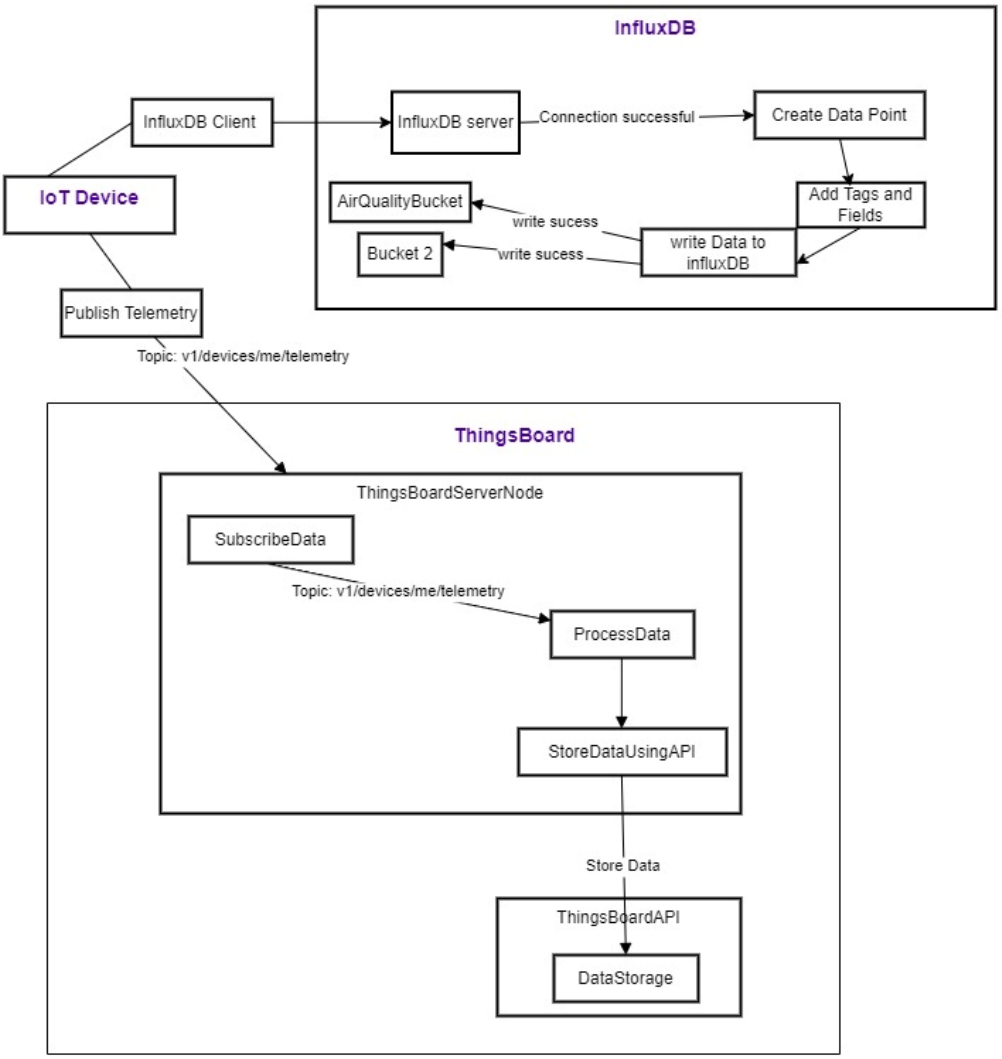


Picture 7: Proposed Architecture for Data Visualization and Analysis Utilizing InfluxDB and Grafana.

We integrated an IoT device with the InfluxDB Cloud service and Grafana to achieve effective data visualization and analysis. The IoT device securely transmits sensor data to the InfluxDB Cloud. Using the InfluxDBClient library, the IoT device establishes a secure connection to the InfluxDB Cloud, where the data is securely stored.

Grafana serves as the visualization and monitoring layer, connecting to the InfluxDB Cloud as a data source. Through Grafana's user-friendly web interface, users can effortlessly create dashboards displaying real-time or historical sensor data in various visual formats, such as charts and graphs.

2.4 Data Collection and Storage



Picture 8: IoT Data Collection and Storage for InfluxDB and ThingsBoard

In InfluxDB, the device establishes a connection to the InfluxDB server using specified credentials. It creates data points containing sensor data along with relevant tags and fields. These data points are then written to the designated InfluxDB bucket. If successful, the data is stored in the bucket. If writing fails, the data points are buffered locally for later transmission. The device periodically checks for stored data and attempts to send it when a connection is available.

In ThingsBoard, the device publishes telemetry data to the MQTT broker on a specific topic. The ThingsBoard server node subscribes to that topic, processes the incoming data, and stores it using the ThingsBoard API. This API facilitates

efficient storage and management of the telemetry data within the ThingsBoard platform.

2.5 Data Visualization and Analysis

2.5.1 Data Visualization

The ThingsBoard platform serves as the canvas for data visualization through user-created dashboards employing widgets. These dashboards enable the representation of real-time and historical sensor data using ThingsBoard API. The visualization includes time-series line charts depicting trends and patterns, along with total alarms triggered (see Picture 15).

While InfluxDB stores the data, the actual visualization takes place in Grafana. Utilizing InfluxDB Cloud as a data source, Grafana dashboards showcase current temperature and humidity. Visualization is extended to include minimum, maximum, and average values, depicted in time series charts. Additional visual representations like standard deviation and derivative analysis, offering a comprehensive view of the data dynamics (see Picture 16).

2.5.2 Data Analysis

The collected temperature and humidity data were stored in an InfluxDB bucket. It is linked to Grafana for further analysis. The Grafana dashboard was configured to calculate and display essential statistics, including the minimum, maximum, average, and standard deviation of temperature and humidity. Time series charts were utilized to observe trends and patterns over time. Additionally, derivative charts were generated to analyze the rate of change in temperature and humidity, providing deeper insights into the environmental dynamics (see picture 16). This approach ensured a thorough and meaningful analysis of the sensor data.

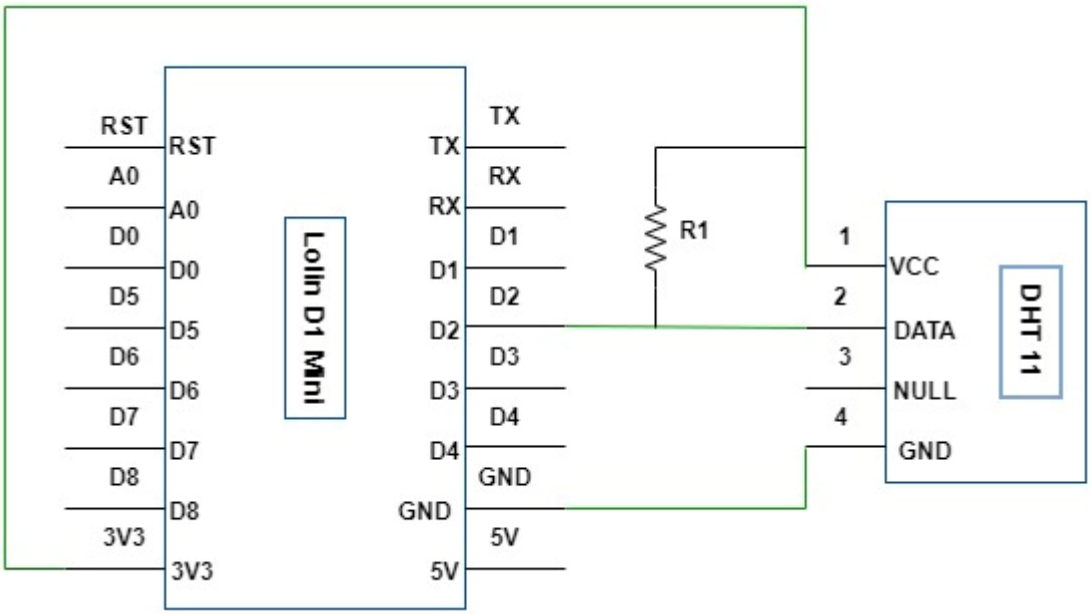
Also, We utilized the ThingsBoard Rule Engine to monitor and analyze humidity levels, generating alarms for unexpected values to enable real-time anomaly detection. Additionally, we integrated Trenz Analytics within ThingsBoard to provide predictive analytics, offering forecasts of humidity and temperature trends. This combination of rule-based processing and predictive analytics

enabled us to perform data analysis, enhancing our ability to understand, monitor, and predict environmental conditions effectively (see picture 15 &17).

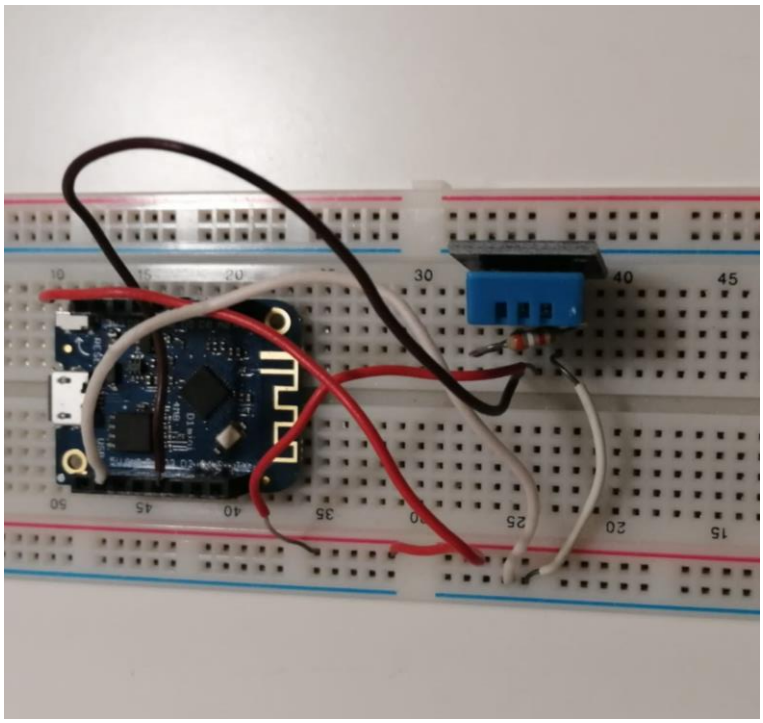
3 Implementation

3.1 Hardware Setup

The hardware setup involves connecting a DHT11 sensor to an ESP8266 NodeMCU (Lolin D1 Mini) using additional components like a resistor, jumping wires, and a breadboard.



Picture 9: Circuit diagram



Picture 10 : IOT device to meaure temperature and humidity.

3.2 Developing the Data Collection and Storage System

After setting up the software (see Appendix 3), we used both InfluxDB and ThingsBoard to efficiently capture and store environmental data. The ESP8266 microcontroller was programmed using the Arduino IDE and interfaced with a DHT11 sensor to collect temperature and humidity data (see Appendices 1 and 2). Wi-Fi credentials were configured for both InfluxDB and ThingsBoard to enable the ESP8266 to connect to the internet.

For InfluxDB integration, the InfluxDB client library was included in the ESP8266 code to facilitate communication with the InfluxDB server. The device connected to the InfluxDB server using the specified URL, organization, bucket, and authentication token. Data points were created with relevant tags and fields to encapsulate sensor readings. Error handling mechanisms were implemented to buffer data points locally if the initial write to the InfluxDB server failed, ensuring periodic checks for sending buffered data once the connection was restored.

For ThingsBoard integration, the ESP8266 used the PubSubClient library to publish telemetry data to the ThingsBoard MQTT broker. The device connected to the ThingsBoard server using the provided access token and periodically published telemetry data, formatted as JSON, to the "v1/devices/me/telemetry" topic. Connection retries and error handling ensured reliable data transmission to ThingsBoard.

3.3 Developing the Data Visualization and Analysis System

In the development of the data visualization and analysis system, dashboards were created on ThingsBoard Cloud and Grafana Cloud. ThingsBoard widgets were configured for real-time and historical sensor data visualization, while Grafana offered a versatile range of visualization options. The implementation included the incorporation of custom rules within ThingsBoard for enhanced data analysis. In Grafana, flux query can be used for data visualization and analysis.

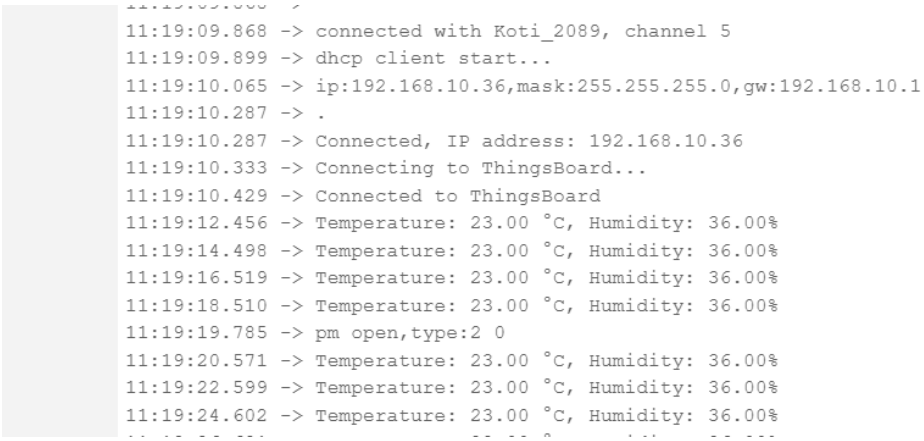
3.4 Integration of the System Components

The integration process focused on ensuring communication between hardware and software components. This involved verifying that the ESP8266 effectively transmitted data to both ThingsBoard Cloud and InfluxDB Cloud. Additionally, the configured dashboards in ThingsBoard and Grafana were examined to ensure accurate reflection of real-time and historical sensor readings.

4 Results and Analysis

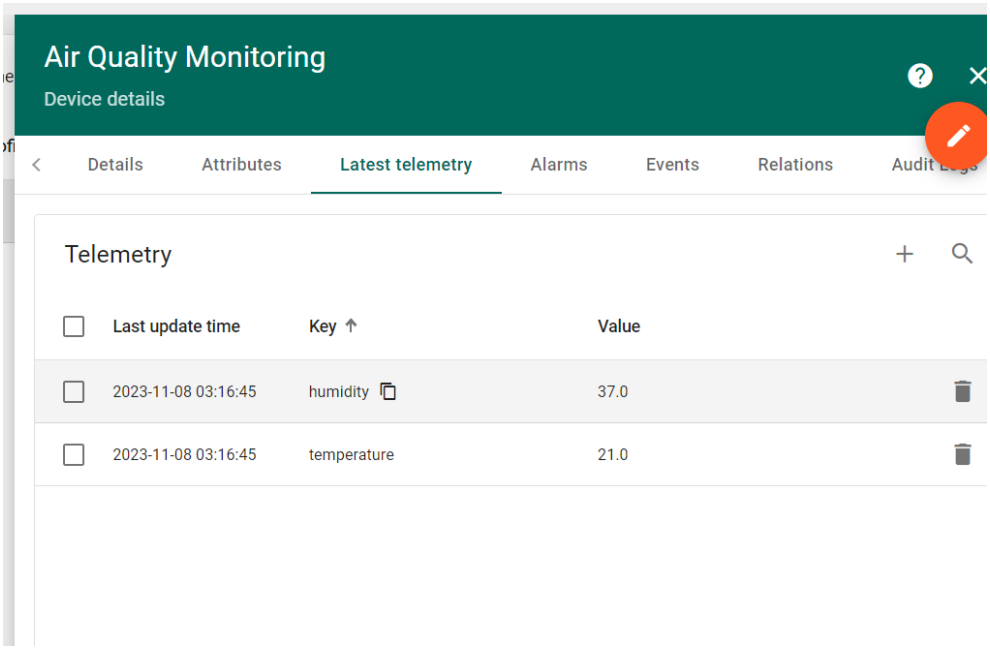
4.1 Embedded Hardware and Software Result

After uploading code (appendix 1) to esp8266, the Arduino IDE's Serial Monitor was used to display real-time temperature and humidity readings captured by the DHT11 sensor. This provided immediate feedback on the sensor's performance and ensured that the data was being accurately captured. Upon establishing a connection with the ThingsBoard Cloud platform, a confirmation message was displayed, indicating that the ESP8266 microcontroller successfully communicated with ThingsBoard. It validates the connectivity and data transmission capabilities of the device.



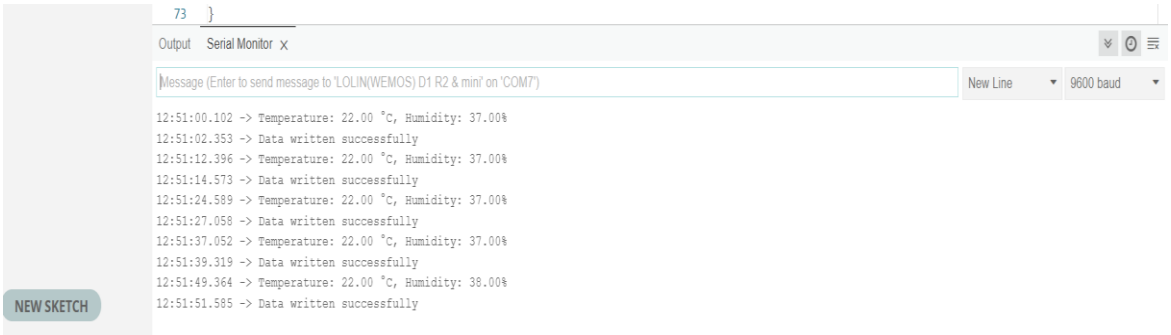
Picture 11: Arduino IDE serial monitor result for sending data to things board.

Then, the data received from the ESP8266 is showcased on the ThingsBoard device’s latest telemetry, confirming that real-time data is being sent.



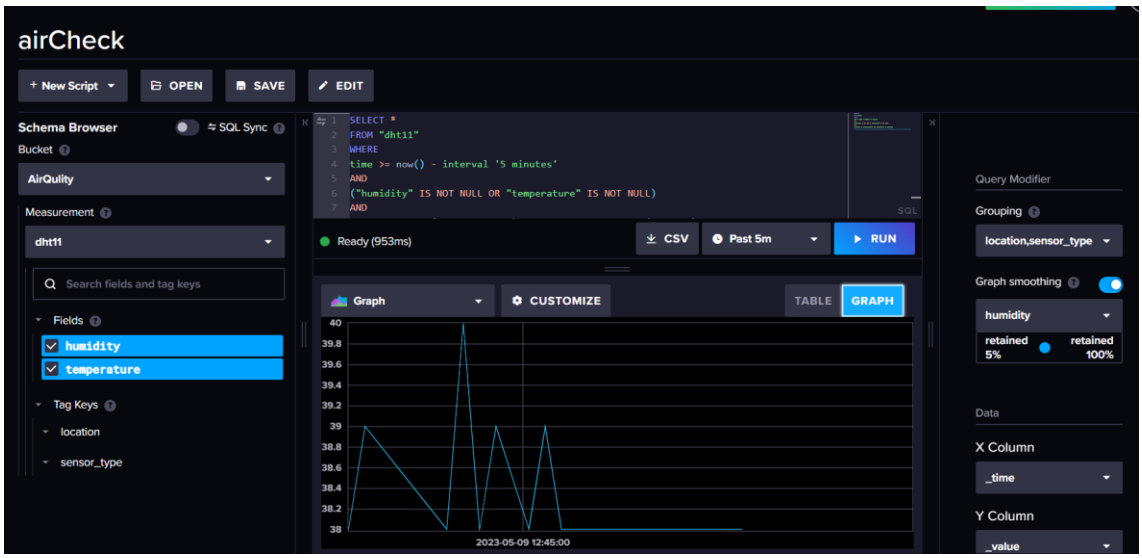
Picture 13: Air Quality Monitoring device in Things Board.

By uploading a new sketch (appendix 2) to the ESP8266, data was successfully written to InfluxDB. Also, this ensures that the sensor data is captured.



Picture 12: Arduino IDE serial monitor result for sending data to InfluxDB.

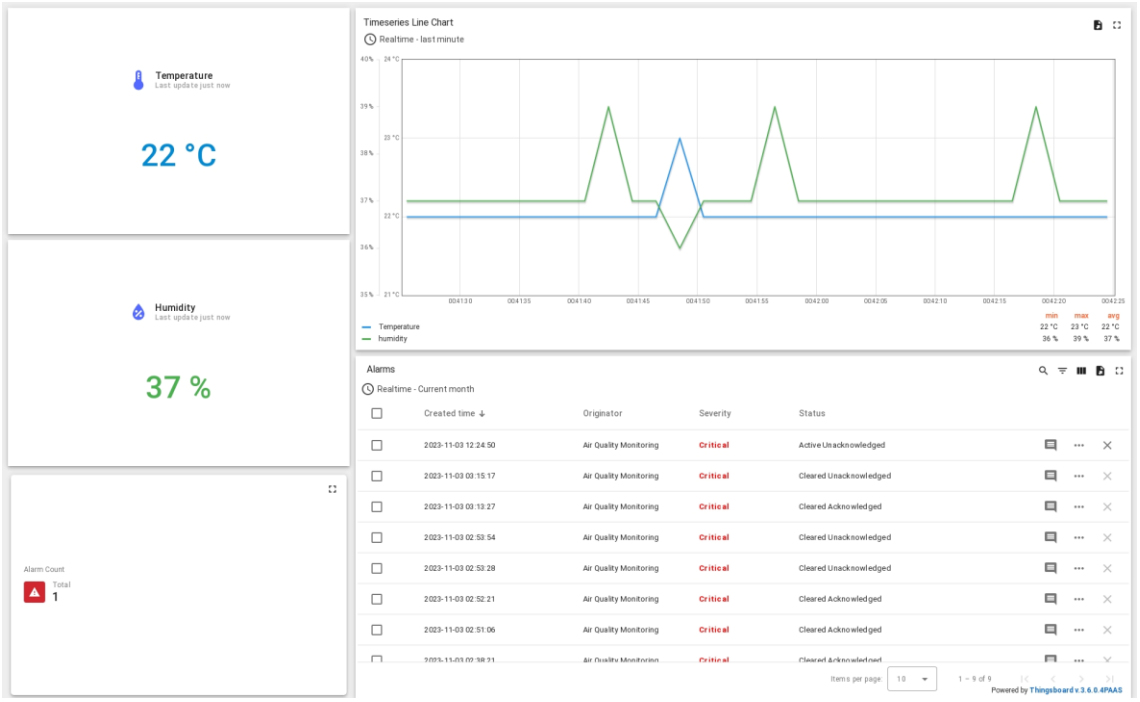
A dedicated InfluxDB Cloud bucket airCheck is used to securely store the sensor data. By querying the database, it is possible to confirm the successful transmission and storage of data.



Picture 14: Visual confirmation of successful data transmission in InfluxDB.

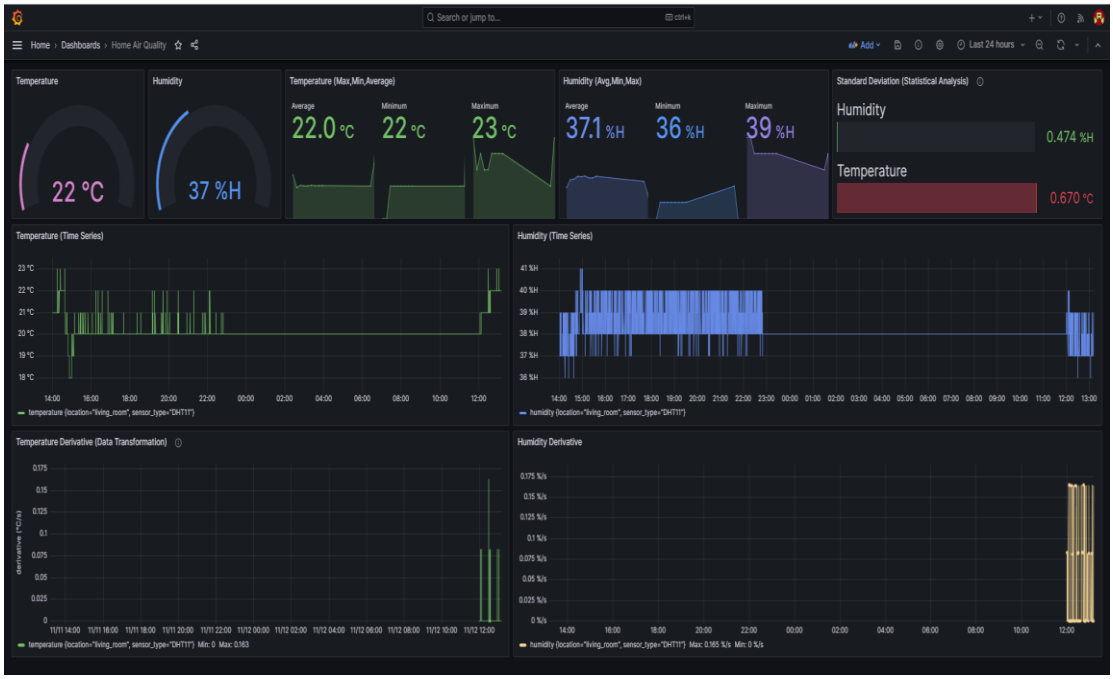
4.2 Data Visualization Results

The ThingsBoard Cloud dashboard presents a user-friendly interface with widgets displaying real-time temperature and humidity data. Custom rules, including alarms triggered by unexpected humidity levels, are visible on the dashboard. The successful implementation of these visualizations enhances the user's ability to monitor and interpret the IoT device's environmental data.



Picture 15: Data visualization in ThingsBoard.

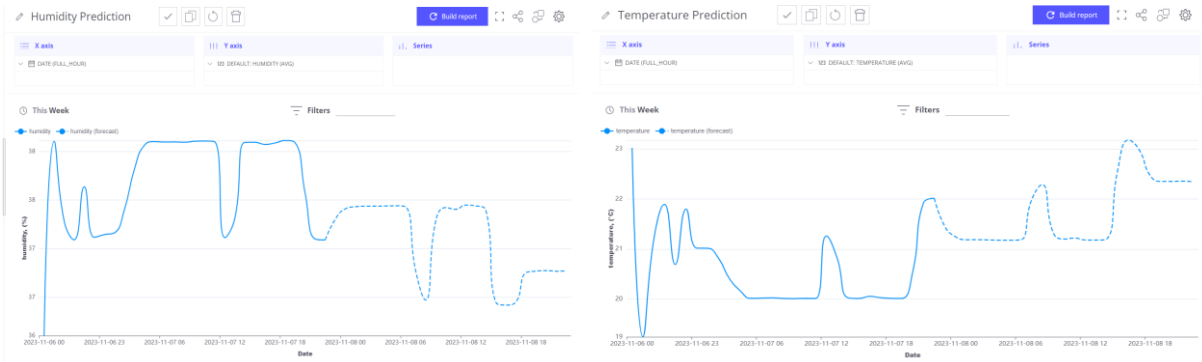
The Grafana Cloud dashboard shows a diverse set of visualizations, including current temperature and humidity, minimum, maximum, and average values, time series charts, standard deviation, and derivative analysis. This versatile dashboard offers users a holistic view of sensor data.



Picture 16: Data visualization in Grafana.

4.3 Data Analysis Results

Trenz Analytics in ThingsBoard uses Fourier transformation and average aggregation on the last one week's data to predict the next 24 hours of humidity and temperature. The screenshots reveal insightful visualizations, offering users a forecast of humidity and temperature trends. This predictive analysis enhances the system's capability to anticipate environmental changes, contributing to a more proactive approach in decision-making.



Picture 17: Predictive analysis using Trenz Analytics.

In Grafana, additional data analysis is performed through standard deviation and derivative calculations. Standard deviation visualizations provide insights into data variability, helping identify periods of unusual sensor behaviour. Derivative analysis captures the rate of change in temperature and humidity, offering valuable information on trends and transitions. These analyses, integrated into the Grafana Cloud dashboard, contribute to a more comprehensive understanding of the dynamics within the collected sensor data (see picture 16).

4.4 Comparison and Discussion of Results

This section compares the existing solution used in TAMK's Water Engineering Department course with the newly proposed IoT-enabled system and discusses the results of its implementation.

Table 1: Comparison of Existing and New Solutions

Aspect	Existing Solution	New Solution
Data Storage and Retrieval	Relies on SD cards, needing physical access to retrieve data.	Uses cloud platforms (e.g., InfluxDB, Thingsboard API) for centralized and secure data storage with remote access.
Data Visualization and analysis	Utilizes Microsoft Excel for static, manual visualizations.	Implements dynamic dashboards on platforms like ThingsBoard and Grafana for real-time interaction.
Scalability and Flexibility	Limited scalability and flexibility.	Designed for scalability and adaptability to evolving needs.
Cost-effectiveness	Lower initial costs but higher long-term maintenance.	Higher initial setup costs but more cost-effective over time due to remote monitoring.
Student Learning Experience	Limited exposure to modern IoT technologies.	Provides hands-on experience with modern IoT technologies, enhancing data visualization and analysis skills.

The discussion of results reveals the effectiveness of the implemented system in IoT-enabled data visualization and analysis for small-scale environmental monitoring. Both InfluxDB and ThingsBoard API demonstrated reliable data collection

and storage, ensuring data integrity and accessibility. While ThingsBoard provided intuitive real-time visualization with customizable widgets and custom rules for anomaly detection, Grafana offered diverse visualization options for comprehensive data analysis, including standard deviation and derivative analysis. The integration of Trenz Analytics in ThingsBoard enabled predictive analysis, forecasting humidity and temperature trends based on historical data. Overall, the system excelled in real-time monitoring, analysis, and prediction of environmental conditions, offering valuable insights for informed decision-making.

5 Conclusion

In conclusion, this thesis has presented a proposal for enhancing the educational experience in water quality monitoring at TAMK's Water Engineering Department through the implementation of an IoT-enabled data visualization and analysis system. The focus was on improving the existing methodology used by students for data collection, visualization, and analysis by incorporating modern IoT technologies.

The comparison between the proposed IoT-enabled system and the existing methodology highlighted significant advancements. The new system offers centralized and secure data storage, real-time interaction through dynamic dashboards. Moreover, it provides hands-on experience with modern IoT technologies, aligning with the evolving needs of the water engineering field.

The successful implementation of the system demonstrated its effectiveness in real-time monitoring, analysis, and prediction of environmental conditions, in the context of air quality checking. While the example IoT device used was for air quality monitoring, the principles and methodologies proposed in this thesis can be adapted for water quality monitoring with minor modifications.

Overall, the thesis has laid a solid foundation for the integration of IoT technologies into water quality monitoring. It has provided information for future research and development in this area.

6 Future Work

Moving forward, there are several ways for future work and research to be explored to further enhance the proposed system. Firstly, there is a need to adapt the system specifically for water quality monitoring by modifying it to accommodate the integration of appropriate hardware components relevant to water quality assessment. This may involve replacing the example IoT device used for air quality checking with one tailored for water quality monitoring, thus ensuring the system's applicability to the intended educational context. Integrating Wi-Fi modules into the existing systems would eliminate the reliance on SD cards and RTC clocks, streamlining data transmission and storage processes. This step would not only enhance the system's efficiency but also make it more user-friendly and aligned with modern IoT practices.

Additionally, the exploration of additional sensors could enrich the system's capabilities by capturing a broader range of water quality parameters, such as pH, dissolved oxygen, turbidity, and conductivity. By incorporating these sensors, students would have access to more comprehensive data for analysis and decision-making, thus enhancing their learning experience.

Furthermore, continuous improvement of the data visualization tools used in the system is essential. Exploring new software platforms or developing custom visualization solutions tailored to specific educational needs would provide students with more advanced and customizable visualization options, thus enriching their understanding of water quality data analysis.

Lastly, conducting real-world evaluations of the adapted system in actual water treatment facilities would be beneficial. Field trials and assessments would allow for the assessment of the system's practical applicability and effectiveness in diverse environmental conditions. This real-world feedback would be invaluable for further refinement and optimization of the system, ensuring its relevance and efficacy in preparing students for the challenges of the modern water engineering field.

REFERENCES

Jiao, D. L., & Luo, X. (2018). Water quality monitoring system based on Lo-Ra. DEStech Trans. Comput. Sci. Eng, 10, 1107-1116.

Ajith, J. B., Manimegalai, R., & Ilayaraja, V. (2020). An IoT-Based Smart Water Quality Monitoring System using Cloud. IEEE.

Baskaran, P., Selva Pandiyan, D., Jebakumar Immanuel, D., & Bhavadharini, R. M. (2019). IoT-Based Water Quality Monitoring System using Machine Learning. International Journal of Recent Technology and Engineering (IJRTE), Volume-8, Issue-4, November 2019.

Mahajan, K. N., & Gokhale, L. A. (2018, March 15). Comparative Study of Static and Interactive Visualization Approaches.

Tableau. (2023). The Tableau Platform: The world's leading analytics platform. [https://www.tableau.com/products/our-platform] (https://www.tableau.com/products/our-platform)

Microsoft. (2023). Power BI. https://powerbi.microsoft.com/

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data-Driven Documents. IEEE Transactions on Visualization and Computer Graphics, 17(12), 2301-2309.

Grafana Labs. (2023). Grafana documentation. https://grafana.com/docs/grafana/latest/

InfluxData. (2023). InfluxDB: Getting Started with InfluxDB and Grafana. https://www.influxdata.com/blog/getting-started-influxdb-grafana/

Abhishek K. S (2023). Apache Kafka: Understanding Key Concepts and Components. https://www.linkedin.com/pulse/apache-kafka-understanding-key-concepts-components-singh/

Gębiś, W. (2022, November 29). What is Apache Flink? Architecture, Use Cases, and Benefits. Retrieved from https://nexocode.com/blog/posts/what-is-apache-flink/

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. IEEE Internet of Things Journal, 3(5), 637-646.

Helsel, D. R., & Hirsch, R. M. (2002). Statistical methods in water resources. Techniques of Water-Resources Investigations, Book 4, Chapter A3.

Azrour, M., Mabrouki, J., Fattah, G., Guezzaz, A., & Aziz, F. (2021). Machine learning algorithms for efficient water quality prediction. Springer Nature Switzerland AG.

Kogekar, A. P., Nayak, R., & Pati, U. C. (2021). Forecasting of Water Quality for the River Ganga using Univariate Time-series Models. 2021 8th International Conference on Smart Computing and Communications (ICSCC), Kochi, Kerala, India, 52-57. doi: 10.1109/ICSCC51209.2021.9528216.

Oseke, F. I., Anornu, G. K., Adjei, K. A., & Eduvie, M. O. (2021). Assessment of water quality using GIS techniques and water quality index in reservoirs affected by water diversion. Water-Energy Nexus, 4, 25-34. https://doi.org/10.1016/j.wen.2020.12.002

Open Source IOT Platform ThingsBoard. (2023). ThingsBoard Documentation. Retrieved October 12, 2023, from https://thingsboard.io/docs/pe/

Chen, Y., & Han, D. (2018). Water quality monitoring in smart city: A pilot project. Water and Environment Management Research Centre, Department of Civil Engineering, University of Bristol, Bristol BS8 1TR, UK.

InfluxData. (2023). InfluxDB 2.0 Documentation. Retrieved October 12, 2023, from https://docs.influxdata.com/influxdb/v2.0/

Keshipeddi, S. B. (2021). "IoT Based Smart Water Quality Monitoring System." 7 Pages Posted: 27 Sep 2021. Kakatiya Institute of Technology & Science, Warangal. [Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3904842]

Hamid, S. A., Rahim, A. M. A., Fadhlullah, S. Y., Abdullah, S., Muhammad, Z., & Leh, N. A. M. (2020). IoT-based Water Quality Monitoring System and Evaluation. 2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 102-106. doi: 10.1109/ICCSCE50387.2020.9204931.

Salleh, A., Hashim, N. M. Z., & Mohamad, N. R. (2022). Realization of IoT Water Monitoring System using NodeMCU ESP8266 Microcontroller and Blynk Application. Published by the Center for Telecommunication Research & Innovation (CeTRI), Faculty of Electronic and Computer Engineering (FKEKK), Universiti Teknikal Malaysia Melaka (UTeM), Durian Tunggal, Melaka, Malaysia.

Zahid Ali. (2019, March 5). Introduction to DHT11. The Engineering Projects. Retrieved from https://www.theengineeringprojects.com/2019/03/introduction-to-dht11.html

Wemos. (2021). Wemos D1 Mini Documentation. Retrieved from https://www.wemos.cc/en/latest/d1/d1_mini.html

OpenAI. (2023). ChatGPT (4.0). Retrieved November 2023, from https://www.openai.com/

APPENDICES

Appendix 1. C++ code for sending sensor data to ThingsBoard Cloud

```
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define DHTPIN D2     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);

const char* THINGSBOARD_HOST = "thingsboard.cloud";
const char* ACCESS_TOKEN = "4HCCTE9p06d1m9U8MBMG";

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(9600);
  dht.begin();
  Serial.println();
  WiFi.begin("WIFI", "SSID");

  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());


  client.setServer(THINGSBOARD_HOST, 1883);
  while (!client.connected()) {
    Serial.println("Connecting to ThingsBoard...");
    if (client.connect("ESP8266 Device", ACCESS_TOKEN, NULL)) {
      Serial.println("Connected to ThingsBoard");
    } else {
      Serial.println("Failed to connect to ThingsBoard, retrying...");
      delay(2000);
    }
  }
}
```

```
void loop() {
  delay(2000); // Wait for 2 seconds between readings

  float temperature = dht.readTemperature(); // Read temperature in Cel-
sius
  float humidity = dht.readHumidity(); // Read humidity as a percentage

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C, Humidity: ");
  Serial.print(humidity);
  Serial.println("%");
  // Create a JSON payload with the temperature and humidity data
  String payload = "{\"temperature\":" + String(temperature) + ",\"humid-
ity\":" + String(humidity) + "}";

  // Publish the payload to the ThingsBoard MQTT broker
  client.publish("v1/devices/me/telemetry", (char*) payload.c_str());

  // Wait for the publish operation to complete
  client.loop();
}
```

Appendix 2. C++ code for sending sensor data to InfluxDB cloud server

```
#include <ESP8266WiFi.h>
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>
#include <DHT.h>

#define INFLUXDB_URL "https://eu-central-1-1.aws.cloud2.influxdata.com"
#define INFLUXDB_TOKEN "Token"
#define INFLUXDB_ORG "c921803260fff"
#define INFLUXDB_BUCKET "AirQulity" //Bucket name

#define DHTPIN D2
#define DHTTYPE DHT11

#define TZ_INFO "UTC3"

InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, IN-
FLUXDB_TOKEN, InfluxDbCloud2CACert);
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
  WiFi.begin("SSID", "PASSWORD");
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");

  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C, Humidity: ");
  Serial.print(humidity);
  Serial.println("%");

  // Create a data point for the sensor readings
  Point dataPoint("dht11");

  // Add tags and fields to the data point
  dataPoint.addTag("location", "living_room");
  dataPoint.addTag("sensor_type", "DHT11");
  dataPoint.addField("temperature", temperature);
  dataPoint.addField("humidity", humidity);

  // Write the data point to the InfluxDB server
  if (client.writePoint(dataPoint)) {
    Serial.println("Data written to InfluxDB.");
  } else {
    Serial.print("InfluxDB write failed: ");
    Serial.println(client.getLastErrorMessage());
    // Store the data point in a buffer for offline sending
    client.storePoint(dataPoint);
  }

  // Check if there are any stored data points to send
  if (client.hasStoredPoints()) {
    // Send the stored data points to the InfluxDB server
    if (client.sendStoredPoints()) {
      Serial.println("Stored data points sent to InfluxDB.");
    } else {
```

```
      Serial.print("Failed to send stored data points: ");
      Serial.println(client.getLastErrorMessage());
    }
  }

  delay(10000);
}
```

Appendix 3. Software Setup

To enable seamless communication between the hardware components and facilitate efficient data processing, the following software tools were utilized:

- Arduino IDE: Install the ESP8266 board support in Arduino IDE, select the appropriate board (Lolin D1 Mini), and ensure the correct port is chosen for programming.
- ThingsBoard Cloud: Set up an account on ThingsBoard Cloud, create a new device corresponding to the ESP8266, and configure telemetry settings to enable the transmission of sensor data to ThingsBoard.
- InfluxDB Cloud: Set up an InfluxDB Cloud account, configure a bucket to store the data, and establish a connection with the ESP8266 to store the sensor readings.
- Grafana Cloud: Create an account on Grafana Cloud, connect it to the InfluxDB Cloud as a data source, and design dashboards showcasing real-time and historical sensor data.

Appendix 4. Flux Query Samples used in making Grafana dashboard.

Calculate the rate of change of temperature over time:

```
from(bucket: "AirQulity")
  |> range(start: -12h)
  |> filter(fn: (r) => r["_measurement"] == "dht11" and r["_field"] == "temperature")
  |> derivative(unit: 1s, nonNegative: true, columns: ["_value"])
```

Fetches the most recent humidity reading from the "dht11" measurement within the last 5 minutes:

```
from(bucket: "AirQulity")
  |> range(start: -5m)
  |> filter(fn: (r) => r["_measurement"] == "dht11" and r["_field"] == "humidity")
  |> last()
```