VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY
FACULTY OF COMPUTER ENGINEERING AND ELECTRONICS

GRADUATION PROJECT

AIOT-BASED WEATHER FORECASTING SYSTEM USING MACHINE LEARNING

Student's Full Name : *Thai Van Hoa*

Class : *21IR*

Major : *Information Technology*

Specialization : *IoT & Robotics*

Instructor : *Ph.D. Vuong Cong Dat*

Da Nang - 12/2025

VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY
FACULTY OF COMPUTER ENGINEERING AND ELECTRONICS

GRADUATION PROJECT

AIOT-BASED WEATHER FORECASTING SYSTEM USING MACHINE LEARNING

Student's Full Name : *Thai Van Hoa*

Class : *21IR*

Major : *Information Technology*

Specialization : *IoT & Robotics*

Instructor : *Ph.D. Vuong Cong Dat*

Da Nang - 12/2025

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to the Faculty of Computer Engineering and Electronics at the VietNam - Korean University Of Information And Communication Technology for providing favorable conditions and supporting me throughout my four years of study and research. It is thanks to the care and facilitation of the university that I have had the best learning environment to develop my knowledge and skills.

I would like to sincerely thank Ph.D. Vuong Cong Dat for his dedicated guidance and valuable feedback, which have greatly helped me complete this thesis. He has devoted significant time and effort to assist me in overcoming the challenges and difficulties during the research process, as well as imparting invaluable knowledge and practical experience.

At the same time, I would also like to extend my heartfelt thanks to my family and friends—those who have always been by my side, sharing, encouraging, and supporting me throughout the process of studying and completing this thesis. The support and encouragement from my family and friends have given me the motivation and confidence to successfully accomplish my tasks.

Inevitably, there are shortcomings and mistakes in this thesis. I sincerely hope to receive feedback from esteemed lecturers so that I can improve in the future. I also look forward to continuing to receive guidance and advice from you in my future research endeavors.

Thank you very much !

Student

Thai Van Hoa

SUPERVISOR'S EVALUATION

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Da Nang,December 2025

Supervisor

Ph.D. Vuong Cong Dat

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i
SUPERVISOR'S EVALUATION ii
TABLE OF CONTENTS iii
LIST OF TABLES vi
LIST OF FIGURES vii
INTRODUCTION 1
CHAPTER I: OVERVIEW 3
1.1. BACKGROUND AND MOTIVATION 3
1.1.1. Current context and practical needs 3
1.1.2. Motivation for combining IoT and Machine Learning 3
1.2. PROBLEM STATEMENT 3
1.2.1. Limitations of existing solutions 4
1.2.2. Key challenges addressed by this project 4
1.3. OBJECTIVES 4
1.3.1. System and data objectives 4
1.3.2. Orecasting and application objectives 4
1.4. METHODOLOGY 5
1.4.1. Proposed methodology 5
CHAPTER II: THEORETICAL BASIS 6
2.1. INTERNET OF THINGS (IOT) 6
2.1.1. Introduction to IoT 6
2.1.2. IoT Architecture (4 layers) 6
2.1.3. IoT in environmental monitoring 7
2.2. COMMUNICATION PROTOCOLS 8
2.2.1. MQTT Protocol 8
2.2.2. TCP/IP Protocol 8
2.3. LORA TECHNOLOGY 9
2.3.1. LoRa Introduction 9
2.3.2. Main Specifications 10
2.3.3. Module E32-433T20D 11

2.3.4. Comparison of LoRa with Wi-Fi/4G 12
2.4. MACHINE LEARNING FOR WEATHER FORECASTING 13
2.4.1. ML Overview for Forecasting 13
2.4.2. Time Series Forecasting 14
2.4.3. Prophet Model 14
2.4.4. LightGBM Model 16
2.5. WEB TECHNOLOGIES AND DATA PROCESSING 16
2.5.1. Node-RED 17
2.5.2. MySQL Database 17
2.5.3. FastAPI Framework 18
2.6. THE HARDWARE 19
2.6.1. Raspberry Pi 4B 19
2.6.2. Esp32 20
2.6.3. MQ-135 21
2.6.4. DHT11 Temperature and Humidity Sensor 22
2.6.5. BMP280 + ATH20 22
2.6.6. OLED 0.96 23
2.6.7. GP2Y1010AU0F 23
2.6.8. Lora E32 433T20D 24
CHAPTER III: SYSTEM ANALYSIS AND DESIGN 25
3.1. SYSTEM OVERVIEW 25
3.2. OVERALL SYSTEM ARCHITECTURE 26
3.2.1. System Block Diagram 27
3.2.2. Algorithm Flowcharts 28
3.3. HARDWARE DESIGN 28
3.3.1. Schematic Design 28
3.3.2. PCB Design 29
3.3.3. Finished Products 30
3.4. SOFTWARE DESIGN 31
3.4.1. Data Ingestion and Processing Layer (MQTT/Node-RED) 31
3.4.2. FastAPI Backend 32

3.4.3. Web Frontend 32
CHAPTER IV: RESULTS AND EVALUATION.....34
4.1. SYSTEM IMPLEMENTATION RESULTS.....34
4.2. FORECASTING EVALUATION 34
4.2.1. Evaluation methodology and metrics 34
4.2.2. Comparison using Forecast vs Actual plots 35
4.2.3. Feature importance 36
4.3. OVERALL ASSESSMENT.....36
CONCLUSION AND DEVELOPMENT DIRECTION.....37
REFERENCES 40

LIST OF TABLES

Table 2.1. LoRa Comparison Table.....12

LIST OF FIGURES

Figure 2.1. Internet of Things **6**

Figure 2.2. 4 layers of IOT **7**

Figure 2.3. MQTT Protocol **8**

Figure 2.4. TCP/IP **9**

Figure 2.5. LoRa & LoRaWAN **10**

Figure 2.6. E32-433T20D **11**

Figure 2.7. Prophet model **15**

Figure 2.8. Node RED **17**

Figure 2.9. MySQL Database **18**

Figure 2.10. FastAPI **18**

Figure 2.11. Raspberry Pi 4B **19**

Figure 2.12. Esp32 Wroom Pin Diagram **20**

Figure 2.13. MQ-135 ..... **21**

Figure 2.14. DHT11 Sensor **22**

Figure 2.15. BMP280+ATH20 ..... **23**

Figure 2.16. 0.96-inch OLED **23**

Figure 2.17. GP2Y1010AU0F **24**

Figure 2.18. LoRa E32-433T20D **24**

Figure 3.1. System Overview **25**

Figure 3.2. Overall System Architecture **26**

Figure 3.3. System Block Diagram **27**

Figure 3.4. Algorithm Flowcharts **28**

Figure 3.5. Schematic **29**

Figure 3.6. 3D PCB **29**

Figure 3.8. Node-RED flow **31**

Figure 3.9. API UI 32

Figure 3.10. Dashboard.....33

Figure 4.1. Interface Forecasting 34

Figure 4.2. Train Model Data File.....35

Figure 4.3. Compare Models 36

INTRODUCTION

1. Reason for Choosing the Topic

In recent years, weather and environmental conditions have become increasingly unpredictable due to climate change and rapid urbanization. Extreme heat, sudden rainfall, large humidity fluctuations, and deteriorating air quality have directly affected public health, transportation safety, and daily activities. For areas such as schools, public parks, and densely populated neighborhoods, continuous monitoring of environmental parameters and early weather forecasting is essential to reduce risks and support proactive decision-making.

However, in many locations, people often rely only on generalized online weather information, which may not accurately represent the microclimate at a specific site (e.g., inside a school campus or a public park). In addition, traditional monitoring methods if available are often manual, discontinuous, and lack long-term data storage, making it difficult to analyze trends, detect anomalies early, or provide timely warnings.

With the advancement of the Internet of Things (IoT), low-cost sensor nodes such as ESP32 enable continuous measurement of key parameters including temperature, humidity, pressure, CO_2, dust/PM2.5, and air-quality indicators. Nevertheless, monitoring alone is not sufficient; users increasingly need predictive capabilities to anticipate upcoming conditions (e.g., rainfall likelihood, temperature/humidity trends, or air-quality risks) for planning and prevention.

Meanwhile, Machine Learning has demonstrated strong performance in time-series forecasting by learning trends and seasonal patterns from historical data. When IoT sensor data is continuously collected, stored in a database (e.g., MySQL), and integrated with Weather API data, it becomes feasible to build a complete AIoT pipeline that supports both real-time monitoring and short-term forecasting with measurable accuracy.

For these reasons, I chose the graduation project **"AIoT-Based Weather Forecasting System Using Machine Learning"** to develop an integrated system for data acquisition, storage and management, model training for forecasting, and practical visualization for end users.

2. Research purpose

The purpose of this project is to develop an AIoT-based environmental monitoring and weather forecasting system that improves upon conventional approaches by

combining real-time sensing, gateway processing, database management, and machine learning forecasting.

Specifically, the project aims to:

- **Build a real-time monitoring system** using ESP32 and environmental sensors to measure parameters such as temperature, humidity, pressure, CO_2, and dust/PM2.5 for a specific site.
- **Integrate Weather API data** (e.g., wind speed, wind direction, rainfall, UV index) to enrich the dataset and support improved forecasting performance.
- **Design a complete data pipeline** where sensor and API data are transmitted to a Raspberry Pi gateway, processed via Node-RED, and stored in a **MySQL database** for long-term analysis.
- **Develop and deploy machine learning time-series forecasting models**, trained on the collected data to produce short-term forecasts (e.g., 24 hours and 7 days), supporting early awareness of rain occurrence and environmental changes.
- **Provide a user-friendly dashboard/web interface** to display real-time readings, historical charts, forecasting results, and to support data management operations (filtering, exporting, deleting old records) for efficient system maintenance.

In addition, the project places strong emphasis on **system robustness, scalability, and long-term usability** in practical AIoT deployments. The overall architecture is designed in a modular and layered manner, enabling the system to be easily extended with additional sensor nodes, new environmental parameters, or alternative communication technologies without requiring major structural changes. This flexibility allows the solution to be adapted to different deployment scenarios such as schools, public parks, urban residential areas, or small industrial zones.

Furthermore, by adopting an **edge–cloud collaborative approach**, the Raspberry Pi gateway performs local data aggregation, preprocessing, and temporary decision support before storing data in the database and forwarding results to the application layer. This design reduces network dependency, minimizes latency, and improves system resilience in cases of intermittent internet connectivity. The use of open-source platforms such as Node-RED, MySQL, and Python-based machine learning frameworks also enhances transparency, maintainability, and cost efficiency.

CHAPTER I: OVERVIEW

1.1. BACKGROUND AND MOTIVATION

1.1.1. Current context and practical needs

In recent years, climate change and rapid urbanization have made weather and environmental conditions more volatile and less predictable. Extreme heat, sudden heavy rainfall, abrupt humidity changes, and increasing fine-dust pollution negatively impact public health as well as daily activities. In sites such as **schools**, **public parks**, and densely populated areas, real-time monitoring of environmental conditions is increasingly necessary to support timely warnings and practical decisions (e.g., outdoor activity scheduling, limiting exposure to fine dust, or preparing for rain).

In practice, users often rely on general forecasting apps or websites. However, such sources usually provide region-level information and may not accurately represent the **local microclimate** at a specific site. Even within the same city, temperature, humidity, and air quality can vary depending on traffic density, vegetation, and terrain. Therefore, an on-site monitoring system can provide more relevant and accurate information.

1.1.2. Motivation for combining IoT and Machine Learning

The advancement of the **Internet of Things (IoT)** enables the deployment of low-cost sensor nodes such as **ESP32** for continuous, automated, and scalable data acquisition. Key parameters for environmental assessment and forecasting include **temperature**, **humidity**, **pressure**, **CO_2**, **dust/PM2.5**, and related indicators. With a gateway such as **Raspberry Pi** and data-flow processing via **Node-RED**, the collected data can be normalized and stored centrally in **MySQL** for long-term querying and analytics.

Nevertheless, real-time monitoring only reflects the "current state," while practical users often require **forecasting** to take proactive actions: anticipating temperature/humidity changes, rainfall likelihood, or air quality trends. This is where Machine Learning becomes crucial. With sufficient historical data, ML models can learn trends, seasonal patterns, and relationships among variables to produce short-term forecasts such as **24-hour** and **7-day** predictions.

Moreover, fusing on-site sensor data with **Weather API** information (wind speed, wind direction, rainfall, UV index, etc.) enriches the dataset and can improve forecasting stability and accuracy. Therefore, this project adopts an **AIoT (AI + IoT)** approach to build a practical system that supports both monitoring and forecasting.

1.2. PROBLEM STATEMENT

1.2.1. Limitations of existing solutions

Common limitations in current weather/environment monitoring include:

- Lack of on-site measurements: users mostly rely on generalized forecasts rather than local conditions.
- Manual and discontinuous monitoring: sparse data makes long-term trend analysis difficult.
- Poor data management: data is often fragmented without centralized storage for efficient queries and reporting.
- Limited forecasting capability: many systems only display real-time sensor readings without predictive models.

1.2.2. Key challenges addressed by this project

Based on the above limitations, this project addresses key challenges:

- Building a robust architecture for stable data acquisition from multiple sources (sensors + API).
- Designing a MySQL database schema optimized for historical queries and ML training.
- Implementing preprocessing steps (unit normalization, missing-value handling, basic noise/outlier removal).
- Selecting and deploying suitable Machine Learning models for time-series forecasting.
- Presenting forecasts clearly in the user interface for practical interpretation and use.

1.3. OBJECTIVES

1.3.1. System and data objectives

- Build an ESP32-based sensing system to measure environmental parameters: temperature, humidity, pressure, CO_2, and dust/PM2.5.
- Collect additional Weather API data: wind speed, wind direction, rainfall, and UV index.
- Deploy Raspberry Pi + Node-RED as a gateway to process data streams and store data in MySQL, building structured historical datasets for analytics and training.

1.3.2. Orecasting and application objectives

- Develop ML time-series forecasting models for key indicators (24-hour and 7-day horizons).

- Implement rainfall occurrence prediction based on historical and related variables.
- Design a web interface for real-time monitoring, historical charts, forecasts, and warning status.
- Provide data administration functions (time filtering, export, deleting old records) to support long-term stability.

1.4. METHODOLOGY

1.4.1. Proposed methodology

The methodology of the proposed system is structured as a complete and continuous data-processing workflow, starting from data acquisition and ending with visualization and decision support. First, environmental data are collected in real time from ESP32-based sensor nodes, including parameters such as temperature, humidity, pressure, CO_2 concentration, and dust/PM2.5 levels. In parallel, external meteorological information is retrieved from a Weather API, providi

Next, all incoming data streams are transmitted to the Raspberry Pi gateway, where Node-RED is used as the central processing and orchestration platform. Within Node-RED, the raw sensor and API data are filtered, normalized, time-aligned, and packaged into a consistent format suitable for storage and machine learning. This step ens

After preprocessing, the structured data are stored in a MySQL database, enabling the construction of long-term historical time-series datasets. The database design supports efficient querying, aggregation, and maintenance operations, serving both real-time monitoring and offline analysis purposes. Based on these historical records, machine learning forecasting models are trained to learn temporal patterns and seasonal trends in the environmental data. The performance of the forecasting models is then evaluated using standard regression metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to assess prediction accuracy and stability.

Finally, the generated forecast results are saved into dedicated result tables within the database and made available to the application layer. A web-based dashboard is developed to visualize real-time sensor readings, historical trends, and short-term forecast outputs (e.g., 24-hour and 7-day predictions). This interface allows users to intuitively monitor environmental conditions, analyze past data, and utilize forecasting information for timely awareness and informed decision-making.

CHAPTER II: THEORETICAL BASIS

2.1. INTERNET OF THINGS (IOT)

2.1.1. Introduction to IoT

The Internet of Things (IoT) represents a paradigm shift in how physical devices interact with digital systems. IoT encompasses a network of physical objects embedded with sensors, software, and connectivity capabilities that enable them to collect, exchange, and act upon data without human intervention. According to recent estimates, the number of connected IoT devices worldwide has surpassed 15 billion and continues to grow exponentially.

In the context of environmental monitoring, IoT technology provides unprecedented opportunities for real-time data collection, analysis, and decision-making. The fundamental concept involves deploying sensor nodes across geographical areas to continuously monitor environmental parameters such as temperature, humidity, atmospheric pressure, and air quality.

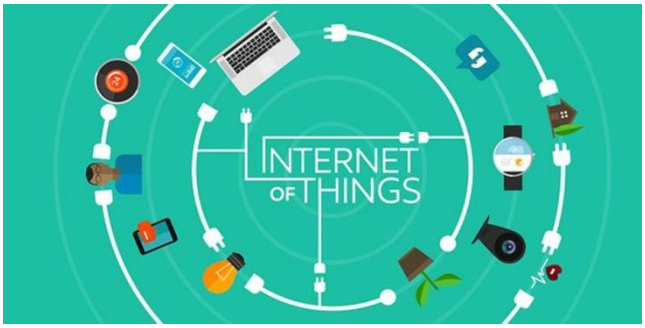

Figure 2.1. Internet of Things

2.1.2. IoT Architecture (4 layers)

A comprehensive IoT system is typically structured in four hierarchical layers, each serving distinct functions:

1. Perception Layer (Sensing Layer) This foundational layer consists of physical sensors and actuators that interact directly with the environment. In weather monitoring systems, this includes temperature sensors, humidity sensors, barometric pressure sensors, and other environmental monitoring devices. The perception layer is responsible for data acquisition and preliminary signal processing.

2. Network Layer (Transmission Layer) The network layer handles data transmission from sensing devices to processing systems. This layer employs various communication protocols and technologies such as LoRa, Wi-Fi, Zigbee, or cellular

networks. The choice of technology depends on factors including transmission range, power consumption, data rate requirements, and deployment environment.

3. Processing Layer (Middleware Layer) This layer manages data storage, processing, and analysis. It typically includes cloud servers, databases, and computational resources that handle large volumes of sensor data. The processing layer performs functions such as data aggregation, filtering, analytics, and machine learning model execution.

4. Application Layer The topmost layer provides user interfaces and application-specific services. This includes web dashboards, mobile applications, alert systems, and APIs that enable end-users to access processed information and insights derived from sensor data.

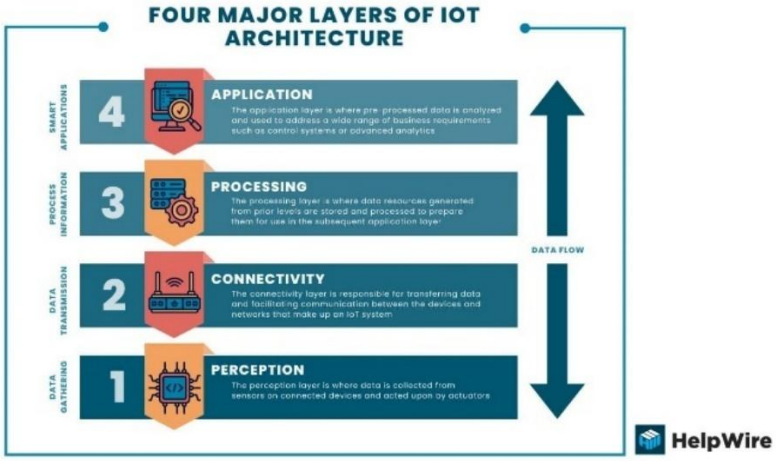

Figure 2.2. 4 layers of IOT

2.1.3. IoT in environmental monitoring

IoT technology has revolutionized environmental monitoring by enabling continuous, automated, and large-scale data collection. Traditional weather stations were limited by high costs, manual data collection requirements, and sparse geographical coverage. IoT-based monitoring systems overcome these limitations through:

- Distributed Sensing: Deployment of multiple low-cost sensor nodes across wide areas provides granular spatial resolution
- Real-time Monitoring: Continuous data streaming enables immediate detection of environmental changes
- Cost Efficiency: Modern sensor technology and wireless communication reduce infrastructure costs significantly

- Scalability: Systems can easily expand by adding more sensor nodes without major infrastructure changes
- Data Integration: IoT platforms facilitate integration of multiple data sources for comprehensive environmental analysis

Applications in environmental monitoring include weather forecasting, air quality monitoring, agricultural management, disaster early warning systems, and climate research.

2.2. COMMUNICATION PROTOCOLS

2.2.1. MQTT Protocol

MQTT (Message Queuing Telemetry Transport) is a message transmission protocol based on the Publish/Subscribe communication model, suitable for remote data transport. It is a very lightweight protocol therefore used to communicate devices (M2M Machine to Machine), WSN (Wireless Sensor Networks) and is most popular in IoT projects. This protocol is designed to exchange data between the server and the client. Also with its compact, simple size, low energy usage, optimized and easy-to-implement data plans make it even more ideal.

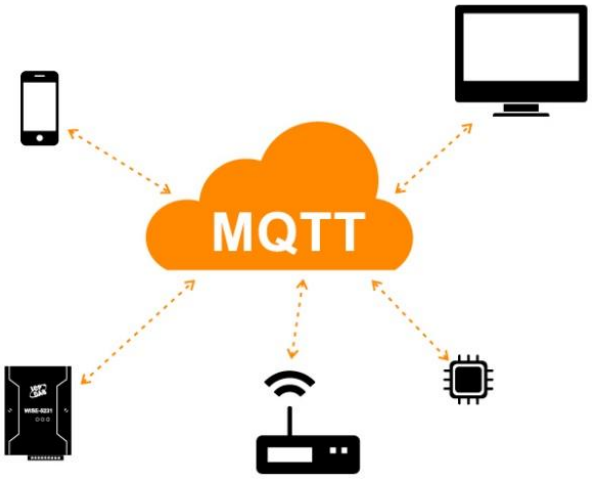

Figure 2.3. MQTT Protocol

2.2.2. TCP/IP Protocol

Transmission Control Protocol/Internet Protocol (TCP/IP) forms the foundation of Internet communication and provides the underlying transport mechanism for MQTT and other application-layer protocols.

TCP/IP Protocol Suite Layers:

1. **Application Layer**: Hosts protocols like HTTP, MQTT, FTP that applications use directly

2. **Transport Layer**: TCP provides reliable, connection-oriented data delivery with error checking and flow control

3. **Network Layer**: IP handles routing and addressing, enabling data packets to reach their destination across networks

4. **Link Layer**: Manages physical network connections and hardware addressing

MQTT over TCP/IP

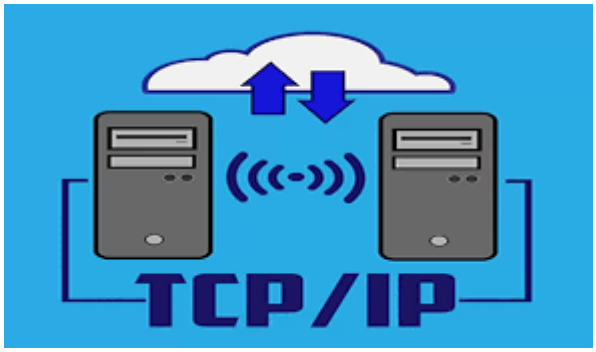

Figure 2.4. TCP/IP

MQTT operates at the application layer and relies on TCP for reliable data transmission. The typical communication flow involves:

1. Client establishes TCP connection with MQTT broker (default port 1883 or 8883 for secure connections)

2. Client sends MQTT CONNECT packet to authenticate and initialize the session

3. Broker responds with CONNACK packet confirming connection

4. Client publishes or subscribes to topics using MQTT packets transmitted over the TCP connection

5. Broker distributes messages according to subscriptions

TCP's reliability mechanisms (acknowledgments, retransmission, ordering) complement MQTT's QoS levels, ensuring dependable data delivery in IoT systems. For resource-constrained devices or networks with high packet loss, alternatives like MQTT-SN (over UDP) may be considered.

2.3. LORA TECHNOLOGY

2.3.1. LoRa Introduction

Long Range (LoRa) is a wireless modulation technique designed for long-distance, low-power communication in IoT applications. Developed by Semtech Corporation,

LoRa uses Chirp Spread Spectrum (CSS) modulation to achieve exceptional range and penetration characteristics while maintaining minimal power consumption.

LoRa operates in unlicensed ISM (Industrial, Scientific, and Medical) frequency bands, which vary by region: 868 MHz in Europe, 915 MHz in North America, and 433 MHz in Asia. This project utilizes the 433 MHz band, which offers excellent range characteristics and building penetration in the Vietnamese deployment environment.

LoRaWAN Quick Overview

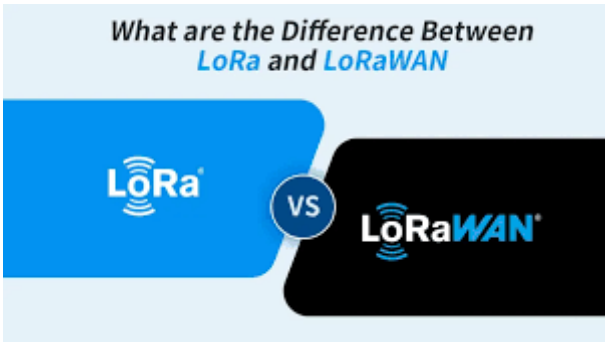

Figure 2.5. LoRa & LoRaWAN

LoRaWAN (Long Range Wide Area Network) is a network protocol built on top of LoRa physical layer technology. While this project uses point-to-point LoRa communication, LoRaWAN provides additional features for large-scale deployments:

- Network architecture with end-devices, gateways, network servers, and application servers
- Device classes (A, B, C) for different power/latency trade-offs
- Built-in security with AES-128 encryption
- Adaptive data rate (ADR) for optimization

2.3.2. Main Specifications

Long Range Capability LoRa technology achieves transmission ranges of 2-5 km in urban environments and up to 15-20 km in rural areas with line-of-sight. This extended range results from CSS modulation, which spreads the signal across a wide frequency band, making it resilient to interference and enabling the receiver to decode signals below the noise floor.

Data Rate and Trade-offs LoRa data rates typically range from 0.3 kbps to 50 kbps depending on configuration. The relatively low data rate is acceptable for IoT applications transmitting small data packets periodically (e.g., sensor readings every few minutes).

Spreading Factor (SF) Spreading Factor determines how much the signal is spread across the frequency band, ranging from SF7 (fastest, shortest range) to SF12 (slowest, longest range). Higher spreading factors improve sensitivity and range but reduce data rate and increase transmission time. For weather monitoring, SF9 or SF10 typically provides optimal balance.

Bandwidth (BW) LoRa supports bandwidths of 125 kHz, 250 kHz, and 500 kHz. Narrower bandwidths improve receiver sensitivity and range but reduce data rate. The 125 kHz bandwidth is most common for long-range applications.

Link Budget LoRa modems can achieve link budgets exceeding 155 dB, enabling reception of extremely weak signals. This is the key enabler of long-range communication with low transmission power.

2.3.3. Module E32-433T20D

The E32-433T20D is a LoRa transceiver module manufactured by Ebyte, featuring the Semtech SX1278 LoRa chip. This module is particularly suitable for weather monitoring applications due to its robust specifications and low power consumption.

Figure 2.6. E32-433T20D

Key Specifications:

- Frequency: 410-441 MHz (433 MHz ISM band)
- Transmission Power: 20 dBm (100 mW) maximum
- Receiver Sensitivity: -136 dBm at SF12/125kHz
- Communication Distance: Up to 3 km in urban areas, 8 km in open areas
- Interface: UART (TTL level)
- Supply Voltage: 2.3V - 5.2V

- Sleep Current: < 2 μA

Operating Modes: The module supports multiple operating modes to optimize power consumption:

1. **Normal Mode**: Full functionality for transmitting and receiving
2. **Wake-Up Mode**: Periodically checks for incoming messages while conserving power
3. **Power-Saving Mode**: Ultra-low power consumption with configurable wake-up periods
4. **Sleep Mode**: Minimal power consumption with wake-up on external trigger

Battery Operation Strategy For battery-powered sensor nodes, the module operates primarily in power-saving mode, waking periodically to collect sensor data and transmit to the gateway. With optimized duty cycling (e.g., 1-minute transmission intervals with sleep between), a node can operate for 1-2 years on battery power.

Role in the System:

- **Sensor Node**: E32 module transmits sensor data to the gateway
- **Gateway**: E32 module receives data from multiple sensor nodes and forwards to the server

2.3.4. Comparison of LoRa with Wi-Fi/4G

Different wireless technologies suit different IoT application requirements. Here's a comparative analysis:

Table 2.1. LoRa Comparison Table

Feature	LoRa	Wi-Fi	4G/LTE
Range	2-15 km	50-100 m	2-10 km
Data Rate	0.3-50 kbps	150+ Mbps	1-100+ Mbps
Power Consumption	Very Low	Medium-High	High
Battery Life	Years	Days-Weeks	Days
Infrastructure Cost	Low	Medium	High (subscription)

Feature	LoRa	Wi-Fi	4G/LTE
Deployment	Private network	Requires router	Network coverage
License	Unlicensed ISM	Unlicensed	Licensed spectrum
Best For	Long-range sensors	High-bandwidth local	Mobile, high-speed

Why LoRa for Weather Monitoring:

- Extended range enables sparse gateway deployment covering large areas
- Ultra-low power consumption allows long-term battery operation in remote locations
- Low infrastructure cost for private network deployment
- Sufficient data rate for periodic transmission of small sensor data packets
- No recurring subscription fees unlike cellular networks
- Excellent penetration through buildings and vegetation

2.4. MACHINE LEARNING FOR WEATHER FORECASTING

2.4.1. ML Overview for Forecasting

Machine Learning (ML) has emerged as a powerful approach for weather forecasting, complementing traditional numerical weather prediction models. ML algorithms excel at identifying complex patterns in historical data and generating predictions based on learned relationships.

Advantages of ML in Weather Forecasting:

- **Pattern Recognition**: ML models can identify non-linear relationships between meteorological variables
- **Scalability**: Once trained, models generate predictions rapidly without intensive computation
- **Adaptability**: Models can be continuously updated with new data to improve accuracy
- **Local Optimization**: Models can be trained on local data to capture region-specific patterns
- **Multiple Variables**: ML handles multivariate time series effectively

ML Workflow for Weather Prediction:

1. Data Collection: Gather historical weather observations
2. Data Preprocessing: Clean, normalize, and engineer features
3. Model Training: Train algorithms on historical patterns
4. Validation: Evaluate model performance on test data
5. Deployment: Generate real-time forecasts from current conditions
6. Monitoring: Continuously assess and improve model accuracy

2.4.2. Time Series Forecasting

Time series forecasting involves predicting future values based on previously observed data points ordered chronologically. Weather data is inherently a time series with distinct characteristics that influence modeling approaches.

Time Series Characteristics:

1. Trend Long-term increase or decrease in the data. For example, gradual temperature changes across seasons or climate trends over years.

2. Seasonality Regular patterns that repeat at fixed intervals. Daily temperature cycles (warm days, cool nights) and seasonal variations (summer heat, winter cold) are classic examples in weather data.

3. Cyclicity Patterns that repeat at irregular intervals, such as multi-year climate oscillations (El Niño/La Niña effects).

4. Autocorrelation Correlation of the time series with lagged versions of itself. Tomorrow's temperature is highly correlated with today's temperature.

5. Stationarity Statistical properties (mean, variance) remain constant over time. Many ML models assume or require stationary data, though weather data often exhibits non-stationarity due to trends and seasonality.

2.4.3. Prophet Model

Prophet is an open-source forecasting library developed by Facebook's Core Data Science team, designed for forecasting time series data with strong seasonal patterns and multiple seasons of historical data.

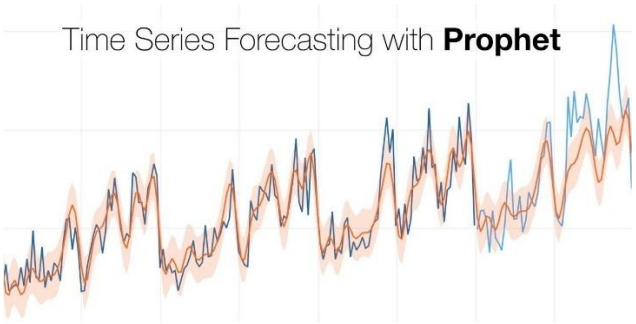

Figure 2.7. Prophet model

Principle of Operation

Prophet decomposes time series into three main components using an additive model:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

Where:

- **g(t)**: Trend function modeling non-periodic changes (piecewise linear or logistic growth)
- **s(t)**: Seasonal component representing periodic changes (Fourier series)
- **h(t)**: Holiday effects and irregular events
- ε_t: Error term representing idiosyncratic changes

Key Features:

Automatic Seasonality Detection Prophet automatically detects daily, weekly, and yearly seasonal patterns using Fourier series. For weather data, this captures diurnal temperature cycles and annual seasonal variations.

Trend Changepoints The model automatically identifies points where the time series trend changes, allowing for flexible trend modeling without manual intervention.

Robust to Missing Data Prophet handles missing values and outliers gracefully, making it suitable for real-world sensor data that may have gaps.

Interpretable Parameters Model components (trend, seasonality, holidays) are interpretable, enabling understanding of what drives predictions.

Pros and Cons

Advantages:

- Intuitive parameters requiring minimal tuning

- Excellent handling of seasonal patterns and trends
- Robust to missing data and outliers
- Fast training and prediction
- Uncertainty intervals included in forecasts
- Works well with relatively small datasets
- Interpretable results aid in understanding predictions

Disadvantages:

- Limited ability to capture complex non-linear relationships
- Assumes additive structure may not fit all data patterns
- Less effective for short-term (hourly) forecasts with high volatility
- Cannot automatically incorporate external variables (requires manual feature engineering)
- May struggle with data having irregular patterns or multiple overlapping cycles

2.4.4. LightGBM Model

LightGBM (Light Gradient Boosting Machine) is a gradient boosting framework developed by Microsoft that uses tree-based learning algorithms. It is highly efficient and has become popular for both classification and regression tasks, including time series forecasting.

Gradient Boosting Principle

Gradient Boosting builds an ensemble of decision trees sequentially, where each new tree corrects errors made by the existing ensemble.

Mathematical Formulation:

$$\mathbf{F(x) = F_0(x) + \Sigma\, \gamma_m\, h_m(x)}$$

Where:

- $F(x)$ is the final prediction
- $F_0(x)$ is the initial prediction (baseline)
- $h_m(x)$ is the m-th decision tree
- γ_m is the learning rate (shrinkage parameter)

2.5. WEB TECHNOLOGIES AND DATA PROCESSING

2.5.1. Node-RED

Node-RED is a flow-based development tool originally developed by IBM for wiring together hardware devices, APIs, and online services. It provides a browser-based visual editor that makes it easy to create data processing workflows using a wide range of pre-built nodes.

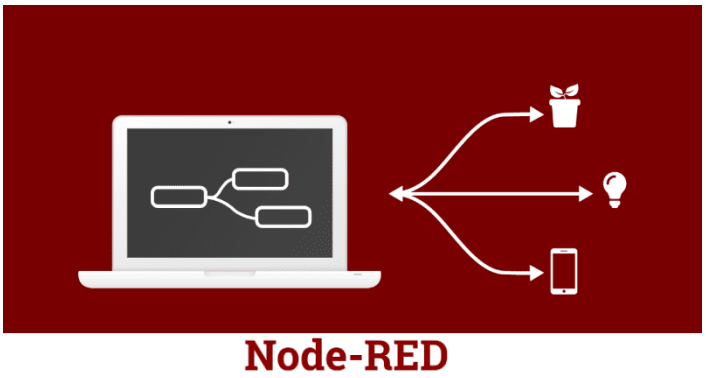

Figure 2.8. Node RED

Advantages:

- Visual programming reduces development time
- Extensive library of pre-built nodes for common tasks
- Real-time data processing with low latency
- Easy integration of multiple protocols and services
- Built-in debugging and monitoring tools

2.5.2. MySQL Database

MariaDB and MySQL are two popular open-source Relational Database Management Systems (RDBMS) that are based on the Structured Query Language (SQL) for managing tabular data. MariaDB is a fork from MySQL (due to ownership concerns after Oracle bought MySQL), so they are highly compatible and easy to migrate between the two parties. By 2025, MariaDB is generally faster at some tasks (bulk inserts, replication, analytic queries), supports more storage engines and powerful community features, while MySQL has better native JSON support and enterprise from Oracle.

Figure 2.9. MySQL Database

2.5.3. FastAPI Framework

FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.7+ based on standard Python type hints. It serves as the backend API layer connecting the database to web/mobile clients and integrating machine learning models.

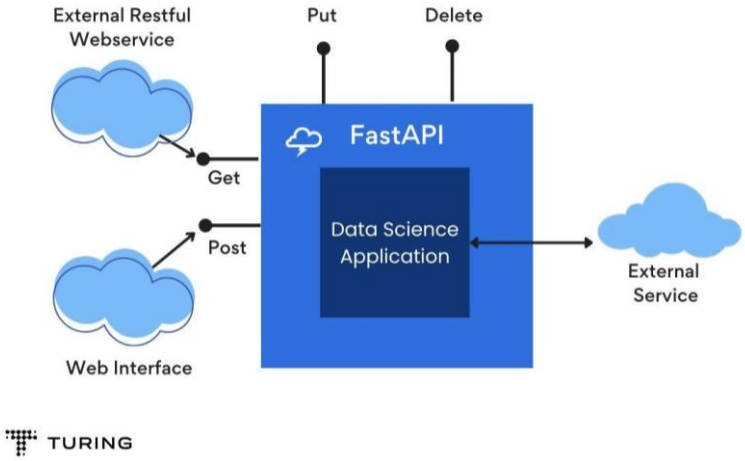

Figure 2.10. FastAPI

Key Features:

1. High Performance FastAPI is one of the fastest Python frameworks, comparable to NodeJS and Go, built on Starlette for web parts and Pydantic for data validation.

2. Automatic API Documentation FastAPI automatically generates interactive API documentation (Swagger UI and ReDoc) based on OpenAPI standards, making API testing and integration straightforward.

3. Type Safety Python type hints enable automatic request validation, serialization, and documentation, reducing bugs and development time.

4. Async Support Native support for asynchronous request handling enables high concurrency for I/O-bound operations like database queries.

API Endpoints in the Project:

GET /api/sensors Returns list of all registered sensor nodes with their metadata and status.

GET /api/sensors/{sensor_id}/data Retrieves historical sensor readings for a specific sensor, with optional time range filtering.

GET /api/sensors/{sensor_id}/forecast Returns weather forecasts for a specific sensor location, including predictions from both Prophet and LightGBM models.

POST /api/forecast/generate Triggers forecast generation for specified sensors, running ML models on latest data.

GET /api/statistics Provides aggregated statistics (min, max, average) for specified time periods.

2.6. THE HARDWARE

2.6.1. Raspberry Pi 4B

The Raspberry Pi 4B is a small-sized computer that runs on the Linux operating system. This compact computer is primarily used to execute large programs to achieve fast output signals.

The Raspberry Pi 4B features a quad-core processor and comes in three different versions with varying RAM capacities. The Pi 4 uses mini HDMI and supports dual ports for two 4K displays.

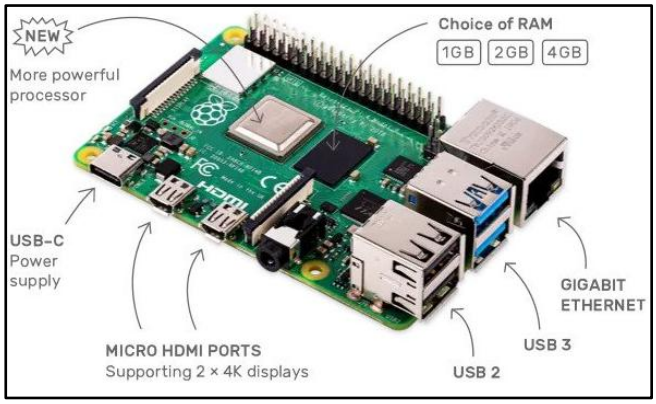

Figure 2.11. Raspberry Pi 4B

Processor: Quad-core Cortex-A72 @ 1.5GHz

RAM: 2GB/4GB/8GB LPDDR4

Wi-Fi: 2.4GHz/5GHz 802.11ac

Bluetooth: 5.0 BLE

Ethernet: Gigabit port

USB: 2x USB 3.0, 2x USB 2.0

GPIO: 40-pin, backward compatible

HDMI: Dual Micro HDMI, 4K support

Camera: MIPI DSI/CSI connectors

Audio: 4-pole AV jack

Video: H.265 4Kp60, H.264 1080p60

Graphics: OpenGL ES 3.0

Storage: MicroSD slot

Power: 5V 3A USB-C or GPIO (PoE supported with HAT).

2.6.2. Esp32

The ESP32 is a powerful, versatile module that supports Wi-Fi, Bluetooth, and BLE, making it ideal for a wide range of projects. It is well-suited for applications requiring networking, low power consumption, or demanding tasks such as voice encoding, music playback, and MP3 decoding.

ESP32 has a dual-core CPU, with each core being independently controllable, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz.

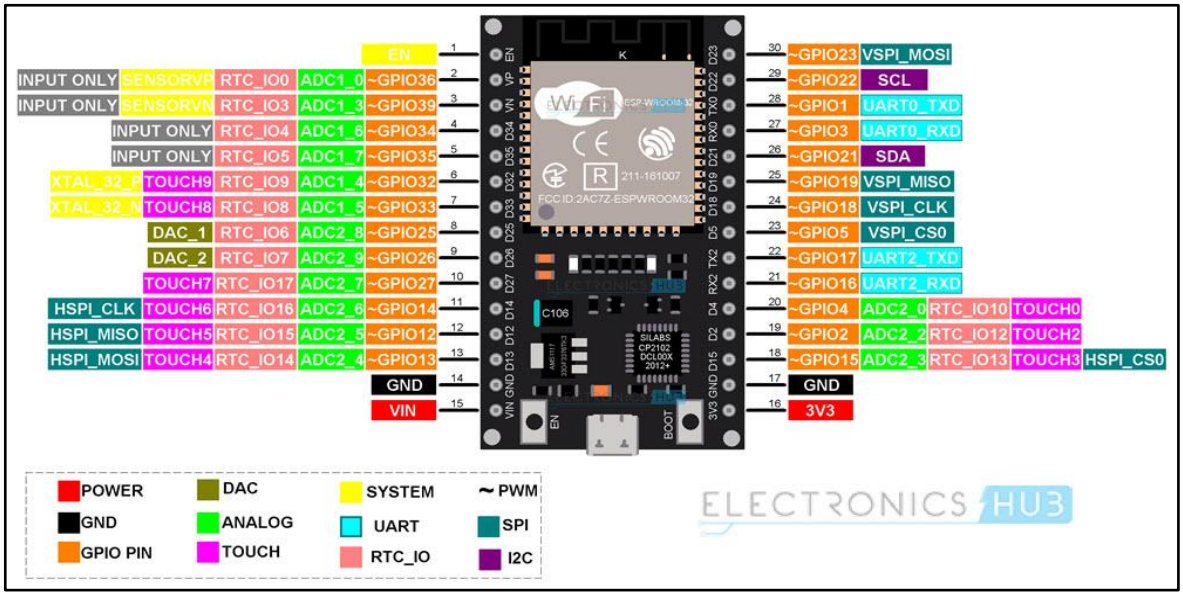

Figure 2.12. Esp32 Wroom Pin Diagram

Port: Type C/Micro USB

Model: ESP32 (38/30 pins)

Power: 5V USB, 3.3V I/O

Frequency: 240 MHz

Wi-Fi: 802.11 b/g/n/e/i (2.4 GHz, 150 Mbps)

Bluetooth: 4.2 dual-mode

Memory: 448 KB ROM, 520 KB SRAM, 6 KB RTC SRAM

GPIO: 24 pins (some input-only)

Analog: 12-bit ADC (18 channels, PGA on some pins)

Security: IEEE 802.11, WPA/WPA2/WAPI.

2.6.3. MQ-135

MQ-135 is a gas sensor from the MQ series, based on a **SnO_2** sensing element whose conductivity varies with gas concentration. It is commonly used for **air-quality monitoring** by responding to gases/vapors such as **NH_3 (ammonia), NOx, CO_2, smoke, alcohol vapor, benzene**, and other volatile compounds. Typical breakout modules provide an **analog output** (voltage related to sensor resistance), and sometimes a **digital output** via an onboard comparator with an adjustable threshold. In IoT applications, MQ-135 is often used for **trend monitoring of pollution/air quality**, while accurate ppm measurement requires **proper calibration** for the target gas and environment.

Figure 2.13. MQ-135

- **Module supply voltage:** 5 VDC (typical)
- **Outputs:** Analog (0–5 V depending on module); Digital (optional comparator)
- **Target gases / use cases:** NH_3, NOx, CO_2, smoke, benzene, alcohol/volatile gases (different sensitivities)

- **Principle:** SnO_2 resistance changes with gas concentration
- **Heater requirement:** warm-up time needed; initial burn-in recommended

2.6.4. DHT11 Temperature and Humidity Sensor

The DHT11 temperature and humidity sensor is widely used due to its low cost and ease of data retrieval via one-wire communication. The integrated signal preprocessor in the sensor ensures accurate data without the need for any additional calculations.

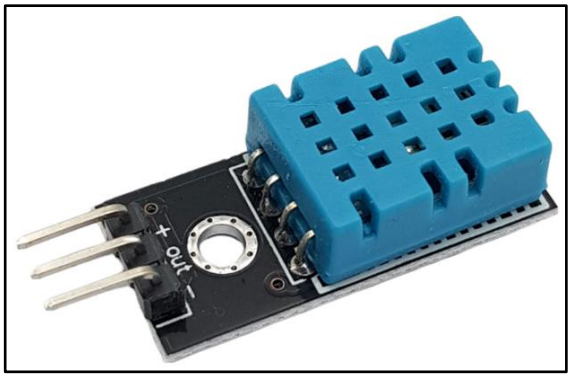

Figure 2.14. DHT11 Sensor

Operating voltage: 5VDC

Communication protocol: TTL, One-wire

Humidity range: 20%-80% RH, accuracy ± 5% RH

Temperature range: 0-50°C, accuracy ± 2°C

Maximum sampling frequency: 1Hz (once per second)

2.6.5. BMP280 + ATH20

BMP280 is a **barometric pressure** sensor with an integrated **temperature** measurement. It is widely used for **atmospheric pressure monitoring**, **relative altitude estimation**, and weather-trend analysis. The sensor supports **I²C/SPI** interfaces, low power operation, and good stability for IoT deployments.

AHT20 is a digital **temperature and humidity** sensor designed for stable real-world measurements. It communicates via **I²C**, making it suitable for environmental monitoring stations thanks to good accuracy and a straightforward readout process.

Figure 2.15. BMP280+ATH20

2.6.6. OLED 0.96

The **0.96-inch OLED** (commonly based on the **SSD1306** controller) is a monochrome display (often **white/blue**) that provides **high contrast**, good readability, and **low power consumption** thanks to OLED technology. In IoT systems, a 0.96" OLED is typically used to show **real-time sensor readings** (temperature, humidity, pressure, CO_2, dust, AQI), connectivity status (Wi-Fi/MQTT/LoRa), and alerts. Most modules use the **I²C** interface for simple integration with ESP32/Raspberry Pi.

Figure 2.16. 0.96-inch OLED

2.6.7. GP2Y1010AU0F

GP2Y1010AU0F is an optical **dust sensor** based on **light scattering**. It integrates an **IR LED** and a **photodiode**; airborne particles scatter the emitted light, and the received signal is converted to an **analog voltage output** that correlates with dust density. The sensor is suitable for **trend monitoring of particulate matter** (e.g., PM2.5/PM10 as indicative levels) in IoT environmental monitoring systems. For stable readings, the LED is typically **pulsed** and the ADC samples at a defined timing; airflow, dust accumulation, and ambient conditions can affect results, so **filtering and calibration** are recommended for concentration estimation.

Figure 2.17. GP2Y1010AU0F

2.6.8. Lora E32 433T20D

LoRa E32-433T20D (Ebyte) is a long-range wireless module based on **LoRa** technology (powered by the **Semtech SX1278**) with a **TTL-level UART** interface operating in transparent serial transmission mode. It is well-suited for IoT deployments requiring long-range, low-power, and robust links (e.g., environmental monitoring). The module provides **M0/M1** pins for operating-mode selection and an **AUX** pin for busy/ready status indication during TX/RX.

Figure 2.18. LoRa E32-433T20D

CHAPTER III: SYSTEM ANALYSIS AND DESIGN

3.1. SYSTEM OVERVIEW

The **Raspberry Pi 4B** and **ESP32 sensor nodes** (one or multiple nodes) operate within the same networking infrastructure (Wi-Fi; or Wi-Fi for the gateway and **LoRa** between nodes and the gateway depending on deployment).

Each **ESP32 Sensor Node** interfaces with **environmental sensors** (e.g., temperature, humidity, pressure, CO_2, particulate/dust) and samples data periodically.

The **Raspberry Pi 4B** acts as the **central hub/gateway**, responsible for:

- Running an **MQTT broker** (e.g., Mosquitto) to support the publish/subscribe messaging model.
- Running **Node-RED** for data-flow orchestration: subscribing to topics, validating/normalizing payloads, timestamping, and coordinating system logic.
- Persisting collected data into **MySQL** (or an equivalent database) for analytics and model training.

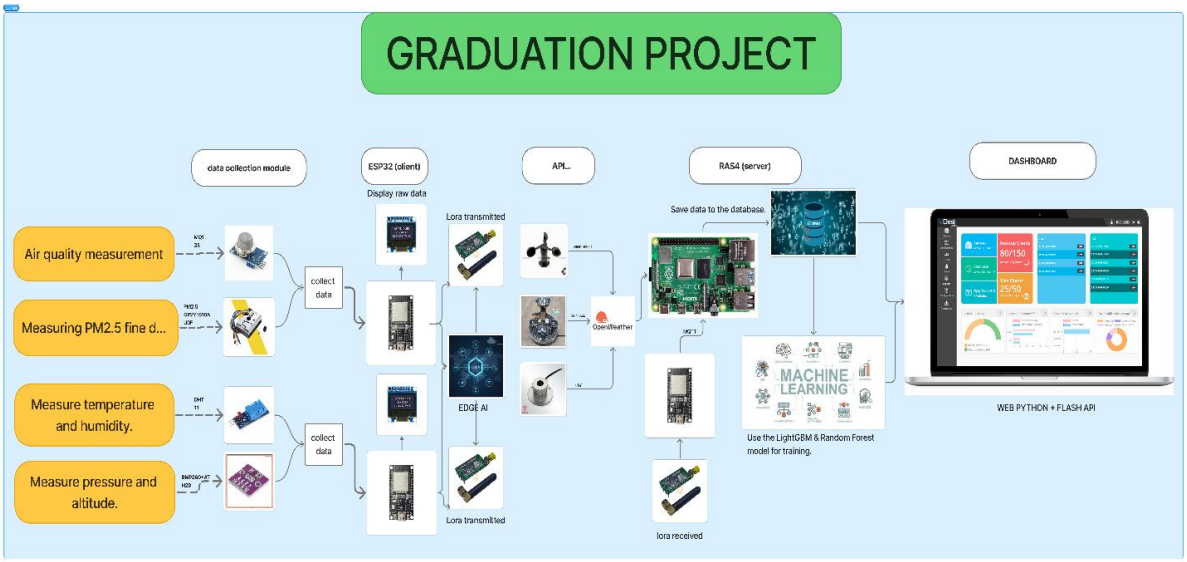

Figure 3.1. System Overview

The **ESP32 devices** function as **MQTT clients**:

- They **publish** sensor measurements to structured topics (by node and by sensor type).
- They **subscribe** to control topics (when actuators such as relays/fans are included) and execute received commands.

On the Raspberry Pi, the system deploys a **backend service (e.g., FastAPI)** and a **machine learning module** to:

- Extract historical time-series data from the database.
- Train/update forecasting models (e.g., **24-hour** and **7-day** predictions for key environmental metrics).
- Store forecast outputs back to the database and expose APIs for the user interface.

3.2. OVERALL SYSTEM ARCHITECTURE

The system adopts a multi-layer AIoT architecture: (1) device/sensing layer (ESP32 + environmental sensors), (2) long-range communication layer (LoRa) for uplink to a central point, (3) gateway/edge layer (Raspberry Pi + Node-RED + MQTT Broker + MySQL) for ingestion, integration, and persistence, and (4) application layer (FastAPI + Web Frontend) for API services, visualization, and administration. This design fits deployments in schools/parks where nodes may be distant from the server, energy efficiency matters, and communication stability is prioritized. Edge computing on the Raspberry Pi reduces reliance on cloud services while still enabling local forecasting and visualization.

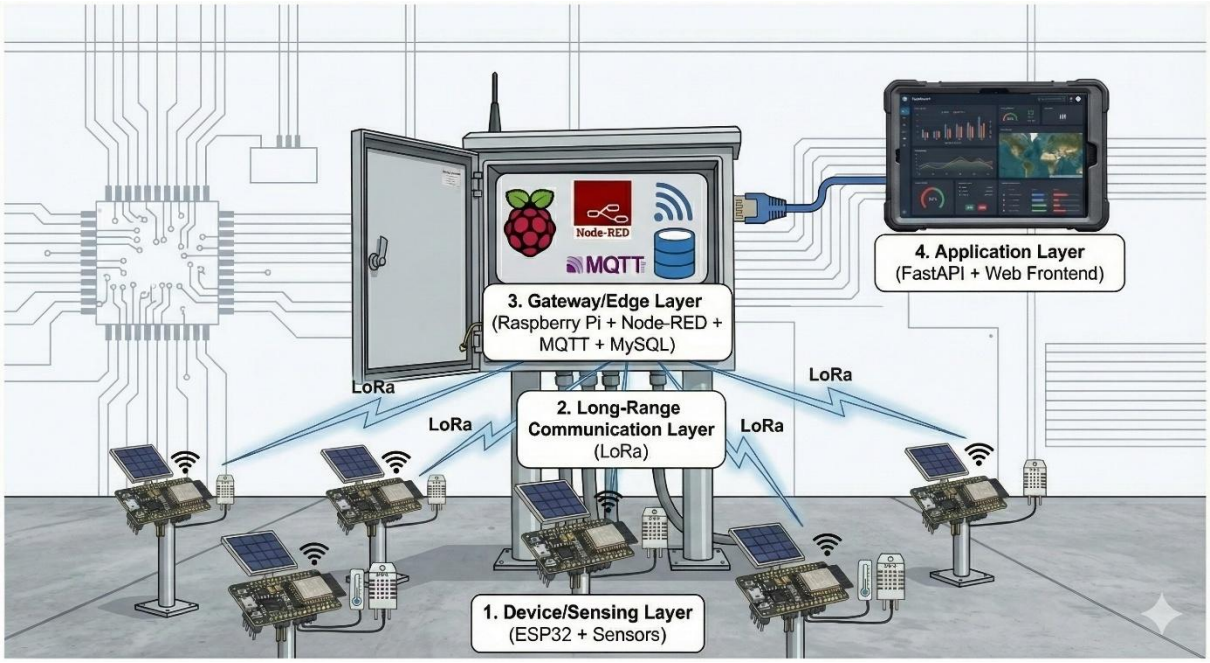

Figure 3.2. Overall System Architecture

3.2.1. System Block Diagram

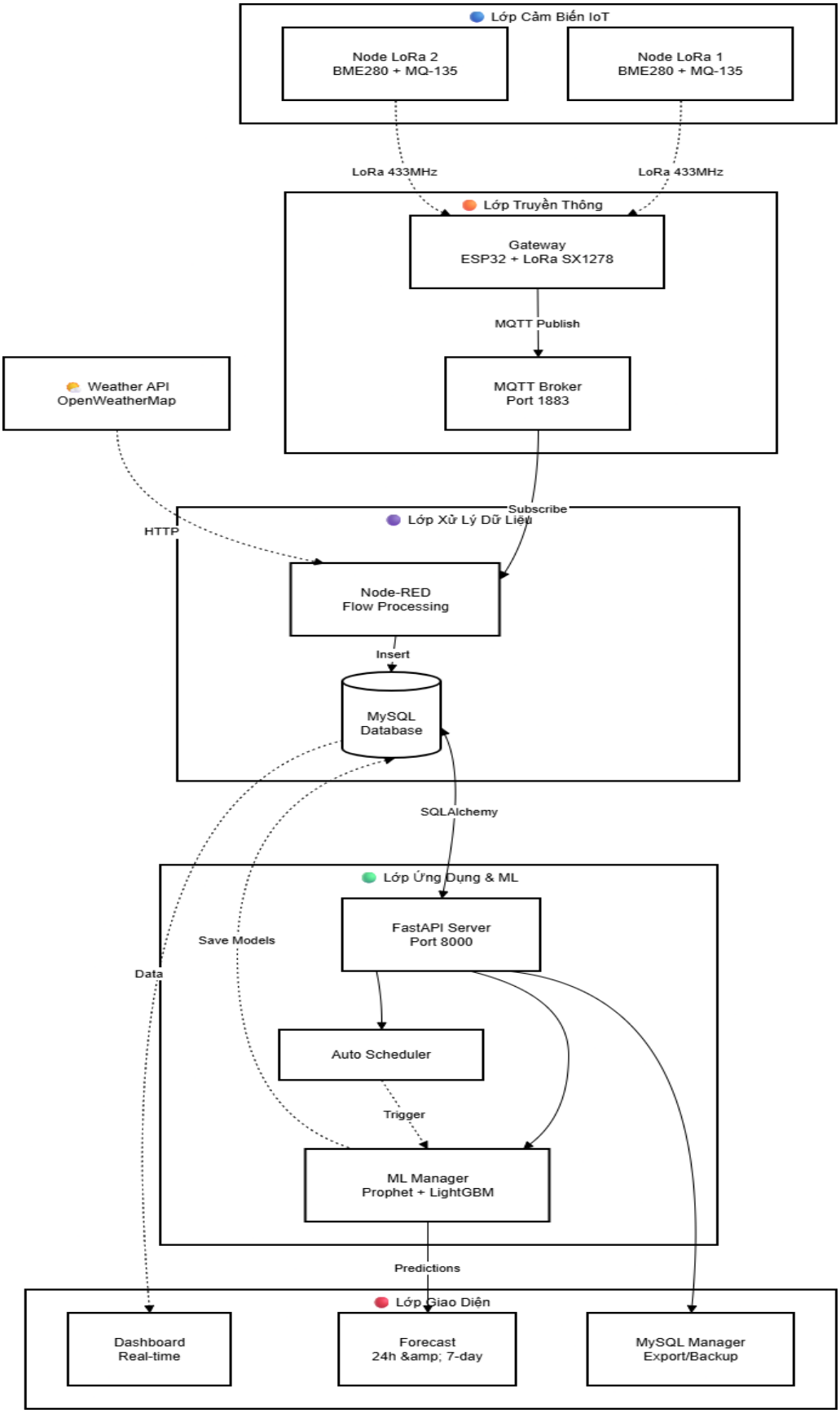

Figure 3.3. System Block Diagram

3.2.2. Algorithm Flowcharts

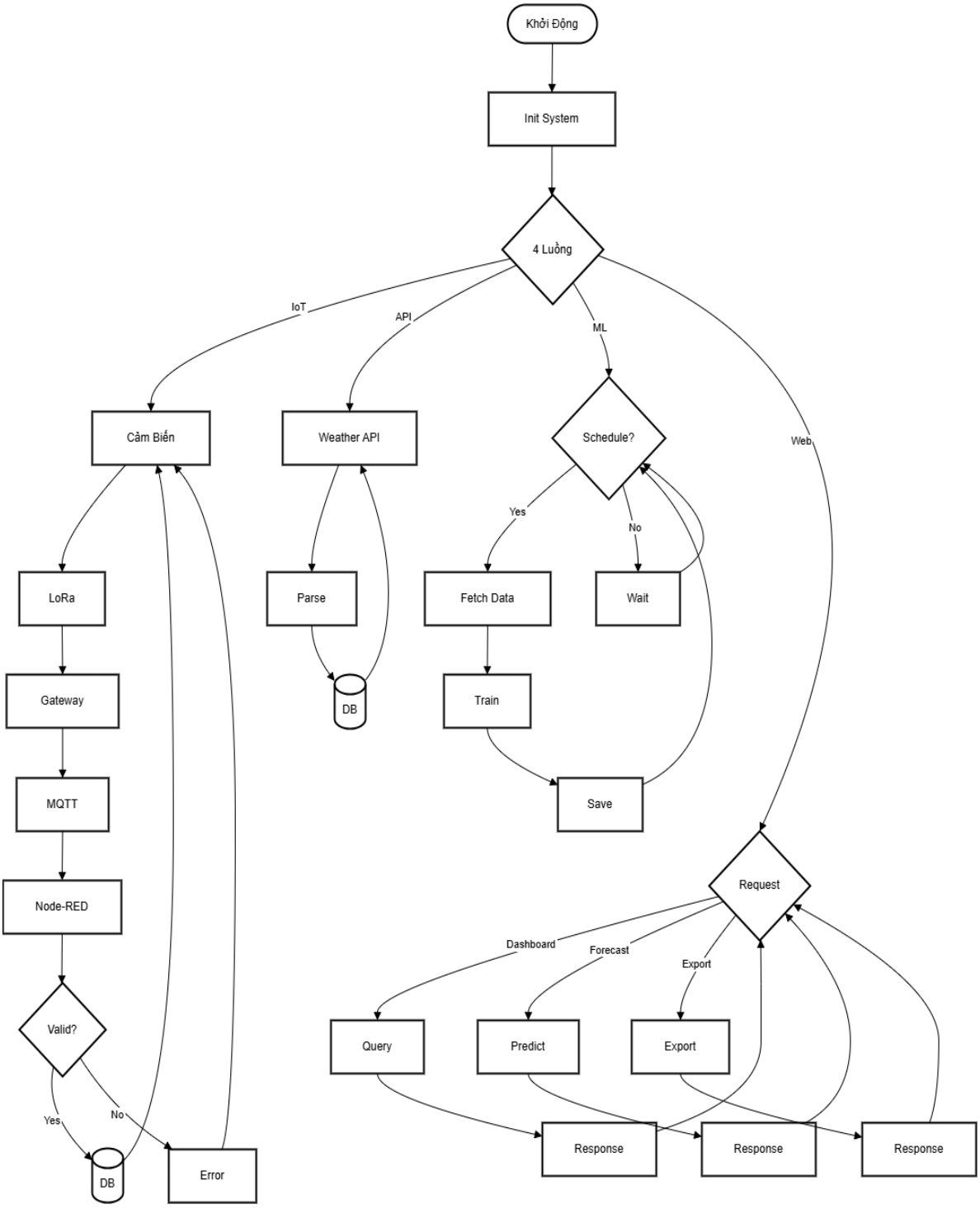

Figure 3.4. Algorithm Flowcharts

3.3. HARDWARE DESIGN

3.3.1. Schematic Design

The hardware consists of two main blocks: **(1) environmental sensing nodes** and **(2) gateway/edge ingestion**. In each node, the ESP32 acts as the controller, interfacing with sensors through appropriate buses: temperature/humidity sensor (DHT11/AHT20)

via GPIO or I2C, BMP280 pressure sensor via I2C, MQ-135 gas sensor via ADC (analog voltage interpreted as a relative concentration), and dust sensor (GP2Y1010/GP2Y1014) via ADC with LED drive timing per datasheet to stabilize readings. The E32 LoRa module connects to the ESP32 through UART

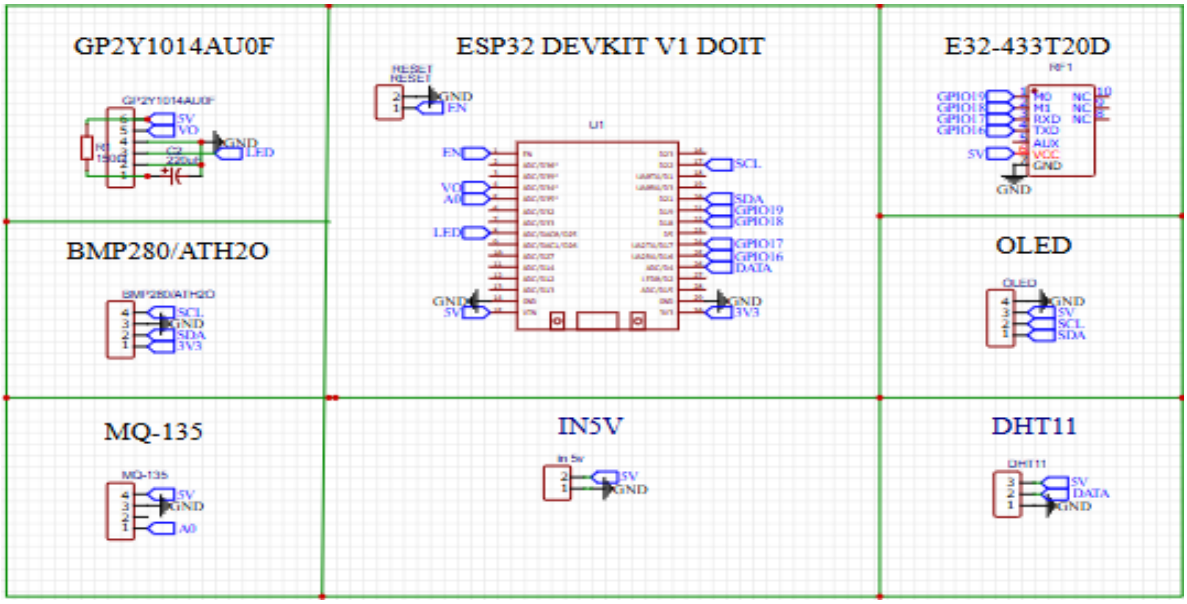

Figure 3.5. Schematic

3.3.2. PCB Design

PCB design focuses on analog measurement stability and RF communication quality. For sensing nodes, components are placed by functional zones: power section (input terminal, diode, regulator, bulk/decoupling capacitors), digital section (ESP32, I2C pull-ups, programming header), analog section (MQ-135 and ADC traces, dust sensing circuitry), and RF/LoRa section

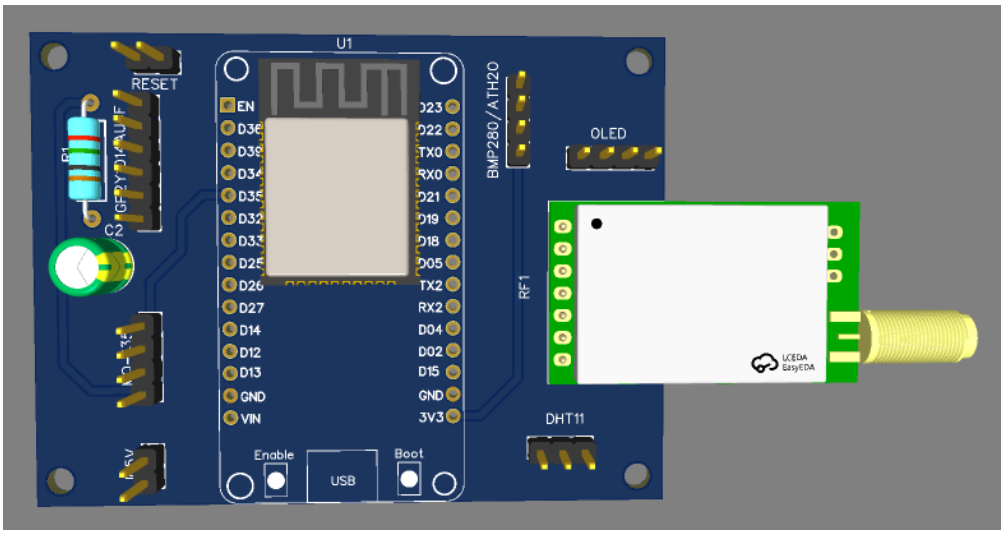

Figure 3.6. 3D PCB

3.3.3. Finished Products

The device is an **enclosure-based environmental monitoring module** designed to collect and display real-time air parameters. On the front panel, the main components include a **0.96" OLED display** for data visualization, a **push button** for switching modes/viewing different parameter pages, a **status LED** indicating power/operation, and integrated **environmental sensors**.

The sensor system includes:

- A **temperature–humidity sensor module** (DHT-type) located on the right side.
- An **atmospheric pressure sensor breakout board** (BMP280-type) located on the left side.
- A **metal-can gas sensor** (MQ series, commonly used for air-quality trend monitoring) placed in the center to assess pollution/gas levels in the air in a relative manner.

An **antenna** is mounted at the top, making the device suitable for wireless data transmission scenarios (e.g., **Wi-Fi/LoRa**, depending on the design). This layout improves visibility, ease of operation, and maintenance, and it meets the requirements for environmental monitoring in classrooms, corridors, school campuses, or other public areas

Figure 3.7. Finished Products

The sensor system includes:

• A **temperature–humidity sensor module** (DHT-type) located on the right side.

• An **atmospheric pressure sensor breakout board** (BMP280-type) located on the left side.

• A **metal-can gas sensor** (MQ series, commonly used for air-quality trend monitoring) placed in the center to assess pollution/gas levels in the air in a relative manner.

An **antenna** is mounted at the top, making the device suitable for wireless data transmission scenarios (e.g., **Wi-Fi/LoRa**, depending on the design). This layout improves visibility, ease of operation, and maintenance, and it meets the requirements for environmental monitoring in classrooms, corridors, school campuses, or other public areas

3.4. SOFTWARE DESIGN

3.4.1. Data Ingestion and Processing Layer (MQTT/Node-RED)

Node-RED orchestrates the ingestion pipeline: MQTT subscription, payload parsing, time alignment, and MySQL persistence.

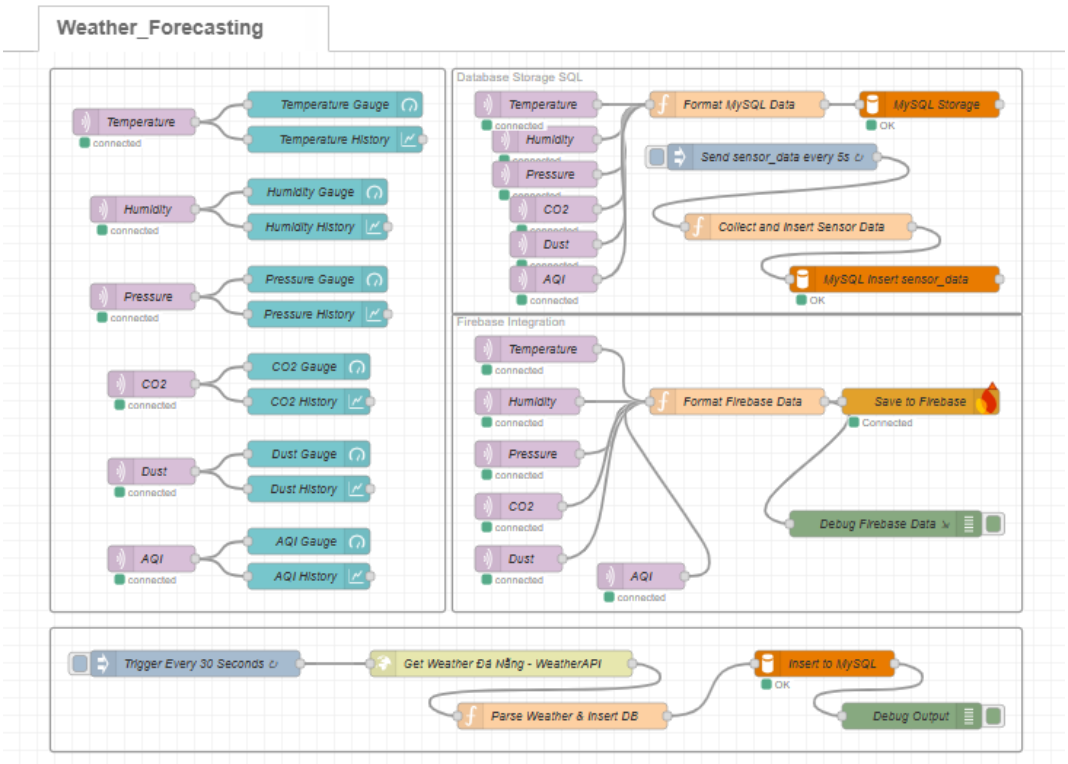

Figure 3.8. Node-RED flow

The flow is designed to be fault-tolerant: if the DB connection fails, Node-RED logs the issue and may buffer messages (context/file) to reduce data loss; if fields are missing, the flow flags and discards or stores them in an error table for analysis. Node-

RED can also periodically call the Weather API and persist results, separating API ingestion from the backend, reducing API server load, and simplifying deployment on the Raspberry Pi.

3.4.2. FastAPI Backend

FastAPI is designed as the central service layer of the web application, acting as the core backend that connects the database, machine learning components, and the user interface. It exposes a set of well-structured RESTful APIs that support key functions such as querying real-time and historical data, managing stored records, and triggering machine learning training and forecasting processes on demand. This design allows the frontend dashboard and management pages to interact with the system in a decoupled and scalable manner

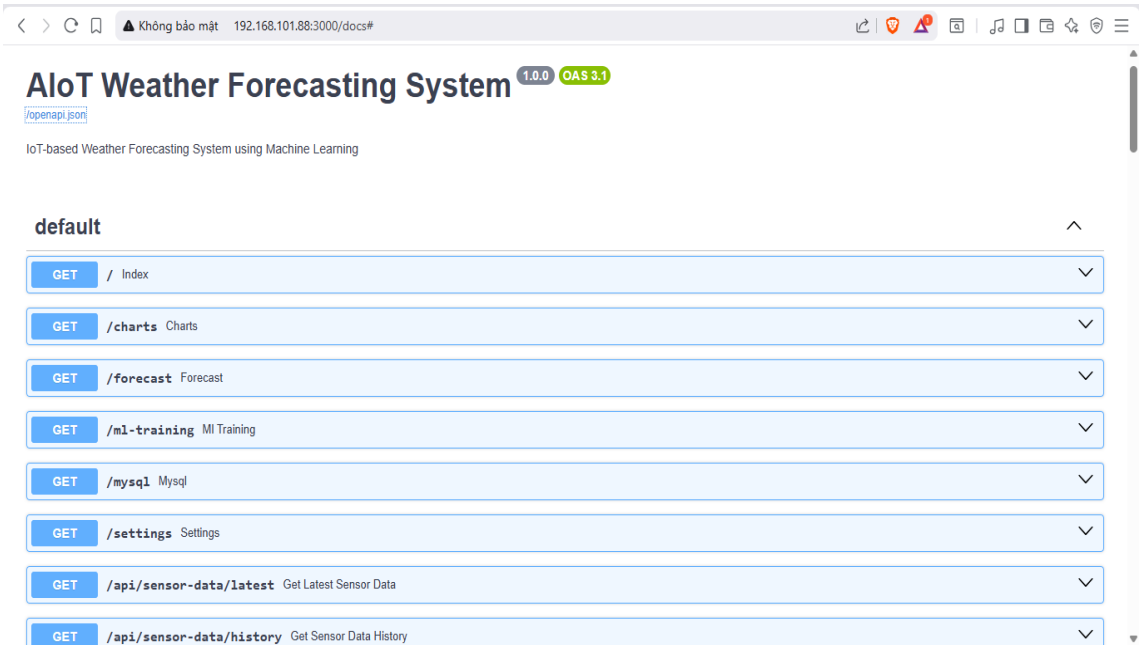

Figure 3.9. API UI

For data modeling and persistence, This abstraction layer ensures consistent data access patterns, improves code maintainability, and significantly reduces the risk of manual SQL errors. In addition, the ORM-based design simplifies schema evolution, supports transactional integrity, and enables easier integration with data validation mechanisms. Combined with FastAPI's asynchronous capabilities and automatic API documentation

3.4.3. Web Frontend

The web interface is designed with a "dashboard-first" approach, focusing on fast observability of environmental conditions and forecast outputs. The Dashboard page

shows cards for current values, time-series line charts for historical telemetry, and system status indicators such as last update time and node online/offline states. The Forecast page provides 24-hour and 7-day forecast charts

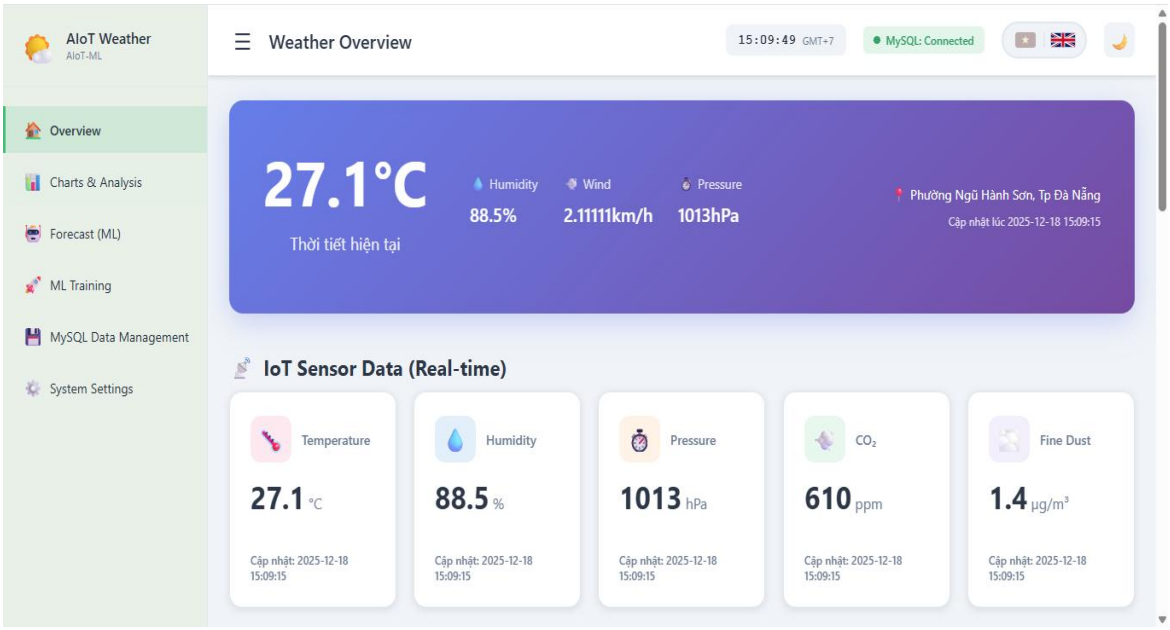

Figure 3.10. Dashboard

In addition, the web interface emphasizes **clarity, interactivity, and operational usability** to support both monitoring and management tasks. The dashboard layout adopts a card-based design that allows users to quickly grasp the current environmental state at a glance, while color coding and icons help distinguish different parameters such as temperature, humidity, pressure, CO_2 concentration, and fine dust levels. Time-series line charts are integrated to visualize historical trends, enabling users to observe diurnal patterns, seasonal variations, and sudden changes in environmental conditions.

Beyond data visualization, the interface also incorporates **system awareness and maintenance support**. Indicators such as the last data update timestamp, database connection status, and node connectivity (online/offline) provide immediate feedback on system health and reliability. On the Forecast page, interactive charts present both 24-hour and 7-day prediction results, allowing users to compare short-term and medium-term outlooks and better anticipate environmental or weather changes. Additional pages support data management functions, including filtering by time range, exporting datasets for offline analysis, and deleting outdated records to maintain database efficiency. Overall, the web interface serves as a centralized control and visualization platform, bridging raw data collection, machine learning forecasts, and user-oriented decision support in a coherent and intuitive manner.

CHAPTER IV: RESULTS AND EVALUATION

4.1. SYSTEM IMPLEMENTATION RESULTS

After the design and implementation phases, the AIoT system was deployed following a multi-layer architecture including IoT sensing nodes, LoRa communication, an edge/gateway layer on Raspberry Pi (MQTT Broker + Node-RED + MySQL), and the application layer (FastAPI + Web Dashboard). During operation, environmental measurements (temperature, humidity, pressure, CO_2, dust/PM2.5, AQI) are periodically collected, packaged into timestamped records, and persistently stored in MySQL. In addition, the system integrates supplementary weather data from a Weather API (wind speed/direction, rainfall, UV index) to enrich the meteorological context and potentially improve forecasting performance.

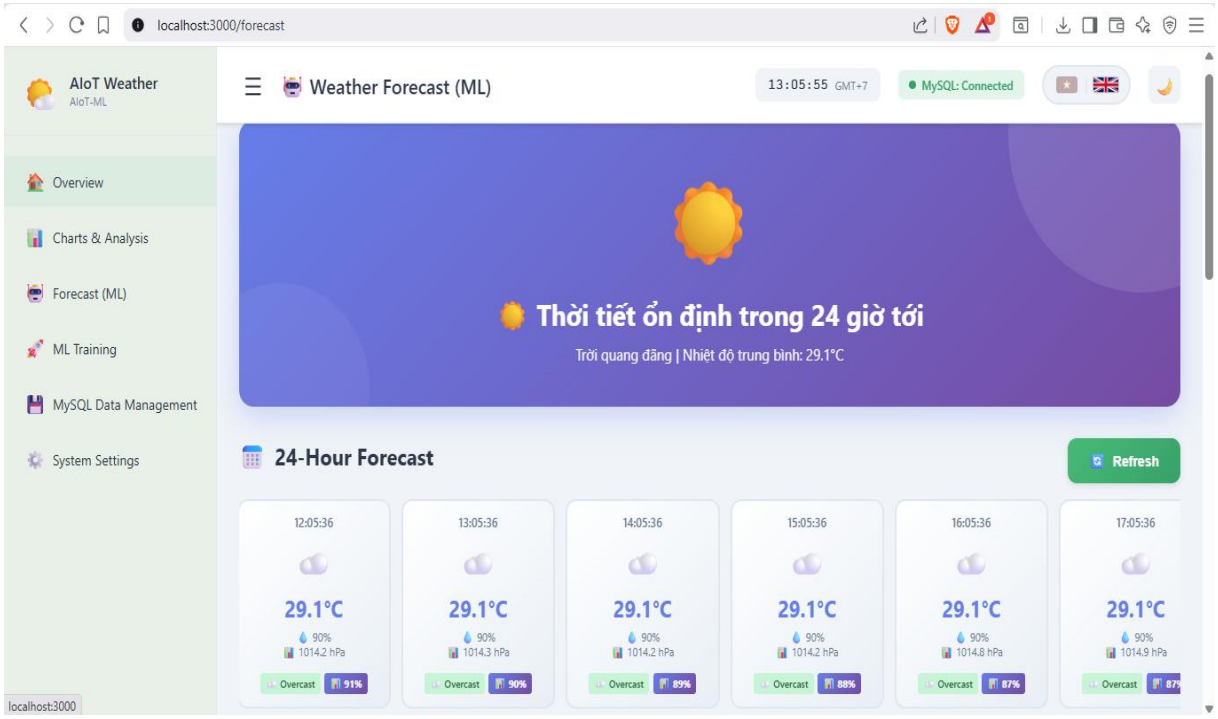

Figure 4.1. Interface Forecasting

On the dashboard, users can monitor real-time readings, inspect time-series trend charts, and access data management functions (time-range filtering, quick queries, exporting, and conditional deletion). With centralized and consistent storage, the dataset for model training preserves time-series continuity, minimizes manual processing, and facilitates systematic evaluation and comparison of forecasting models.

4.2. FORECASTING EVALUATION

4.2.1. Evaluation methodology and metrics

To evaluate forecasting quality, two model families are considered: (i) **Prophet**, well-suited for time series with trend and seasonality; and (ii) **LightGBM**, a gradient-boosting machine learning model capable of leveraging multivariate feature sets. The dataset is split into training and

Time	Temperature (°C)	Humidity (%)	Pressure (mb)	Wind Speed	Rainfall	UV Index	Season	Weather	Accuracy
24/12/2025 07:03:05	29.1°C	90%	1014.1 mb	1.3 km/h	0 mm	1.5 (Low)	Winter	Morning	95%
24/12/2025 08:03:05	29.1°C	90%	1014 mb	1.3 km/h	0 mm	1.6 (Low)	Winter	Morning	94%
24/12/2025 09:03:05	29.1°C	90%	1014 mb	1.3 km/h	0 mm	1.5 (Low)	Winter	Morning	93%
24/12/2025 10:03:05	29.1°C	90%	1014 mb	1.3 km/h	0 mm	1.5 (Low)	Winter	Overcast	92%
24/12/2025 11:03:05	29.1°C	90%	1014.2 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Overcast	91%
24/12/2025 12:03:05	29.1°C	90%	1014.2 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Overcast	91%
24/12/2025 13:03:05	29.1°C	90%	1014.3 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Overcast	90%
24/12/2025 14:03:05	29.1°C	90%	1014.2 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Overcast	89%
24/12/2025 15:03:05	29.1°C	90%	1014.2 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Overcast	88%
24/12/2025 16:03:05	29.1°C	90%	1014.8 mb	1.3 km/h	0 mm	1.3 (Low)	Winter	Overcast	87%
24/12/2025 17:03:05	29.1°C	90%	1014.9 mb	1.3 km/h	0 mm	1.3 (Low)	Winter	Overcast	87%
24/12/2025 18:03:05	29.1°C	90%	1015 mb	1.3 km/h	0 mm	1.3 (Low)	Winter	Overcast	86%
24/12/2025 19:03:05	29.1°C	90%	1014.8 mb	1.3 km/h	0 mm	1.3 (Low)	Winter	Evening	85%
24/12/2025 20:03:05	29.1°C	90%	1014.8 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Evening	84%
24/12/2025 21:03:05	29.1°C	90%	1014.8 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Clear Night	83%
24/12/2025 22:03:05	29.1°C	90%	1014.7 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Clear Night	83%
24/12/2025 23:03:05	29.1°C	90%	1014.6 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Clear Night	82%
25/12/2025 00:03:05	29.1°C	90%	1014.6 mb	1.3 km/h	0 mm	1.4 (Low)	Winter	Overcast	81%
25/12/2025 01:03:05	29.1°C	90%	1014.5 mb	1.3 km/h	0 mm	1.3 (Low)	Winter	Clear Night	80%
25/12/2025 02:03:05	29.1°C	90%	1014.5 mb	1.3 km/h	0 mm	1.3 (Low)	Winter	Clear Night	79%
25/12/2025 03:03:05	29.1°C	90%	1014.5 mb	1.3 km/h	0 mm	1.3 (Low)	Winter	Clear Night	79%

Figure 4.2. Train Model Data File

testing subsets in chronological order to avoid data leakage. Predictions are then compared to the ground-truth measurements over the same test horizon. The main metrics include **MAE** (Mean Absolute Error), **RMSE** (Root Mean Squared Error), and **R^2** (coefficient of determination). Lower MAE/RMSE indicates better accuracy, while R^2 closer to 1 suggests stronger explanatory power.

4.2.2. Comparison using Forecast vs Actual plots

Forecast vs actual plots provide an intuitive view of how closely predictions follow real observations. With Prophet, the forecast curve typically captures overall trend and seasonality (assuming sufficient and stable historical data), but it may respond slowly to abrupt changes (spikes) caused by environmental shifts or sensor noise. In contrast, LightGBM often better tracks local fluctuations when features are properly engineered (e.g., lag features, rolling statistics, and Weather API signals), yet it can

degrade if the dataset is unstable or the feature set fails to reflect the underlying environmental dynamics.

4.2.3. Feature importance

For LightGBM, **feature importance** is used to interpret the contribution of each feature to the prediction. Results typically indicate that temporal features (hour-of-day, day-of-week), lagged variables, and rolling statistics (rolling mean/rolling standard deviation) substantially influence environmental forecasts. Additionally, selected Weather API signals (e.g., rainfall, wind direction, wind speed) can enhance predictions during rapid weather transitions, highlighting the value of integrating exogenous data sources in environmental forecasting tasks.

4.3. OVERALL ASSESSMENT

Overall, the system meets the project's core objectives: (1) collecting multi-sensor environmental data remotely; (2) enabling stable long-range LoRa communication; (3) centralized storage in MySQL via Node-RED/MQTT; (4) providing monitoring and data administration on the dashboard; and (5) integrating training/forecasting modules to generate actionable predictions. Key strengths include a modular design (easy to expand nodes/sensors), feasibility on edge infrastructure (Raspberry Pi) reducing cloud dependency, and an end-to-end data pipeline from acquisition → storage → training → visualization.

Model	Số Targets	Trung Bình Accuracy
Prophet	9	39.07%
LightGBM 🏆	9	71.10%

Figure 4.3. Compare Models

Nevertheless, several limitations remain. First, forecasting quality strongly depends on the length of historical data, sensor stability, and sampling frequency; noisy or missing data degrades both models. Second, under rapidly changing conditions, Prophet may lag behind sudden variations, whereas LightGBM requires robust feature engineering and may overfit if not properly controlled. Third, the current scope mainly focuses on forecasting environmental variables and evaluating errors, leaving further

CONCLUSION AND DEVELOPMENT DIRECTION

1. Conclusion

The project "AIoT-Based Weather Forecasting System Using Machine Learning" successfully designed and deployed an end-to-end AIoT system for environmental monitoring and forecasting using a multi-layer architecture. The system achieves the primary objectives: (i) collecting environmental data from IoT sensing nodes; (ii) enabling long-range uplink via LoRa to a gateway; (iii) performing edge-layer integration on Raspberry Pi using MQTT, Node-RED, and centralized persistence in MySQL; (iv) providing a web dashboard for monitoring, visualization, and data administration; and (v) integrating training/forecasting modules with Prophet and LightGBM for 24-hour and 7-day horizons. The practical deployment demonstrates feasibility for school/park scenarios that require stable long-range communication and reduced dependence on cloud infrastructure.

In terms of contributions, the project delivers a closed-loop pipeline from data acquisition → storage → training → forecasting → visualization, with modular design and scalability. Moreover, the comparative analysis between Prophet (time-series modeling) and LightGBM (machine learning boosting) clarifies strengths and weaknesses of each approach under environmental sensor data conditions, supporting model selection for real-world requirements..

2. imitations of the topic

Despite successfully meeting the stated objectives, the proposed system still exhibits several limitations that should be acknowledged and addressed in future work. First, the accuracy and stability of the forecasting models are highly dependent on the length, continuity, and quality of the historical time-series data. When the collected dataset is relatively short, contains missing samples, or is affected by noise caused by environmental fluctuations or communication interruptions, the learning process of the models becomes less reliable. As a result, prediction errors may increase, particularly for longer forecasting horizons such as 7-day forecasts, where cumulative uncertainty becomes more significant.

Second, limitations related to sensor hardware and deployment conditions can directly affect the reliability of the input data. Over time, low-cost environmental sensors may experience calibration drift, aging effects, or sensitivity degradation, leading to biased measurements. In addition, factors such as sensor placement, physical shielding, exposure to direct sunlight or rain, and unstable power supply can further

distort the collected data. These issues may cause the machine learning models to be trained on data distributions that do not accurately represent the true environmental conditions, thereby reducing generalization performance and robustness.

Third, the current system scope primarily concentrates on real-time monitoring and short-term forecasting of basic environmental and weather-related variables. More advanced analytical and intelligent features—such as anomaly detection for identifying abnormal environmental events, context-aware early warning mechanisms, adaptive alert thresholds, or automated model selection and hyperparameter tuning—have not yet been fully implemented. Consequently, the system still relies on relatively static configurations and manual intervention for maintenance and performance optimization. Addressing these limitations in future iterations would significantly enhance the intelligence, autonomy, and practical value of the AIoT-based environmental monitoring and forecasting platform.

3. Development direction

Future improvements of the proposed system can be developed along several key directions to enhance forecasting accuracy, scalability, intelligence, and long-term sustainability.

First, with respect to **modeling and data analytics**, more advanced time-series forecasting techniques can be explored and integrated into the system. Deep learning–based models such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Temporal Convolutional Networks (TCN), and time-series Transformers are capable of capturing complex temporal dependencies, nonlinear patterns, and long-range correlations that traditional models may fail to represent. In addition, adopting rolling-window training and evaluation strategies, together with systematic hyperparameter optimization, would help improve model generalization, robustness, and adaptability to changing environmental conditions over time.

Second, in terms of **network scalability and system expansion**, the architecture can be extended to support a larger number of sensor nodes across wider geographical areas. This can be achieved by standardizing device identification and addressing schemes, introducing reliable time synchronization mechanisms, and implementing remote configuration and firmware management. Such enhancements would enable centralized monitoring and maintenance of distributed nodes, making the system more suitable for large-scale deployments in smart campuses, urban monitoring networks, or regional environmental observation systems.

Third, for **real-time operation and intelligent response**, a dedicated alert and warning module can be incorporated into the platform. This module may combine threshold-based rules with anomaly detection algorithms to identify abnormal environmental conditions or sudden weather changes. When predefined conditions are met, notifications can be automatically delivered through multiple channels, including the web dashboard, email, or popular messaging platforms. This functionality would allow users and administrators to respond promptly to potential risks, thereby increasing the practical usefulness and safety value of the system.

Finally, regarding **infrastructure and power management**, the system architecture can gradually evolve toward cloud-based or hybrid deployments. Cloud integration would facilitate long-term data storage, remote access, automated backups, and improved system reliability. At the same time, for deployments in remote or hard-to-access locations, solar-powered sensor nodes can be investigated to enhance energy autonomy, reduce dependence on grid power, and lower maintenance costs. Together, these improvements would strengthen the system's scalability, resilience, and sustainability, making it more suitable for real-world, long-term environmental monitoring and weather forecasting applications.

REFERENCES

[1]. Harris, D.M. and Harris, S.L. (2012), *Digital Design and Computer Architecture* (2nd ed.), Morgan Kaufmann, USA.

[2]. Mazidi, M.A., Mazidi, J.G. and McKinlay, R.D. (2006), *The 8051 Microcontroller and Embedded Systems: Using Assembly and C* (2nd ed.), Pearson/Prentice Hall, USA.

[3]. Kleitz, W. (2007), *Digital Electronics: A Practical Approach with VHDL* (2nd ed.), Prentice Hall, USA.

[4]. OASIS (2014), *MQTT Version 3.1.1*, OASIS Standard, 29 Oct 2014.

[5]. Eclipse Foundation (n.d.), *Eclipse Mosquitto Documentation*, Mosquitto Project Documentation.

[6]. Node-RED (n.d.), *Node-RED Documentation / User Guide*, Node-RED Official Documentation.

[7]. Oracle (n.d.), *MySQL 8.0 Reference Manual*, Oracle MySQL Documentation.

[8]. Ramírez, S. (n.d.), *FastAPI Documentation*, FastAPI Official Documentation.

[9]. SQLAlchemy Authors (n.d.), *SQLAlchemy Documentation (Unified Tutorial)*, SQLAlchemy Documentation.

[10]. Espressif Systems (n.d.), *ESP32 Series Datasheet*, Espressif Documentation.

[11]. Espressif Systems (n.d.), *ESP32 Technical Reference Manual*, Espressif Documentation.

[12]. Semtech Corporation (n.d.), *AN1200.22: LoRa Modulation Basics*, Semtech Application Note.

[13]. Chengdu Ebyte Electronic Technology Co., Ltd. (n.d.), *E32-433T20D User Manual*, Ebyte Documentation.

[14]. Raspberry Pi Ltd. (n.d.), *Raspberry Pi Documentation (Getting started / Raspberry Pi OS)*, Raspberry Pi Documentation.

[15]. OpenWeather (n.d.), *OpenWeather API Documentation (One Call API)*, OpenWeather Documentation.