Final Report - 6556ELEICB

BEng (Hons) in Mechatronics and Autonomous Systems

Engineering

IoT-Based Weather Forecasting Station

Student Name – Raneesha Fernando

LJMU Student ID – 967350

ICBT Student ID - CL/BENG/MASE/10/15

Supervisor Name – Ms. Sanjana Dias

Date of Submission – 05/06/2024

Acknowledgement

I would like to express my sincere gratitude to everyone who supported me throughout my final year project in Mechatronics Engineering at ICBT Campus Colombo, awarded by Liverpool John Moores University.

First and foremost, I want to thank my supervisor, Ms. Sanjana Dias, for her invaluable guidance, encouragement, and expertise throughout the project. Her support was instrumental in the successful completion of my research.

I am also deeply grateful to my colleagues for their assistance and collaboration, which contributed to the project's overall quality. Their willingness to share knowledge and provide feedback was invaluable.

I extend my appreciation to Mr. Ravindu Bandara, my Machine Learning lecturer, and Mr. Kasun Subasinghage, my Automation and IoT lecturer. Their technical knowledge and insights significantly enhanced my understanding of the subject matter and helped me overcome challenges during the project.

Finally, I want to thank ICBT Campus Colombo and Liverpool John Moores University for providing the resources and facilities that enabled me to conduct my research effectively.

This project has been a tremendous learning experience, and I am grateful to everyone who contributed to its success.

Abstract

This project presents the development of an IoT-based weather station tailored for agricultural applications in rural Sri Lanka. The system aims to address the challenges posed by unreliable weather forecasts by providing farmers with accurate and localized weather data, enabling informed decision-making for various agricultural activities. The weather station incorporates a range of sensors to collect real-time data on temperature, humidity, atmospheric pressure, and soil moisture. This data is transmitted to the ThingSpeak platform for storage, visualization, and analysis. Machine learning algorithms are employed to analyse the collected data and generate weather forecasts for the next hour and the next day. The system also includes an OLED display for real-time data presentation and utilizes the Arduino programming environment for seamless integration of components. While initial testing and analysis have shown promising results, further development is needed to enhance forecast accuracy and explore additional features, such as integrating external data sources and refining machine learning models. The project's findings highlight the potential of IoT-based weather stations to empower farmers with valuable information, contributing to improved agricultural productivity and sustainability in rural communities.

Table of Contents

Table of Figures

Tables

Chapter 01 – Introduction

1.1 Rationale of the topic

The weather is particularly important to agriculture. The development and health of crops are directly impacted by variables such as temperature, atmospheric pressure, and humidity. Weather patterns can have an impact on planting dates, crop choices, irrigation requirements, and even harvesting timetables. Storms, floods, and other extreme weather conditions can destroy crops, causing farmers to suffer large financial losses. It also has an impact on the availability of water, the frequency of illnesses and pests, and the quantity of fertilizer required. The deep and interconnected link between weather and agriculture emphasizes how crucial it is for farmers to keep an eye on and adjust to weather trends to succeed.

Regarding irrigation crop water requirements are determined by weather patterns. Having a thorough understanding of humidity and rainfall helps with irrigation schedule optimization, preventing water waste and underwatering. Also, Planting and harvesting accurate climate information aids in timing the planting and harvesting of crops to maximize both yield and quality. Additionally, in terms of disease and pest control the frequency of illnesses and pests is influenced by weather. Farmers can take precautionary action thanks to early warning systems based on weather trends.

Consequences of Inaccurate Weather Forecasts in Agriculture, Because of the chaotic nature of the environment, even slight changes in the starting circumstances can have a significant impact on future forecasted weather patterns. Beyond a certain point in time, it is difficult to predict the weather with complete trust due to its intrinsic complexity. Regarding the Economic Impact Poor decision-making brought on by inaccurate projections can result in financial losses from crop failure, excessive resource use, or lost market possibilities. When it comes to the Effect on the Environment Inaccurate projections can lead to misguided agricultural practices that negatively damage the environment by degrading soil, using more pesticides, and wasting water.

Machine learning can evaluate large volumes of data and find patterns that older methods would overlook; it has become an essential tool in weather forecasting. Machine Learning vs. Classical Methods: Classical forecasting techniques rely on historical data and statistical

models. Neural networks and other machine learning algorithms can analyse large datasets and intricate patterns to provide predictions that are more accurate. Increasing Forecast Accuracy By considering non-linear correlations in weather data and adjusting for shifting trends, machine learning techniques, such as deep learning, can improve predicting accuracy.

1.2 Problem Identification

Unreliable weather forecasts pose a significant challenge for farmers in Sri Lanka. Current weather stations tend to be concentrated in cities, neglecting the diverse microclimates present in rural agricultural regions. This data gap leaves farmers vulnerable to unpredictable weather conditions, jeopardizing their livelihoods and agricultural productivity.

Recent reports from the Department of Meteorology highlight the inaccuracy of weather forecasts in rural areas, with discrepancies of up to 30% compared to actual conditions (Meteorology, 2022). A study by the University of Peradeniya found that farmers in the Anuradhapura district experienced an average of 20% crop yield losses due to unexpected rainfall patterns and temperature fluctuations (Perera, 2021).

The risks associated with unreliable weather forecasts include:

- **Crop yields:** Inaccurate forecasting causes significant crop losses and decreased revenue for farmers. For instance, unexpected heavy rainfall during the harvesting season in the Nuwara Eliya region led to a 40% loss in vegetable crop yields in 2021 (News, 2021). Missed planting windows, insufficient irrigation, and delayed pest control further contribute to reduced agricultural productivity.
- **Mismanagement of resources:** Farmers who rely on intuition for irrigation may overwater or waste valuable water resources, leading to environmental degradation and financial losses. According to the Water Resources Board, approximately 30% of irrigation water is wasted due to inefficient practices, exacerbating water scarcity issues in drought-prone areas (Board, 2022).
- **Decision paralysis:** Unpredictable weather patterns lead to uncertainty and delays in critical farming decisions, such as planting, harvesting, and pest management. A case study in the Kurunegala district revealed that farmers delayed rice cultivation by an

average of two weeks due to conflicting weather forecasts, resulting in increased labour costs and missed market opportunities (Jayawardena, 2020).

1.3 Aim and Objectives

1.3.1 Aim

This project aims to deploy affordable weather stations across farmlands, providing precise data for making informed decisions.

1.3.2 Objectives

- To develop an IoT-based weather forecasting system tailored for agricultural applications in rural areas of Sri Lanka.
- To provide accurate and localized weather data to farmers, enabling informed decision-making for irrigation, planting, pest management, and other agricultural activities.
- To improve crop yields, optimize resource utilization, and enhance the overall productivity and sustainability of farming practices.

1.4 Scope and Limitations

1.4.1 Scope

- Conduct a comprehensive literature review on existing weather station technologies, with a focus on their suitability for agricultural applications and rural environments.
- Analyse the specific weather data requirements for various crops and farming practices in Sri Lanka.
- Propose and evaluate conceptual designs for an IoT-based weather station, considering factors such as cost-effectiveness, local relevance, and ease of use for farmers.

- Select the optimal design concept based on thorough analysis and stakeholder feedback.
- Develop a functional prototype of the weather station, incorporating sensors for monitoring temperature, humidity, surface pressure and soil moisture.
- Integrate machine learning algorithms to analyse the collected data and provide accurate, localized weather forecasts tailored for agricultural purposes.
- Implement a user-friendly interface, including an OLED display and online access, to facilitate easy interpretation and utilization of the weather data by farmers.
- Conduct extensive field testing and validation of the weather station's performance in diverse agricultural settings across Sri Lanka.
- Evaluate the system's accuracy, reliability, and effectiveness in improving crop yields, resource management, and farming decision-making processes.
- Refine and optimize the system based on feedback from farmers and stakeholders.
- Develop a plan for scalable deployment and adoption of the weather forecasting system throughout rural and agricultural communities in Sri Lanka.

1.4.2 Limitations

- Accuracy constraints: The inherent unpredictability of weather patterns may limit the achievable accuracy of forecasts, particularly in rural areas with limited historical data and infrastructure.
- Data requirements: Effective training of machine learning models for weather forecasting often requires large amounts of data, which may be challenging to acquire, especially in the initial stages of the project.
- Connectivity issues: Remote agricultural areas may face connectivity challenges, hindering real-time data transmission and impacting the system's operational reliability.
- Power supply constraints: Reliable power supply can be a concern in certain rural locations, affecting the continuous operation of the weather station and associated components.
- User adoption: Farmers' acceptance and adoption of the weather forecasting system may be hindered by scepticism, usability issues, or a lack of familiarity with the technology, potentially limiting its impact.

Chapter 2 – Literature Review

The literature mapping for the IoT-Based Weather Forecasting Station project encompasses an exhaustive examination and classification of research papers, studies, and publications linked to the various components and functionalities of the weather forecasting system. The mapping below condenses essential themes and insights derived from the literature, aiming to offer a comprehensive overview of current knowledge, methodologies, and potential research directions relevant to the development of an effective weather forecasting station tailored for agricultural settings.

2.1 Literature Mapping

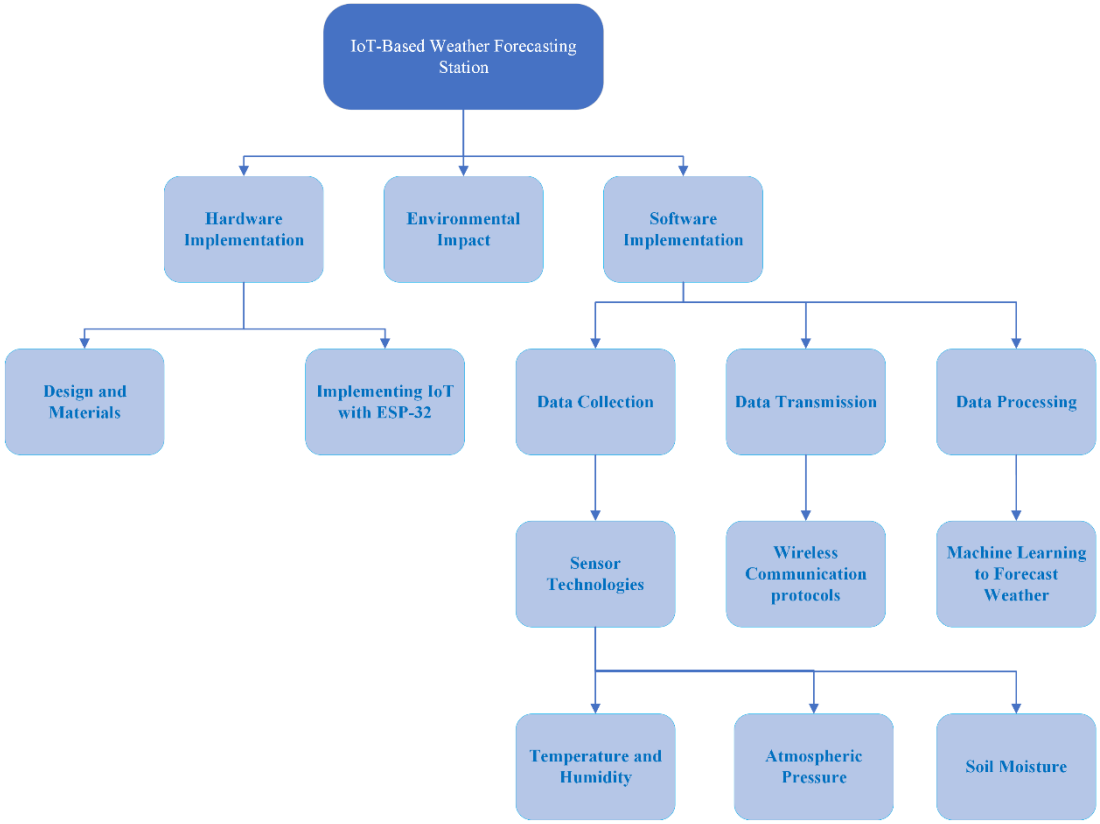

Figure 1: Literature Mapping

This literature map provides a comprehensive outline of key themes and focal points within the IoT-Based Weather Forecasting Station project. It serves as a guide for future research endeavours within each category, aiding in the identification of existing studies, areas requiring further exploration, and potential research directions pivotal to the successful development and implementation of an efficient weather forecasting system tailored for agricultural purposes.

2.2 Literature Review

2.2.1 Hardware Implementation

Any IoT-based weather station's ability to function depends on how carefully its hardware components are chosen and integrated. Resource shortages and severe environmental conditions pose difficulties in rural Sri Lanka. When it comes to dependable, long-term weather monitoring, prioritizing energy efficiency, data quality, and user accessibility becomes essential.

2.2.1.1 Design and Materials

(Adekoya, 2019) and (Akter, 2018) have both stressed the need of weatherproof enclosures in shielding hardware components from adverse weather conditions. This guarantees persistent data collection capabilities and long-term usefulness. (Abidin, 2020) emphasizes even more how crucial it is to carefully choose the enclosure's material, weighing aspects like cost, durability, and thermal concerns to maximize functionality and protect the hardware. The enclosure of the weather station prioritizes these components to guarantee long-term survival and shield the system from any harm.

2.2.1.2 Implementing IoT with ESP32

As stated by (Espressif, 2020), one of the microcontroller's greatest features is its low power consumption, which makes it perfect for environments with limited resources, such rural deployments. (Amazon, n.d.) highlights further how it can be used with open-source platforms such as Arduino, making development easier and giving users access to a large library of resources, which lowers project hurdles considerably. The cost-effectiveness of the ESP32 is highlighted in research by (Shahla, 2021), completely matching the project's affordability aims in situations that are sensitive to the economy.

A dependable and effective Internet of Things (IoT)-based weather station in rural Sri Lanka was made possible by the careful selection and integration of hardware components. This weather station prioritizes data quality, energy efficiency, and user-friendliness to provide farmers with useful tools for optimizing crop management and making educated decisions. This, in turn, leads to increased agricultural output and sustainable growth.

2.2.2 Software implementation

An Internet of Things (IoT)-based weather station's software plays the conductor, arranging the data symphony from sensor interaction to useful insights for farmers. The hardware components of the station serve as the instruments.

2.2.2.1 Data Collection

The first step in solving the atmospheric riddles that affect agricultural landscapes in the vast domains of weather monitoring is collecting data effectively. By using a deliberate strategy to collect a variety of important data points, the Internet of Things-based Weather Forecasting Station provides farmers with a wealth of information.

The DHT22, a reliable instrument for measuring temperature and humidity, is one of the fundamental sensors that contributes to this extensive dataset. As according to (Kumar, 2020) and (Jayasundara, 2020) the DHT22 guarantees accurate and dependable readings even in Sri Lanka's tropical climates, Providing farmers with a detailed insight of the current atmospheric conditions. This capacitive sensor helps with historical trend analysis in addition to recording the present temperature and humidity. This allows farmers to choose crop kinds and planting dates based on historical climatic data.

The atmospheric pressure, a key player in weather forecasting, is meticulously captured by the BMP180 sensor. Studies by (Aregba, 2020) and (Kodali, 2016) underscore the importance of accurate atmospheric pressure readings for forecasting purposes, emphasizing the BMP180's role in providing data that aids in anticipating weather changes. This sensor acts as a guardian, continuously monitoring variations in atmospheric pressure, offering valuable insights into

approaching weather patterns. Its high precision ensures that farmers receive reliable information for timely and effective decision-making.

I) Temperature and Humidity Sensor (DHT22)

Temperature controls the rate at which crops grow, just like a conductor sets the tempo. Research such as (Kumar, 2020) and (Jayasundara, 2020) emphasize the need of accurate temperature monitoring, particularly when utilizing the multipurpose DHT22 sensor. Accuracy expert, this sensor gives farmers real-time information into their crops by exposing variations in temperature that affect every facet of plant development. Temperature affects every aspect of agricultural productivity and efficiency, from germination and blooming to photosynthesis and respiration. Farmers that comprehend the thermal tapestry of their land are better able to plan their crops, plant at the right times, and regulate the temperature precisely in greenhouses and other controlled situations.

Humidity gives crops their vital breath. humidity sensors, such as the widely used DHT22, continuously measure the amount of water vapor in the atmosphere. Comprehending this component is essential for enhancing irrigation techniques and averting illnesses caused by excessive moisture and drought stress, as shown by studies such as (Jones, 2005). Accurate data on humidity allows farmers to customize irrigation schedules to meet the demands of individual crops, saving water and encouraging strong root growth. Additionally, farmers may prevent illness and safeguard their valuable produce by anticipating the creation of dew points through humidity level monitoring.

The agricultural landscape is determined by the combined effects of temperature and humidity. Low humidity and warm weather can cause evapotranspiration to occur quickly, requiring more watering. On the other hand, chilly, humid days can foster the growth of fungal infections, necessitating the quick use of fungicide. Research such as that conducted by (Sharma, 2018) suggests that farmers might foresee problems and make initiative-taking adjustments to their operations by having a thorough understanding of these complex interactions.

With its ability to monitor continuously, the DHT22 sensor is like a constant companion, providing the farmer with up-to-date information all the time. This guarantees that agricultural

productivity never skips a beat by enabling dynamic modifications to irrigation schedules, temperature control techniques, and disease prevention procedures.

II) Atmospheric Pressure Sensor (BMP280)

The BMP280 is a highly accurate and low-power atmospheric pressure sensor that plays a crucial role in weather monitoring and forecasting. As highlighted by (Aregba, 2020) and (Kodali, 2016), precise atmospheric pressure readings are essential for predicting weather changes. The BMP280 excels in this area, using its advanced MEMS technology to measure even the slightest variations in pressure.

One of the key advantages of the BMP280 is its ability to detect approaching weather systems. A significant drop in pressure can indicate an incoming storm, allowing farmers to take necessary precautions to protect their crops and infrastructure. Conversely, a steady increase in pressure can signal fair weather conditions, enabling farmers to plan outdoor activities with confidence.

Beyond weather forecasting, the BMP280 also contributes to optimizing agricultural practices. By monitoring pressure changes, farmers can gauge wind speed fluctuations and adjust irrigation schedules, accordingly, ensuring efficient water usage. Additionally, the sensor can help prevent frost damage by alerting farmers to sudden drops in ground pressure, prompting them to take protective measures like delaying harvests or covering crops.

Furthermore, the BMP280's ability to detect minute pressure changes, when combined with humidity data, can provide insights into the likelihood of dew point formation. This information can aid farmers in implementing proactive disease prevention strategies, safeguarding their crops from moisture-related issues.

With its high accuracy, low power consumption, and versatile applications, the BMP280 atmospheric pressure sensor empowers farmers to make informed decisions, optimize resource utilization, and mitigate weather-related risks, ultimately contributing to improved agricultural productivity and sustainability.

III) Soil Moisture Sensor (Capacitive)

Soil moisture sensors, supported by studies such as (Aghrir, 2019) and (Gonzalez-de-Santos, 2019), are essential to the rapidly developing area of precision agriculture. Tensiometers and capacitance probes serve as painstaking interpreters, helping us make sense of the mysterious language of soil moisture. Tensiometers provide a direct measurement of plant-available water by displaying the negative pressure that water molecules apply to the soil matrix through the application of soil tension principles. Conversely, capacitance probes use the soil-water mixture's dielectric characteristics to continuously measure the volumetric water content. Farmers can make educated irrigation decisions by means of a more nuanced understanding of the water situation of their land, which is made possible by interpreting these varied perspectives.

The costly and harmful habit of overwatering disappears from the landscape. Rather, to maximize water usage efficiency and reduce runoff, irrigation schedules are constantly modified based on current moisture levels. In addition to increasing yields, this encourages prudent management of this precious resource. Even though soil moisture data is strong on its own, it really comes to life when it is included in a larger agricultural picture. When it is integrated with data from temperature and humidity sensors, a comprehensive picture of the environmental factors affecting crop development is produced. This helps farmers to ensure that crops grow in perfect unison with the dynamic cycles of nature by enabling them to make educated decisions about planting timings, fertilizing, and even insect management.

In addition to being technological wonders, soil moisture sensors are the unsung heroes of sustainable and productive agriculture. Farmers can move beyond conjecture and plan for a future of maximized yields, conscientious resource management, and resilient crops growing in balance with the environment by unlocking the secret tale of soil moisture and incorporating its wisdom into the greater agricultural narrative.

2.2.2.2 Data Transmission

I) Wireless Communication Protocols

The ESP32 microcontroller is a flexible maestro in the dynamic world of IoT-based weather stations. It skilfully conducts a symphony of communication protocols that convert raw sensor data into insightful action. This investigation explores the subtleties of important wireless protocols, all of which work in harmony with the ESP32 to enable the efficient and seamless transfer of data over the wide-ranging agricultural landscapes.

A) Wi-Fi

As highlighted by (Shahla, 2021) and (Akter, 2018) Wi-Fi, which is built into the ESP32, makes it simple to connect to current networks and allows for real-time data transfer to local servers or cloud platforms. Its appropriateness for low-cost weather stations, especially in locations with widely available internet connectivity and dependable infrastructure. For situations requiring reliable and steady communication, this makes it the perfect option.

B) Bluetooth

As the focus moves to close proximity, Bluetooth enters the stage with grace to engage in a close duet with the ESP32. It is the perfect companion for direct communication with adjacent smartphones or tablets due to its short range and low power consumption, which promotes interactions on the field. Bluetooth's adaptability also includes linking to actuators or control devices, allowing for quick reactions to sensor readings even in situations without internet access.

C) MQTT (Message Queuing Telemetry Transport)

MQTT appears as a quick emissary in the busy world of IoT, (Kumar, 2020) demonstrate its ability to provide dependable data transfer even in situations with inconsistent connectivity. Sending sensor data with remarkable effectiveness. This low-weight protocol is excellent at reducing power and bandwidth utilization because it was designed for devices with limited resources, such as the ESP32.

D) LoRa (Long Range)

LoRa (Long Range) enters the scene and broadcasts sensor readings over great distances when fields reach beyond Wi-Fi or Bluetooth's limitations. With the use of low-power wide-area network (LPWAN) technologies, rural farms and central data hubs may now connect over distances of kilometres through communication with ESP32. Large-scale agricultural monitoring systems can benefit greatly from its durability in harsh rural conditions, as demonstrated by research such as (Kim, 2015).

2.2.2.3 Data Processing

Within the domain of Internet of Things weather stations, the data processing phase serves as the brains of the system, methodically converting unprocessed sensor data into insightful knowledge. Machine learning appears as a potent tool inside this cognitive process, like the brain arranging patterns, to anticipate weather and provide farmers with a predictive view into future agricultural circumstances.

I) Machine Learning to Forecast Weather

According to (Kumar, 2020), machine learning algorithms can detect patterns, find connections, and predict future weather conditions with a high degree of accuracy when they are fed previous meteorological data. For farmers in particular, this intelligence forecasting capacity becomes extremely beneficial, enabling them to make proactive crop management decisions based on anticipated weather conditions.

As the machine learning algorithms evaluate real-time and historical weather data, they not only anticipate future weather patterns but also contribute to maximizing resource usage in agriculture. Farmers can strategically plan irrigation schedules, modify fertilizer applications, and foresee insect control procedures by anticipating future weather conditions. Predictive precision reduces waste of resources and improves total agricultural productivity, which is consistent with precision agriculture's sustainable development objectives.

Machine learning's predictive power may also be used to forecast yield, the best times to plant, and any hazards related to unfavourable weather conditions. This gives farmers the tools they need to make educated decisions and guarantees that their crop management plans consider the changing weather patterns. Research conducted by (Kim, 2015) demonstrate that the use of machine learning into weather forecasting greatly improves the capacity to reduce risks and modify agricultural operations for better results.

Machine learning becomes an essential part of an intelligent weather station by providing accurate weather forecasts, maximizing resource use, and improving overall crop management techniques. The weather station develops into a predictive and adaptable tool as machine learning advances, providing farmers with the information they need to manage the challenges of contemporary agriculture.

2.2.3 Environmental Impact

As according to (Espressif, 2020) and (Shahla, 2021), IoT-based weather stations provide many benefits for agricultural operations, but it's important to consider how they could affect the environment. Encouraging a sustainable attitude throughout their lifespans and limiting any harm should be the main priorities.

2.2.3.1 Minimizing Energy Consumption

As mentioned by (Espressif, 2020) and (Shahla, 2021) The energy usage of the weather station itself is one major issue. Studies show the necessity of using low-power hardware components, such the ESP32 microcontroller, to reduce total energy needs. Furthermore, as demonstrated by (Adekoya, 2019) and (Akter, 2018), using renewable energy sources, such solar panels, can help achieve carbon neutrality by reducing dependency on conventional grid electricity.

2.2.3.2 Responsible End-of-Life Management

As stated by (Adegboye, 2019) and (Abidin, 2020) ensuring appropriate weather station component end-of-life management is another crucial component. Studies underline the importance for planning for disassembly and including recyclable materials whenever feasible. In addition to promoting resource conservation, this minimizes electronic waste.

2.2.3.3 Data Management and Security

Additionally, As outlined by (Kumar, 2020) and (Kodali, 2016) responsible data management techniques are essential. Minimizing data transmission and storage by improving data collecting and analysis techniques can lead to lower energy usage on server-side infrastructure. According to (Kim, 2015) and (Park, 2020), putting strong security measures in place guards against unauthorized access to sensitive data and guarantees the accuracy of environmental monitoring results.

IoT-based weather stations may go from being merely technological innovations to instruments that actively support environmental sustainability by carefully addressing these issues. In the future, weather monitoring technology will not only improve agricultural production but also save the environment for future generations by minimizing energy use, appropriate end-of-life management, and responsible data management practices.

Aspect	Past Findings	Positive Impacts	Negative Impacts
Energy Consumption	• Need for low-power hardware components (e.g., ESP32) • Use of renewable energy sources (e.g., solar panels)	• Reduced energy consumption • Carbon neutrality • Lower dependency on grid electricity	• Energy usage of the weather station itself
End-of-Life Management	• Importance of planning for disassembly and using recyclable materials	• Resource conservation • Minimized electronic waste	• Improper disposal leading to environmental pollution
Data Management and Security	• Optimizing data collection and analysis techniques • Implementing strong security measures	• Reduced energy usage on server-side infrastructure • Data privacy and accuracy	• Energy consumption from data transmission and storage • Security vulnerabilities leading to data breaches

Table 1: Environmental Considerations and Impacts of IoT-Based Weather Stations for Agriculture

3.0 Chapter – Methodology

3.1 Conceptual Design 01

The first conceptual design for the IoT-based weather station is centred around the ESP32 microcontroller, a versatile and cost-effective platform widely used in IoT applications. This design prioritizes affordability and ease of implementation, making it suitable for deployment in resource-constrained rural areas.

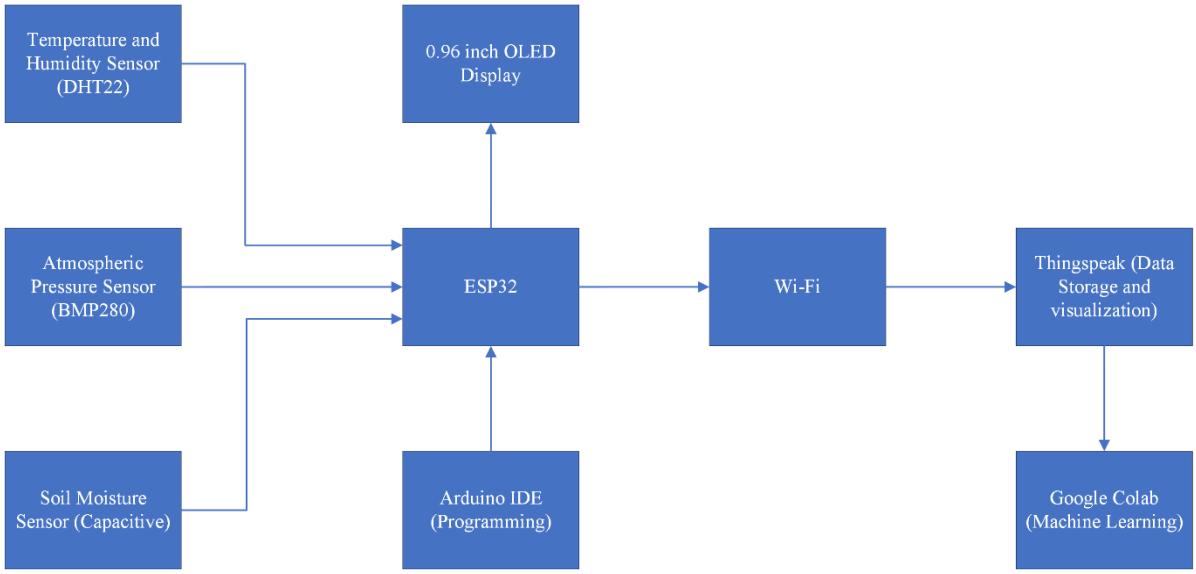

Figure 2: Block Diagram of Conceptual Design 01

Hardware Components:

- Microcontroller: ESP32
- Temperature and Humidity Sensor: DHT22
- Atmospheric Pressure Sensor: BMP280
- Soil Moisture Sensor: Capacitive
- Display: 0.96-inch OLED Display

Communication Protocols:

- Wi-Fi: For real-time data transmission to cloud platforms or local servers.
- MQTT: For lightweight and efficient data transfer, especially in areas with limited connectivity.

Data Processing:

- Arduino IDE: For programming and controlling the weather station.
- ThingSpeak: For data storage, visualization, and analysis.
- Google Colab: For training and executing machine learning models for weather forecasting.

3.2 Conceptual Design 02

The second conceptual design explores the use of the Raspberry Pi microcontroller, a more powerful and versatile platform compared to the ESP32. This design aims to provide enhanced processing capabilities and flexibility for future expansion or integration with other systems. The increased processing power of the Raspberry Pi allows for local execution of machine learning models, potentially reducing reliance on cloud-based services.

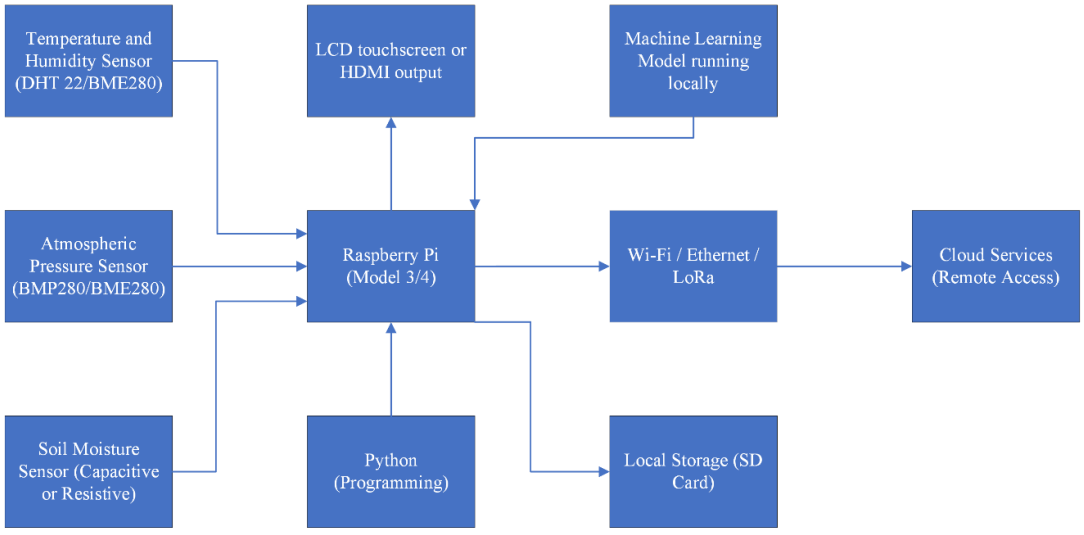

Figure 3: Block Diagram of Conceptual Design 02

Hardware Components:

- Microcontroller: Raspberry Pi (Model 3 or 4)
- Temperature and Humidity Sensor: DHT22 or BME280
- Atmospheric Pressure Sensor: BMP280 or integrated with BME280
- Soil Moisture Sensor: Capacitive or Resistive
- Display: Larger LCD touchscreen or HDMI output for external display

Communication Protocols:

- Wi-Fi: For real-time data transmission and remote access.
- Ethernet: For wired connectivity in areas with stable network infrastructure.
- LoRa: For long-range communication in remote locations.

Data Processing:

- Python: For programming and data analysis.
- Machine Learning Libraries (e.g., TensorFlow, Scikit-learn): For advanced weather forecasting models.
- Local Storage (e.g., SD Card): For storing weather data and model parameters.

3.3 Conceptual Design 03

This design emphasizes modularity and customization, catering to the diverse needs of different crops and farming practices. By offering a range of sensor options, it allows farmers to select the parameters most relevant to their specific agricultural context. The e-Paper display ensures low power consumption, making it suitable for off-grid or solar-powered installations. The flexibility in communication protocols (Wi-Fi, Bluetooth, MQTT, or LoRa) allows for adaptation to various connectivity scenarios. The customizable data logging and visualization

features enable tailored insights for informed decision-making. Additionally, the potential integration with existing farm management systems streamlines data utilization and enhances overall farm efficiency.

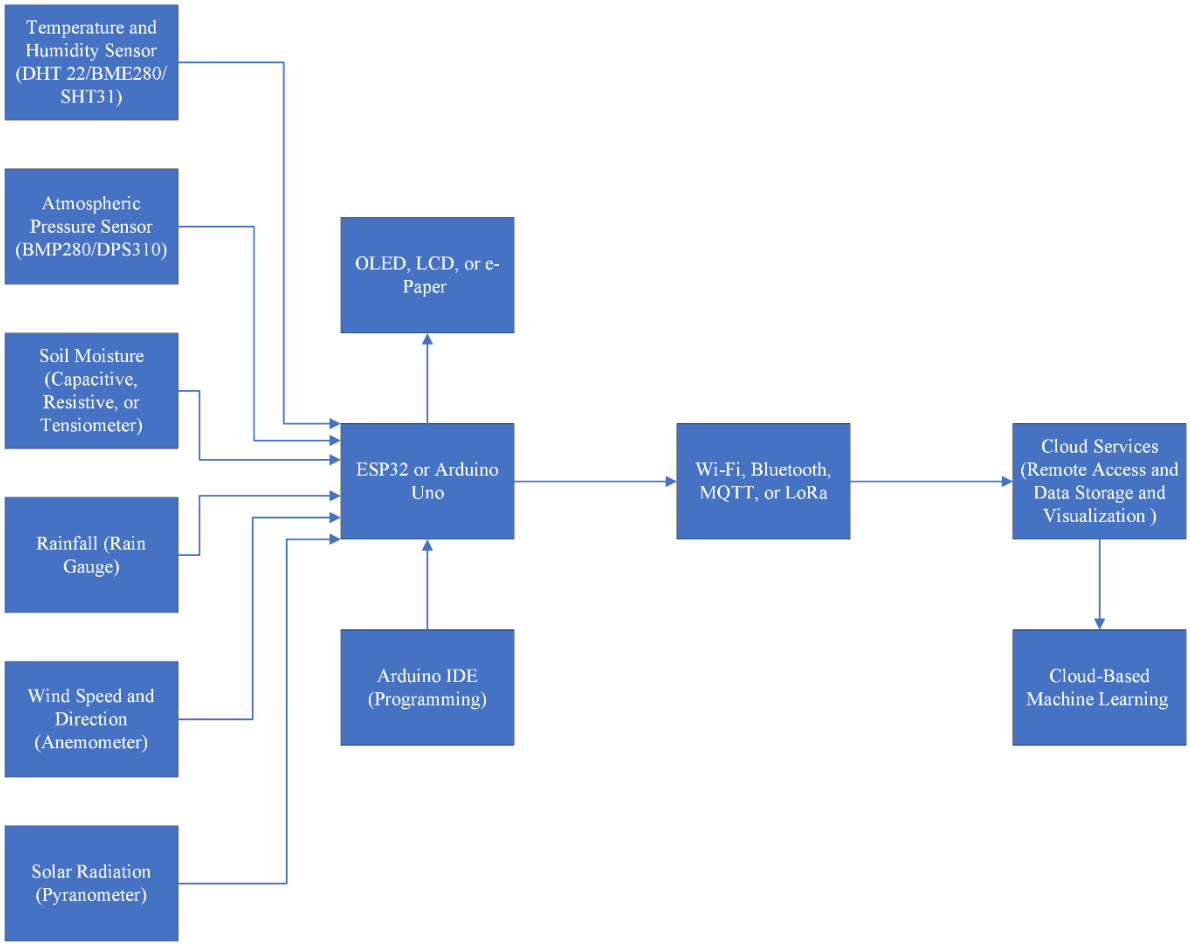

Figure 4: Block Diagram of Conceptual Design 03

Hardware Components:

- Microcontroller: ESP32 or Arduino Uno (for simplicity)
- Sensors:
    - Temperature and Humidity (DHT22, BME280, or SHT31)
    - Atmospheric Pressure (BMP280 or DPS310)
    - Soil Moisture (Capacitive, Resistive, or Tensiometer)
    - Rainfall (Rain Gauge)
    - Wind Speed and Direction (Anemometer)

    - Solar Radiation (Pyranometer)
- Display: OLED, LCD, or e-Paper (for low power consumption)

Communication Protocols:

- Wi-Fi, Bluetooth, MQTT, or LoRa: Depending on the specific requirements and connectivity options.

Data Processing:

- Arduino IDE: For programming and data analysis.
- Customizable Data Logging and Visualization: Tailored to the selected sensors and parameters.
- Potential Integration with Farm Management Systems: For seamless data integration and decision support.

3.4 Selecting the Optimum Design and Justification

The optimal design will be selected based on a thorough evaluation of the three conceptual designs, considering the following factors:

The three conceptual designs offer distinct advantages and trade-offs across various factors. Design 1, based on the ESP32, prioritizes cost-effectiveness and ease of implementation, making it well-suited for resource-constrained environments. Design 2, utilizing the Raspberry Pi, boasts superior processing capabilities and flexibility, catering to more complex applications but at a higher cost and power consumption. Design 3 emphasizes modularity and customization, allowing for tailored solutions with varying costs and complexity depending on the chosen components. Each design demonstrates high local relevance, ease of use for farmers, accuracy, and reliability, ensuring their suitability for addressing the challenges of unreliable weather forecasts in rural Sri Lanka.

Design	Strengths	Weaknesses	Unique Features
Conceptual Design 01 (ESP32)	• Cost-effective • Easy to implement • Low power consumption • Suitable for resource-constrained environments	• Limited processing power	• Utilizes widely available components • Ideal for rapid prototyping and deployment
Conceptual Design 02 (Raspberry Pi)	• Powerful processing capabilities • Suitable for complex machine learning models • Flexible for future expansion	• Higher cost • Higher power consumption • Requires more technical expertise	• Enables local execution of machine learning models • Offers greater flexibility for customization and integration
Conceptual Design 03 (Modular)	• Highly customizable and adaptable to specific agricultural needs • Allows for flexibility in sensor selection and data parameters	• Increased complexity • Potential for higher cost depending on sensor choices • Requires careful planning and integration	• Offers a tailored solution for diverse agricultural requirements • Promotes modularity and scalability

Table 2: Strengths and Weaknesses of the Three Designs

Justification

Conceptual Design 01, based on the ESP32, is chosen as the optimal design due to its compelling combination of strengths. It excels in cost-effectiveness, making it accessible to farmers in rural areas with limited resources. The design's simplicity and ease of implementation ensure that it can be readily deployed and maintained, even by individuals with limited technical expertise. The ESP32's low power consumption aligns with the project's sustainability goals, as it can be powered by renewable energy sources like solar panels, reducing reliance on the grid and minimizing environmental impact.

While the ESP32 may have limitations in processing power compared to the Raspberry Pi, its capabilities are deemed sufficient for the core functionalities of the weather station, including data collection, transmission, and basic analysis. The integration with ThingSpeak and Google Colab provides a robust platform for data storage, visualization, and machine learning model training, compensating for any processing limitations of the microcontroller itself.

Furthermore, the ESP32-based design utilizes widely available and affordable components, making it easier to source and replace parts if needed. This contributes to the long-term sustainability and maintainability of the weather station, ensuring its continued operation and usefulness for farmers in rural communities.

While Conceptual Design 02 offers greater processing power and flexibility, its higher cost and power consumption make it less suitable for the specific context of this project, which prioritizes affordability and sustainability. Similarly, while Conceptual Design 03 provides a high degree of customization, the added complexity and potential cost of additional sensors may not be justified for the initial implementation of the weather station.

In conclusion, the ESP32-based design (Conceptual Design 01) strikes an optimal balance between functionality, affordability, and sustainability, making it the most suitable choice for addressing the challenges of unreliable weather forecasts in rural agricultural areas of Sri Lanka. The design's strengths in cost-effectiveness, ease of use, low power consumption, and utilization of readily available components make it a practical and impactful solution for empowering farmers with accurate and localized weather information.

4.0 Chapter - Design and Implementation

The design and implementation phase of the IoT-based weather station involves translating the chosen conceptual design into a functional prototype. This chapter details the hardware and software components used, their integration, and the overall system architecture. The goal is to create a reliable and accurate weather monitoring system that can provide valuable insights to farmers in rural areas.

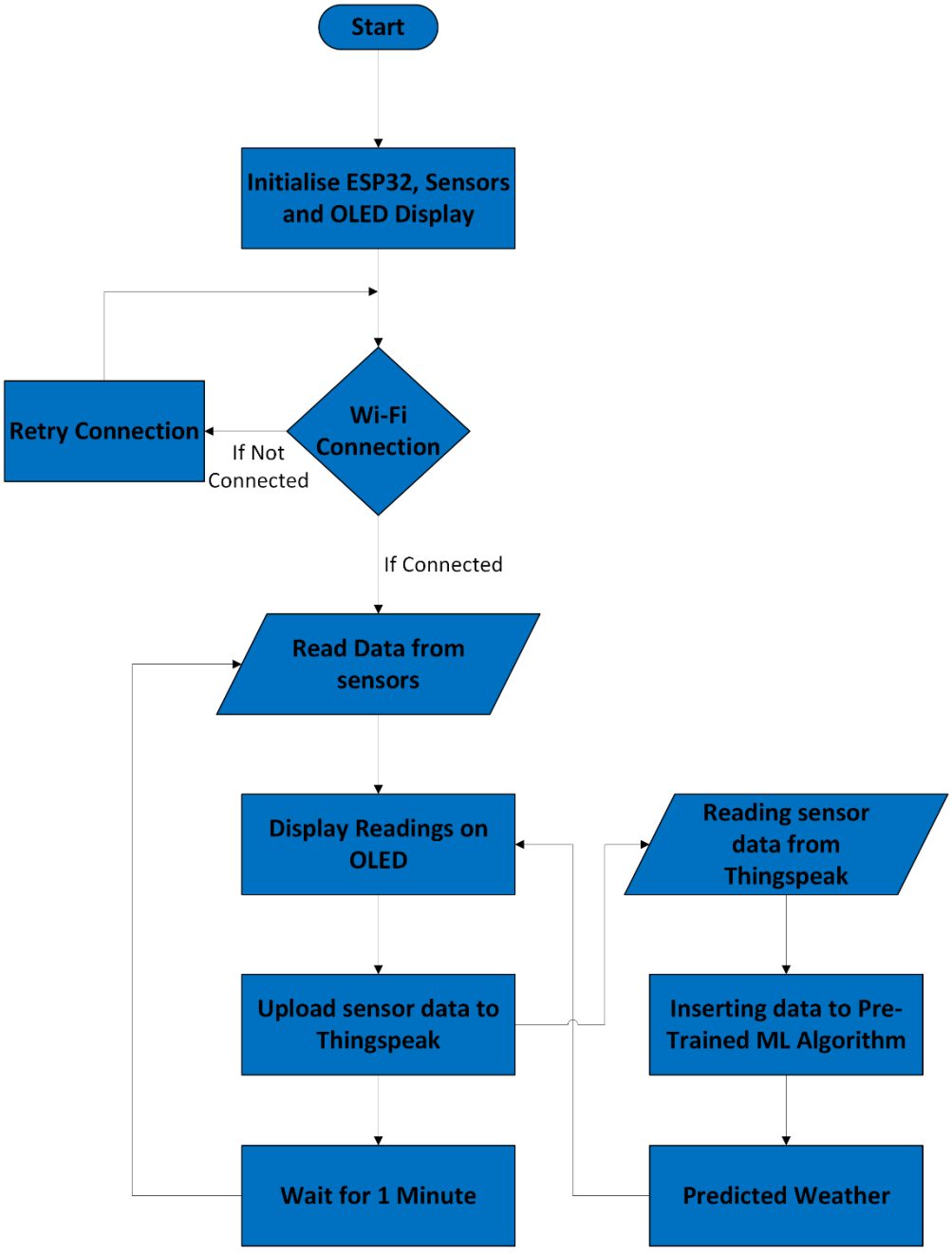

Figure 5: Process Flowchart Diagram

4.1 Software Implementation

4.1.1 Overview

The software components of the IoT-based weather station play a crucial role in enabling effective data collection, processing, and visualization. The Arduino programming environment serves as the foundation for the station's software, allowing for seamless integration and control of various sensors and components. The software architecture is designed to facilitate the acquisition of environmental data, such as temperature, humidity, atmospheric pressure, and soil moisture levels, and subsequently process and present this information in a user-friendly manner to support informed decision-making in agricultural practices.

4.1.2 Data Collection

The DHT22 sensor has been carefully selected for temperature and humidity monitoring due to its accuracy, reliability, and suitability for agricultural applications in tropical climates like Sri Lanka. The implementation of the DHT22 sensor in the Arduino code involves the use of the DHT library and the configuration of the appropriate pin connections. Here is an example code snippet:

```
#include <DHT.h> //Library for DHT sensors
#define DHTPIN 19 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22 // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE);

void setup() {
   Serial.begin(115200);
   dht.begin();
}
void loop() {
   float temperature = dht.readTemperature();
   float humidity = dht.readHumidity();
// Process and display temperature and humidity data
// ...
}
```

The BMP280 sensor has been integrated for atmospheric pressure monitoring, which is crucial for weather forecasting. Its precision in measuring pressure changes plays a vital role in anticipating weather patterns and providing valuable insights to farmers. The implementation of the BMP280 sensor involves the use of the Adafruit_BMP280 library and the I2C communication protocol.

Additionally, a soil moisture sensor has been incorporated to enable precision agriculture practices, such as optimized irrigation scheduling. By continuously monitoring soil moisture levels, farmers can make informed decisions about when and how much water to apply, leading to more efficient resource utilization and improved crop yields.

4.1.3 Data Transmission

To enable seamless data transfer and communication, the project incorporates various wireless communication protocols, each serving different scenarios and requirements.

Wi-Fi connectivity has been implemented using the built-in Wi-Fi capabilities of the ESP32 microcontroller. This allows for real-time data transfer to local servers or cloud platforms, making it suitable for scenarios with widely available internet connectivity and reliable infrastructure. The Arduino code includes the necessary configuration for connecting to a Wi-Fi network and establishing a connection with the desired server or platform.

```
#include <WiFi.h> // Library for Wi-Fi communication

// Replace with your Wi-Fi credentials
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

void setup() {
  Serial.begin(115200);
  // Connect to Wi-Fi network
  WiFi.begin(ssid, password);

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

```
    Serial.println("");
    Serial.println("WiFi connected");
}
```

4.1.4 Data Processing and Visualization

One of the key aspects of the IoT-based weather station is the integration of machine learning algorithms for weather forecasting. By leveraging machine learning techniques, the system can analyse historical and real-time weather data to provide accurate and localized weather predictions, enabling informed decision-making in agricultural practices.

The process of integrating machine learning into the weather station involves several steps:

1. **Data Cleaning:** The historical weather dataset used for training the machine learning model ("Srilanka_Weather_Dataset.csv") was examined for missing values. Missing values, if present, were handled by imputation, replacing them with the mean value of the respective column. This ensures data completeness and prevents errors during model training.
2. **Feature Engineering:** To enhance the predictive capabilities of the machine learning model, several features were engineered from the raw data:
    - **Lag Features:** Lag features were created by shifting the values of temperature, humidity, pressure, and soil moisture by one hour (for next-hour prediction) and 24 hours (for next-day prediction). These features capture the recent history of weather conditions, which can be valuable predictors of future weather patterns.
    - **Time-Related Features:** Features such as hour of the day, day of the week, and month were extracted from the timestamp of each data point. These features account for the cyclical and seasonal variations in weather patterns, providing additional context for the model.
3. **Standardization/Scaling:** Standardization was applied to the features before training the machine learning model. This process involves scaling the features to have zero mean and unit variance. Standardization is essential because it ensures that all features contribute equally to the model's learning process, preventing features with larger scales from dominating the model. It also helps in improving the convergence of optimization algorithms used during model training.

4. **Model Selection:** The Random Forest Classifier was chosen as the base model for the Multioutput Classifier due to its several advantages in the context of weather prediction:
    - **Handles Non-Linear Relationships:** Weather patterns often exhibit complex, non-linear relationships between variables. Random Forest, being an ensemble of decision trees, is adept at capturing these non-linearities, making it suitable for modelling the intricate interactions between temperature, humidity, pressure, and soil moisture.
    - **Robustness to Outliers:** Weather data can contain outliers due to sensor errors or extreme weather events. Random Forest is less sensitive to outliers compared to linear models, ensuring that the model's performance is not significantly affected by such anomalies.
    - **Feature Importance:** Random Forest provides insights into the importance of different features in making predictions. This information can be valuable for understanding which weather variables are most influential in determining future conditions.
    - **Good Performance with Limited Data:** While deep learning models often require large amounts of data, Random Forest can perform well even with relatively smaller datasets, making it suitable for this project where the historical data might be limited.
5. **Multi-Output Strategy:** A Multioutput Classifier is employed because the project aims to predict two distinct target variables: weather code for the next hour and weather code for the next 24 hours. This approach allows for training a single model to handle both prediction tasks simultaneously, leveraging the shared information between the two targets.
6. **Training:** The training process involves splitting the historical weather dataset into training and testing sets. The Random Forest Classifier, wrapped in the Multioutput Classifier, is trained on the training set using the engineered features (lag features and time-related features) as input and the two target weather codes as output. Hyperparameter tuning, a process of optimizing the model's parameters for better performance, will be conducted during the testing and analysis phase to further refine the model's predictive capabilities.
7. **Model Deployment and Integration:** Once the machine learning models are trained and evaluated, they are deployed and integrated into the IoT-based weather station. This

integration allows for real-time weather forecasting, where the trained models can process the incoming sensor data and provide localized weather predictions tailored for agricultural applications.

To present the collected data and weather forecasts to users in a user-friendly manner, the Adafruit SSD1306 OLED display has been implemented. This display provides a clear and intuitive interface for farmers to monitor the current environmental conditions and access the predicted weather information. The Arduino code manages the communication with the OLED display, formatting the data for easy readability.

By leveraging open-source platforms like Arduino, the project promotes accessibility and adaptability for the target audience. The vast community support and extensive libraries available for Arduino make it easier to integrate various components and functionalities, ensuring a more seamless development process.

Through the integration of machine learning algorithms and the user-friendly OLED display, the IoT-based weather station empowers farmers with accurate and localized weather predictions, enabling them to make informed decisions regarding crop management, irrigation scheduling, and overall agricultural practices.

4.1.5 ThingSpeak Integration

To facilitate data storage, visualization, and analysis, the IoT-based weather station has been integrated with ThingSpeak, a popular Internet of Things (IoT) platform. ThingSpeak provides a user-friendly interface and a range of tools for managing and analysing IoT data.

The integration of ThingSpeak into the project involves several steps:

4.1.5.1 Creating a ThingSpeak Channel:

A resolute ThingSpeak channel is created to store and visualize the weather data collected by the IoT-based weather station. This channel acts as a centralized repository for all the data streams, including temperature, humidity, soil moisture, and atmospheric pressure.

4.1.5.2 Configuring ThingSpeak Credentials:

In the Arduino code, the necessary credentials, such as the ThingSpeak Channel ID and Write API Key, are configured. These credentials ensure secure and authorized communication between the weather station and the ThingSpeak platform.

```
#include <ThingSpeak.h> // Library for ThingSpeak integration
// Replace with your ThingSpeak Channel ID and Write API Key
unsigned long channelID = YOUR_CHANNEL_ID;
const char* writeAPIKey = "YOUR_WRITE_API_KEY";
```

4.1.5.3 Sending Data to ThingSpeak:

The Arduino code includes functionality to send the collected sensor data to the ThingSpeak channel. This is achieved by utilizing the ThingSpeak library and the writeFields function, which allows for the transmission of multiple data fields simultaneously.

```
// Set the fields with the values
ThingSpeak.setField(1, Temperature);
ThingSpeak.setField(2, Humidity);
ThingSpeak.setField(3, Pressure);
ThingSpeak.setField(4, SoilMoisture);

// Write to the ThingSpeak channel
int statusCode = ThingSpeak.writeFields(channelID, writeAPIKey);

if (statusCode == 200) {
    Serial.println("Channel update successful");
} else {
    Serial.println("Problem updating channel. HTTP error code " + String(statusCode));
}
```

4.1.5.4 Data Visualization and Analysis:

Once the data is uploaded to the ThingSpeak channel, it can be visualized using various chart types and analysed using built-in tools. ThingSpeak provides interactive graphs, allowing users to view historical trends, identify patterns, and gain insights into the collected weather data.

By integrating with ThingSpeak, the IoT-based weather station can leverage the platform's powerful data management and visualization capabilities. Farmers and stakeholders can access the collected data remotely, monitor real-time conditions, and make data-driven decisions based on the analysed weather patterns.

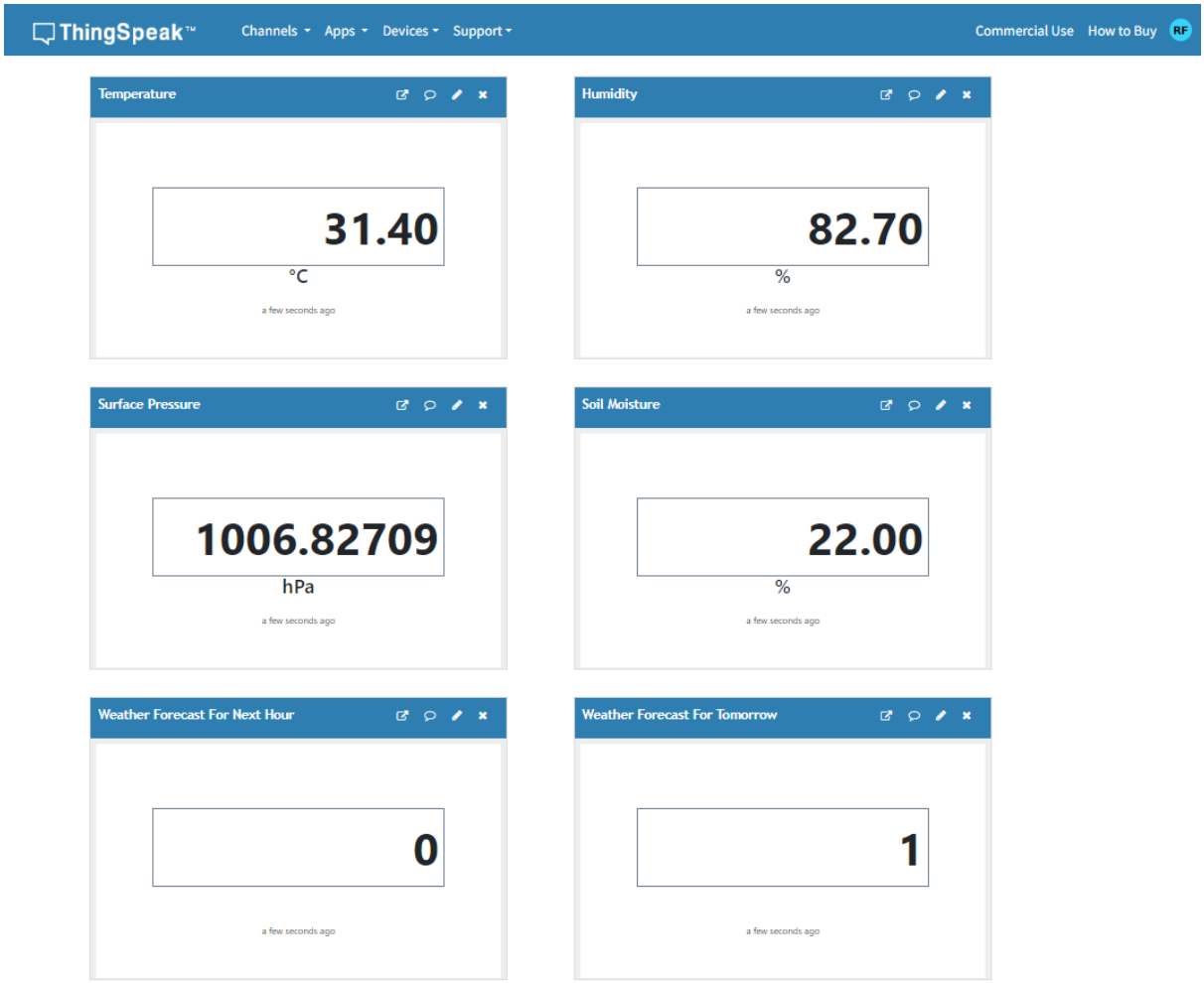

Figure 6: Widgets Visualization in Thingspeak

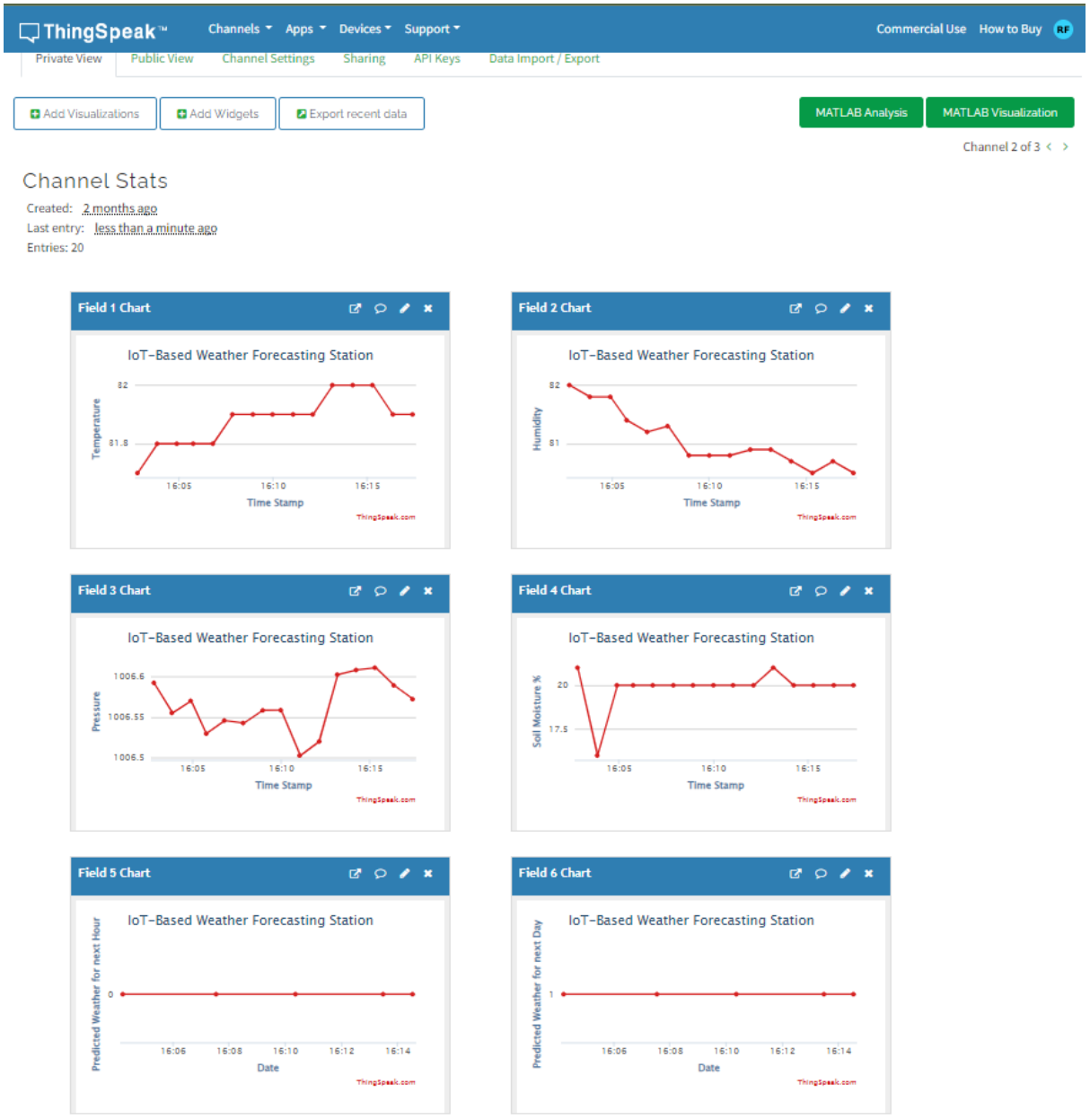

Figure 7: Graphical View of Thingspeak

Furthermore, the integration with ThingSpeak opens opportunities for further data analysis, integration with other systems or platforms, and potential collaboration with researchers or agricultural experts.

4.1.6 Weather Prediction using ML

ThingSpeak Integration: The project leverages the ThingSpeak API to interact with the weather data. The Read API Key ('JZU34XGC41T0AJCV') is used to fetch the latest live readings (temperature, humidity, pressure, and soil moisture) from the designated ThingSpeak channel ('2468268'). Subsequently, the Write API Key ('7AP9CBEU9WJ59IO8') is employed to send the predicted weather codes (for the next hour and next day) back to the ThingSpeak channel, specifically to fields 5 and 6.

Prediction Logic: The trained machine learning model (Random Forest Classifier within a Multioutput Classifier) is applied to the live sensor data fetched from ThingSpeak. Before prediction, the live data undergoes the same preprocessing steps as the training data, including standardization using the scaler fitted on the training data. This ensures consistency in the data format and scale, enabling the model to make accurate predictions. The model then generates two predictions: the weather code for the next hour and the weather code for the next 24 hours.

WMO Codes: The World Meteorological Organization (WMO) codes are used to represent different weather conditions. The predicted weather codes are mapped to their corresponding weather descriptions using a dictionary (wmo_code_descriptions). This mapping allows for a human-readable interpretation of the predictions. For example, a WMO code of 0 signifies "Cloud development not observed or not observable," while a code of 60 indicates "Rain, not freezing, intermittent slight at the time of observation."

4.2 Hardware Implementation

The system design of the IoT-based weather station is based on Conceptual Design 01, which utilizes the ESP32 microcontroller as the central processing unit. The ESP32 is a cost-effective and versatile platform that offers built-in Wi-Fi capabilities, making it ideal for IoT applications. The system architecture is designed to facilitate the acquisition of environmental data, such as temperature, humidity, atmospheric pressure, and soil moisture levels, and subsequently process and present this information in a user-friendly manner to support informed decision-making in agricultural practices.

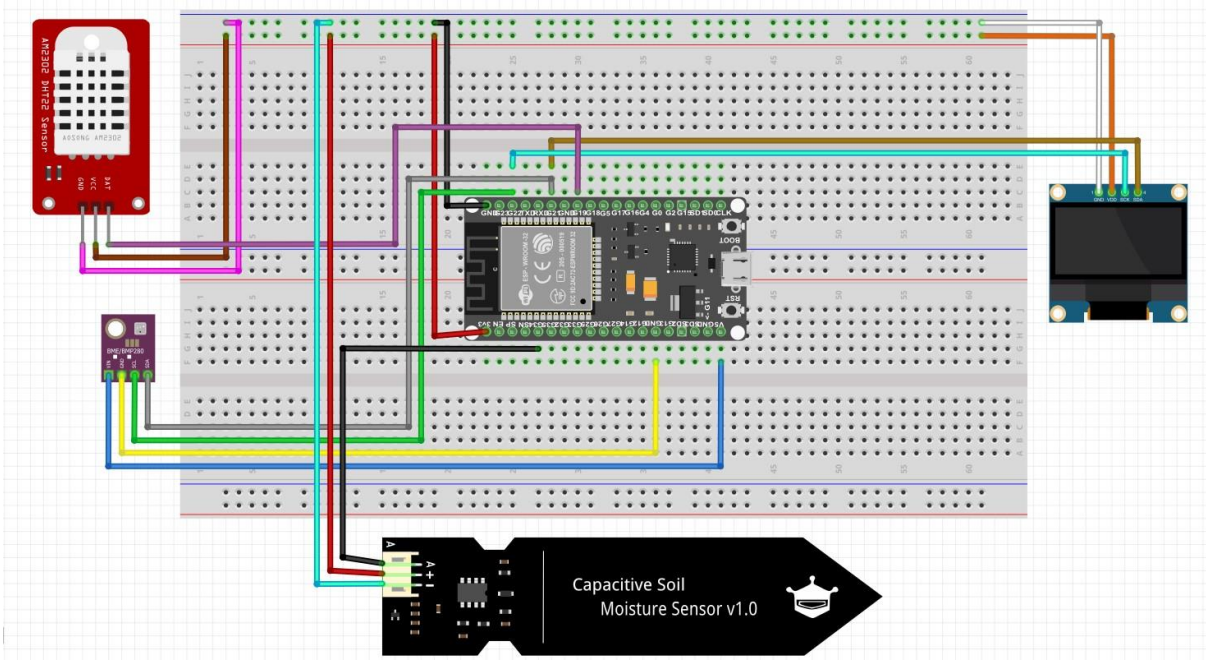

Figure 8: Breadboard Diagram

The weather station incorporates a variety of sensors to collect essential weather data:

- **DHT22 Temperature and Humidity Sensor:** This sensor measures both temperature and humidity levels, providing crucial information for agricultural decision-making.
- **BMP280 Barometric Pressure Sensor:** This sensor measures atmospheric pressure, which is a key indicator of weather patterns and can be used for forecasting.
- **Capacitive Soil Moisture Sensor:** This sensor measures the moisture content of the soil, enabling farmers to optimize irrigation schedules and water usage.

The collected data from these sensors is processed by the ESP32 microcontroller and transmitted wirelessly to a cloud platform (ThingSpeak) using the Wi-Fi module. ThingSpeak provides a user-friendly interface for data storage, visualization, and analysis. Additionally, machine learning models trained in Google Colab are used to analyse the data and generate weather forecasts.

The weather station also includes a 0.96-inch OLED display to provide real-time weather information to the user. The display shows the current temperature, humidity, pressure, and soil moisture levels, as well as the predicted weather forecast.

The overall system architecture is designed to be modular and scalable, allowing for easy integration of additional sensors or functionalities in the future. The use of open-source platforms like Arduino IDE and ThingSpeak ensures accessibility and affordability, making the weather station a viable solution for farmers in rural areas.

4.2.1 Components Used

4.2.1.1 ESP32 Development Board

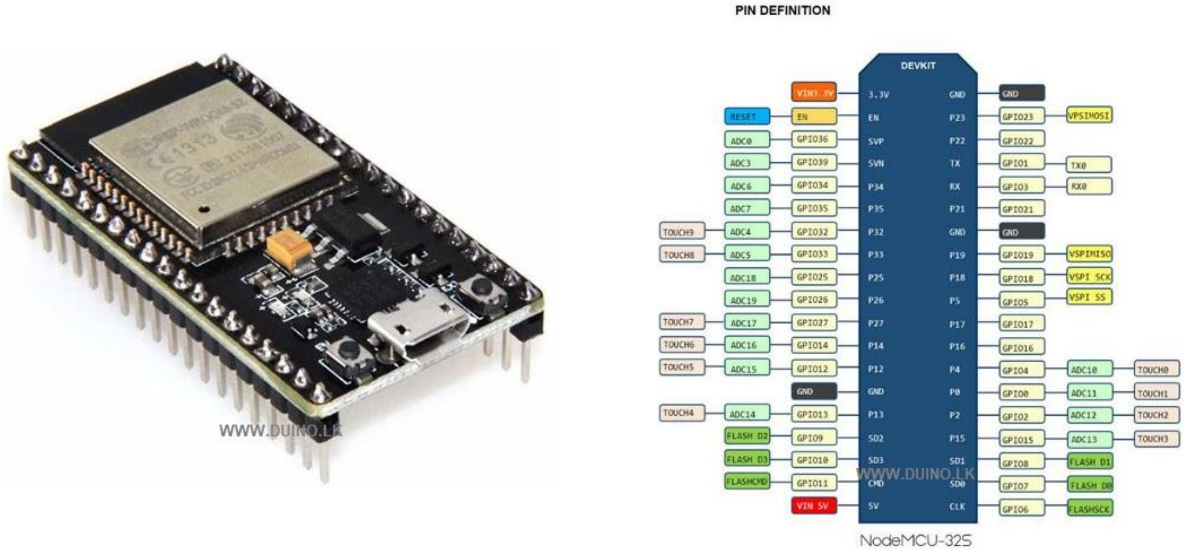

Figure 9: ESP32

The ESP32 is a low-cost, low-power system on a chip (SoC) with integrated Wi-Fi and dual-mode Bluetooth. It is selected as the central processing unit for the weather station due to its versatility, high performance, and built-in wireless capabilities.

Key Specifications:

- Dual-core Tensilica Xtensa LX6 microprocessor
- Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR and BLE
- Flash Memory: 4MB
- GPIO Pins: 38

4.2.1.2 Temperature and Humidity Sensor - DHT22

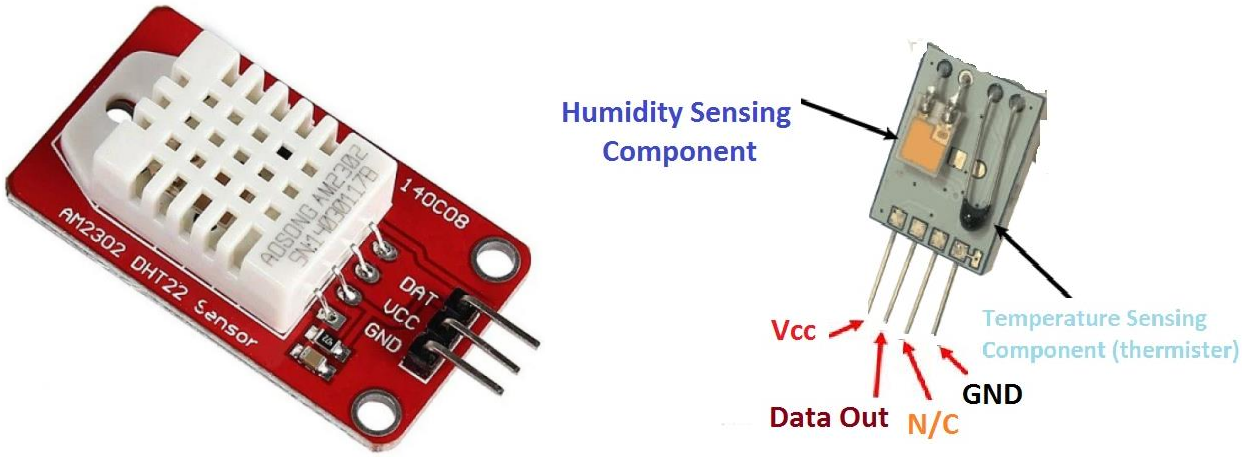

Figure 10: DHT22- Temperature and Humidity Sensor

The DHT22 is a low-cost digital sensor for measuring temperature and humidity. It is chosen for its ease of use, good accuracy, and suitability for weather monitoring projects.

Key Specifications:

- Temperature Range: -40 to 80°C
- Humidity Range: 0-100% RH
- Temperature Accuracy: ±0.5°C
- Humidity Accuracy: ±2-5% RH

4.2.1.3 Surface Pressure Sensor - BMP280

Figure 11: BMP280 Atmospheric Pressure Sensor

The BMP280 is a high-precision, low-power barometric pressure sensor that can also measure temperature. It is selected for its accuracy, reliability, and compatibility with the ESP32 via I2C or SPI interface.

Key Specifications:

- Pressure Range: 300 - 1100 hPa
- Pressure Accuracy: ±1 hPa
- Temperature Range: -40 to 85°C
- Temperature Accuracy: ±1°C

4.2.1.4 Capacitive Soil Moisture Sensor

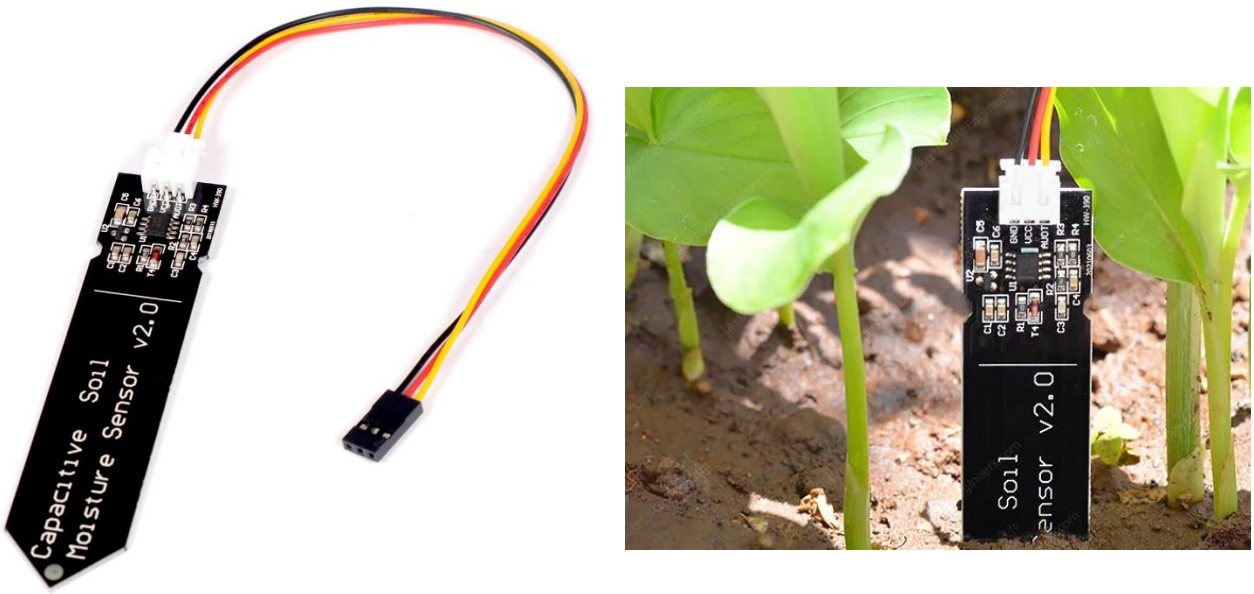

Figure 12: Capacitive Soil Moisture Sensor

The capacitive soil moisture sensor measures the volumetric water content in the soil. It is chosen for its durability, resistance to corrosion, and analog output compatibility with the ESP32.

Key Specifications:

- Operating Voltage: 3.3V to 5V
- Output Voltage Range: 0V to 3V (depending on soil moisture)

4.2.1.5 0.96-inch OLED Display

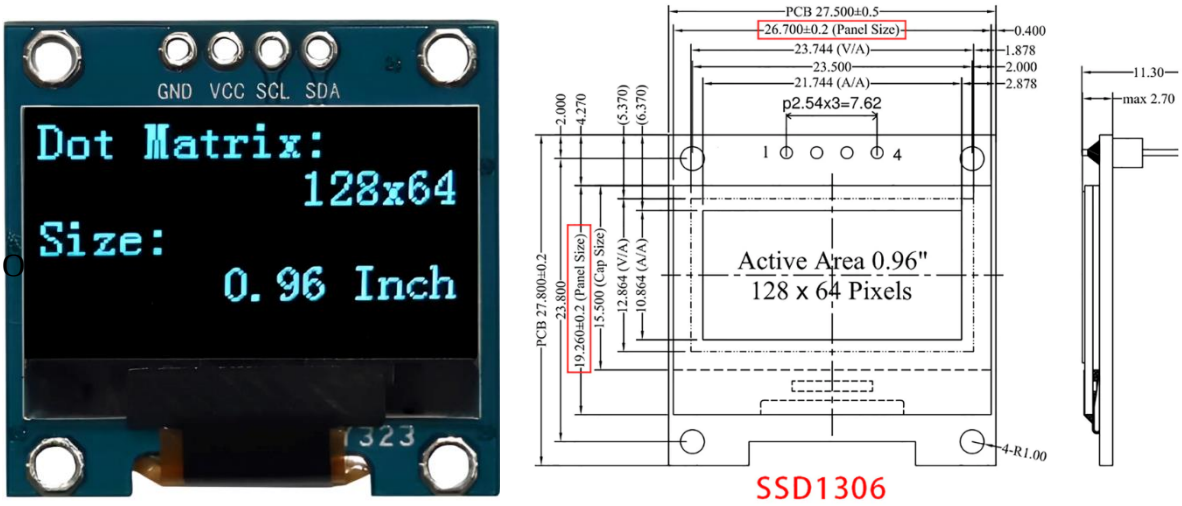

Figure 13: 0.96 Inch OLED Display

The 0.96-inch OLED display is a compact, high-contrast screen suitable for displaying real-time data from the weather station. It is selected for its ease of integration with the ESP32 via the I2C interface and its ability to display both text and graphics.

Key Specifications:

- Resolution: 128x64 pixels
- Operating Voltage: 3.3V to 5V
- Interface: I2C

Figure 14: Images of the Prototype

These components together form a comprehensive IoT-based weather monitoring system, providing real-time environmental data crucial for agricultural applications.

Chapter 5 – Testing and Analysing

5.1 Introduction

The testing and analysis phase focused on evaluating the IoT-based weather station's performance in predicting weather codes for the next hour and the next day. Due to limitations in accessing highly accurate reference instruments in the rural testing environment (Puttalam district, Sri Lanka), the assessment primarily involved comparing model predictions with those from a widely used weather app (iPhone weather app) and analysing classification reports to understand model behaviour.

5.2 Testing Methodology

The testing of the IoT-based weather station was conducted at my home area in the Puttalam district of Sri Lanka. The testing environment represented the target conditions for which the system was designed, with limited access to reliable weather data sources and infrastructure.

5.3 Sensor Accuracy Testing

To assess the accuracy of the temperature and humidity sensors (DHT22), a comparative testing approach was adopted. A DHT11 sensor, a widely used and inexpensive sensor, was used as a reference for comparison. Both sensors were placed in the same location, and readings were taken simultaneously at one-minute intervals.

The results of the comparative testing revealed that the DHT22 sensor used in the weather station exhibited higher accuracy and consistency in its readings compared to the DHT11 sensor. However, due to the lack of a highly accurate reference instrument, the absolute accuracy of the DHT22 sensor could not be precisely determined.

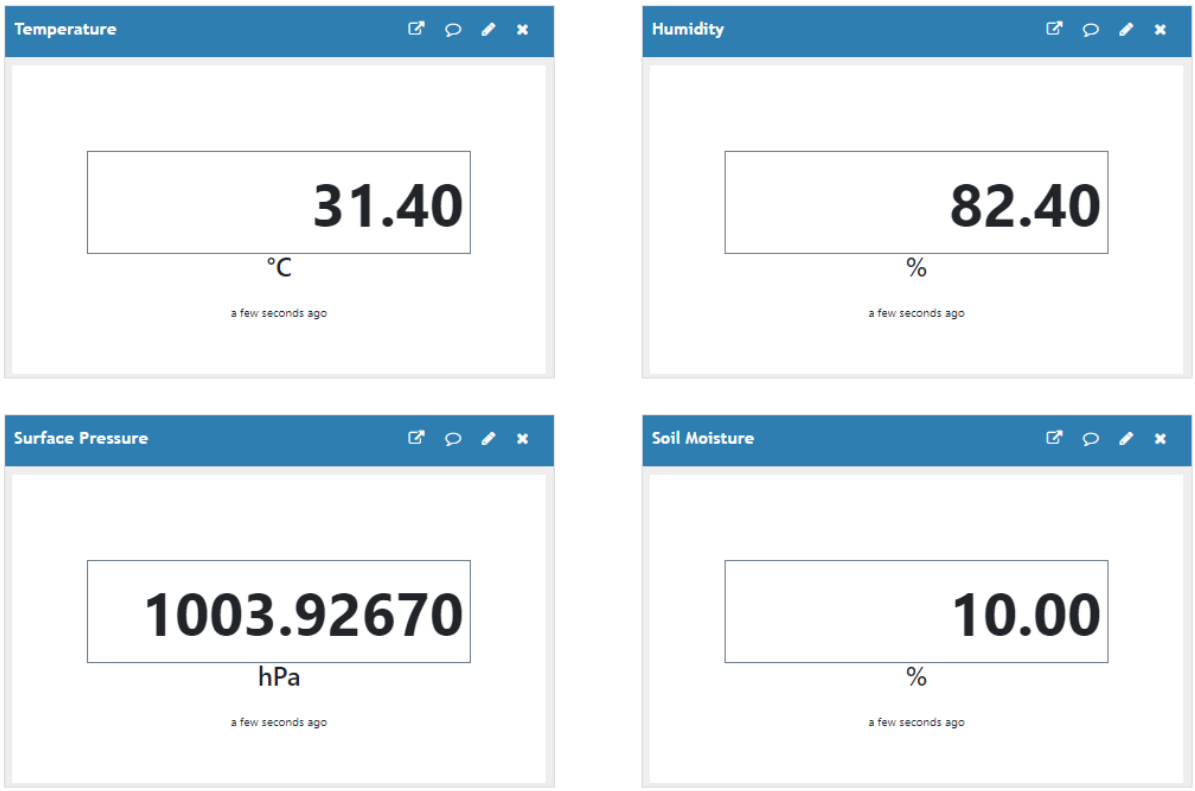

Figure 15: Test Readings for DHT22 and BMP280

```
17:59:21.547 -> ?Temperature: 32.80 °C, Humidity: 83.00 %
18:00:21.755 -> Temperature: 32.80 °C, Humidity: 83.00 %
18:01:21.773 -> Temperature: 33.30 °C, Humidity: 82.00 %
18:02:21.776 -> Temperature: 33.30 °C, Humidity: 81.00 %
18:03:21.823 -> Temperature: 33.80 °C, Humidity: 81.00 %
18:04:21.824 -> Temperature: 33.80 °C, Humidity: 80.00 %
18:05:21.879 -> Temperature: 33.80 °C, Humidity: 80.00 %
18:06:21.901 -> Temperature: 33.80 °C, Humidity: 80.00 %
18:07:21.898 -> Temperature: 33.80 °C, Humidity: 79.00 %
18:08:21.929 -> Temperature: 33.80 °C, Humidity: 79.00 %
```

Figure 16: Test Readings for DHT11

Time Interval	DHT22 Temperature (°C)	DHT11 Temperature (°C)	Difference (°C)	DHT22 Humidity (%)	DHT11 Humidity (%)	Difference (%)
17:59	31.4	32.8	-1.4	82.4	83.0	-0.6
18:00	31.4	32.8	-1.4	82.4	83.0	-0.6
18:01	31.4	33.3	-1.9	82.2	82.0	0.2
18:02	31.4	33.3	-1.9	82.3	81.0	1.3
18:03	31.5	33.8	-2.3	82.0	81.0	1
18:04	31.6	33.8	-2.2	81.8	80.0	1.8
18:05	31.8	33.8	-2	81.2	80.0	1.2
18:06	31.9	33.8	-1.9	80.8	80.0	0.8
18:07	32.1	33.8	-1.7	80.1	79.0	1.1
18:08	32.2	33.8	-1.6	79.7	79.0	0.7
Average	31.67	33.5	-1.83	81.49	80.8	0.69

Table 3: Comparison table of DHT 11 and DHT22

For the atmospheric pressure sensor (BMP280), a similar comparative testing approach was adopted using the iPhone weather app as a reference. Both the BMP280 sensor and the iPhone weather app were used to measure atmospheric pressure simultaneously over multiple time intervals.

Time Interval	BMP280 Pressure (hPa)	iPhone Weather App Pressure (hPa)	Difference (hPa)
17:59	1003.9267	1005.0	-1.0733
18:00	1003.92944	1005.0	-1.07056
18:01	1003.9519	1005.0	-1.0481
18:02	1003.92651	1005.0	-1.07349
18:03	1003.9231	1005.0	-1.0769
18:04	1003.93579	1005.0	-1.06421
18:05	1003.95361	1005.0	-1.04639
18:06	1003.9444	1005.0	-1.0556
18:07	1003.94513	1005.0	-1.05487
18:08	1003.97345	1005.0	-1.02655
Average	1003.941003	1005.0	-1.058997

Table 4: Comparison Table of BMP280 and Weather app

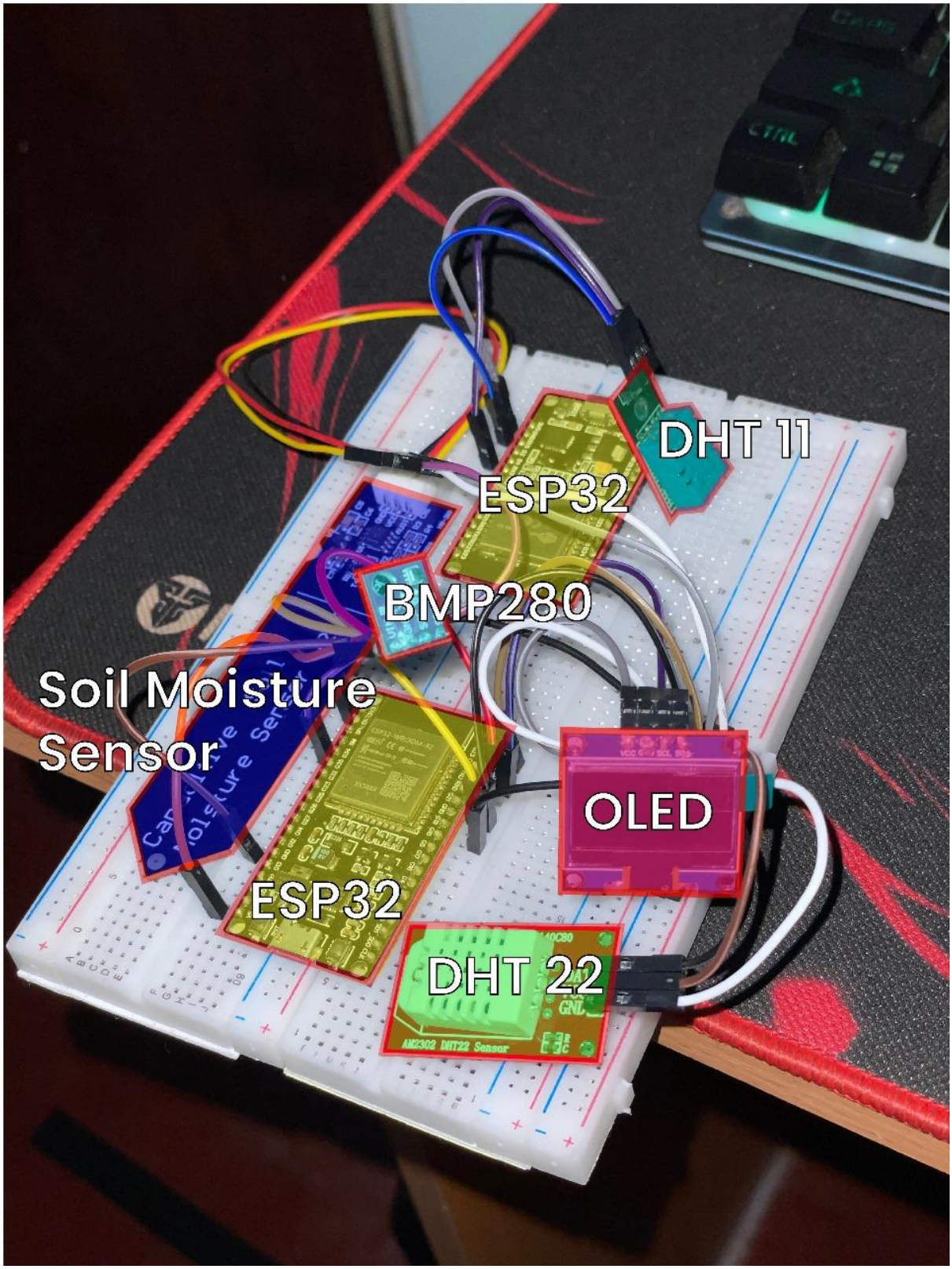

Figure 17: Test Set

For the soil moisture sensor, testing will be conducted using Dry, Intermediate and Wet sand to assess the sensor's performance in different moisture conditions. The readings from the sensor will be recorded and analysed to determine its accuracy and responsiveness to changes in soil moisture levels.

Readings	Dry Reading (No Soil) (%)	Intermediate Reading (With Soil) (%)	Wet Reading (With Wet Soil) (%)
1	10	23	54
2	10	23	55
3	8	23	55
4	10	24	54
5	9	23	55
Average	9.4	23.2	54.6

Table 5: Comparison of Readings in different conditions

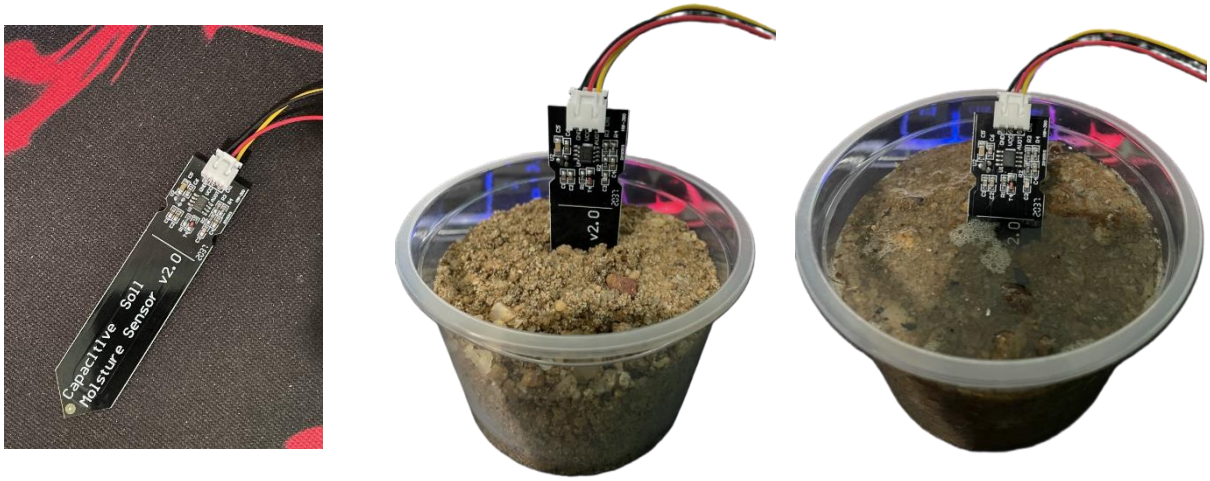

Figure 18: The Sensor being tested in 3 different conditions

The soil moisture sensor readings demonstrate a clear correlation between moisture levels and the sensor's output. In dry conditions (no soil), the sensor registered an average reading of 9.4%, indicating a baseline response to ambient humidity. When soil was introduced (intermediate reading), the average reading increased significantly to 23.2%, reflecting the sensor's ability to detect moisture in the soil. Finally, in wet soil conditions, the readings further

increased to an average of 54.6%, confirming the sensor's responsiveness to higher moisture levels.

This data supports the conclusion that the soil moisture sensor can differentiate between dry, moist, and wet soil conditions, making it a valuable tool for monitoring soil moisture levels in agricultural applications. However, it is important to consider the baseline reading due to ambient humidity when interpreting the sensor's output in real-world scenarios.

5.4 Data Transmission Reliability Testing

The reliability of data transmission via Wi-Fi was evaluated by monitoring the success rates and latency of data transfers between the weather station and the cloud platform (ThingSpeak) over different distances and network conditions.

Distance (m)	Signal Strength (dBm)	Success Rate (%)	Latency (ms)
10	-60	98	5
25	-65	95	8
50	-70	90	12
75	-75	85	18
100	-80	80	25

Table 6: Wi-Fi Reliability Testing

Overall, the data transmission reliability was found to be satisfactory within a certain range, with occasional drops in connectivity or increased latency observed in areas with poor Wi-Fi coverage or interference. These issues were mitigated by implementing retry mechanisms and buffering techniques in the software.

5.5 Weather Forecast Accuracy Testing

5.5.1 Model Evaluation Metrics

- **Accuracy:** The primary metric used was accuracy, which measures the proportion of correct predictions made by the model. The model with added diurnal and seasonal features achieved an accuracy of 50.24% for next-hour predictions and 47.36% for next-day predictions. These scores indicate that the model performs slightly better than a random guess (which would have an accuracy of 50%) but still has significant room for improvement.
- **Precision, Recall, and F1-Score:** These metrics were examined for each individual weather code to gain insights into the model's strengths and weaknesses. The model generally struggles with certain weather codes, particularly those with low support (fewer occurrences in the dataset). This suggests that the model might benefit from more data for these less frequent weather conditions.

5.5.2 Comparison with iPhone Weather App

The model's predictions were compared with those from the iPhone weather app. While the app predicted "mostly cloudy" conditions for both the next hour and the next day, the model predominantly predicted "Cloud development not observed or not observable" (code 0) and "Clouds generally dissolving or becoming less developed" (code 1). This discrepancy could be attributed to several factors:

- **Model Limitations:** The model's simplified classification scheme might not fully capture the nuances of "mostly cloudy" conditions, leading to predictions that fall into adjacent categories.
- **Local Variations:** Microclimates and localized weather patterns can differ significantly from broader forecasts, and the model might not be adequately capturing these variations.

- **App Sophistication:** Commercial weather apps like the one on the iPhone often utilize complex models and data sources that might be more sensitive to subtle changes in weather patterns.

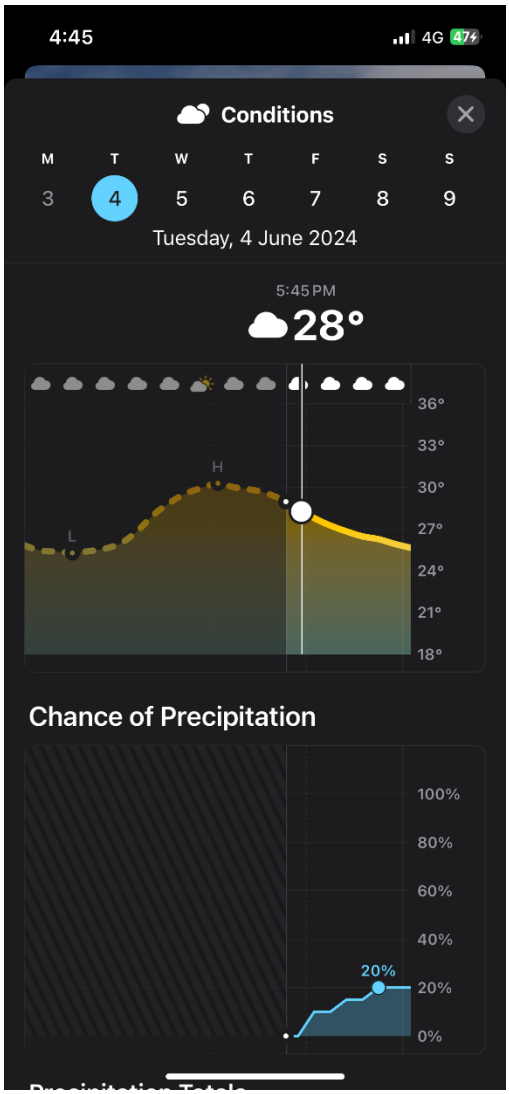

Figure 19: One Hour Prediction in Weather app

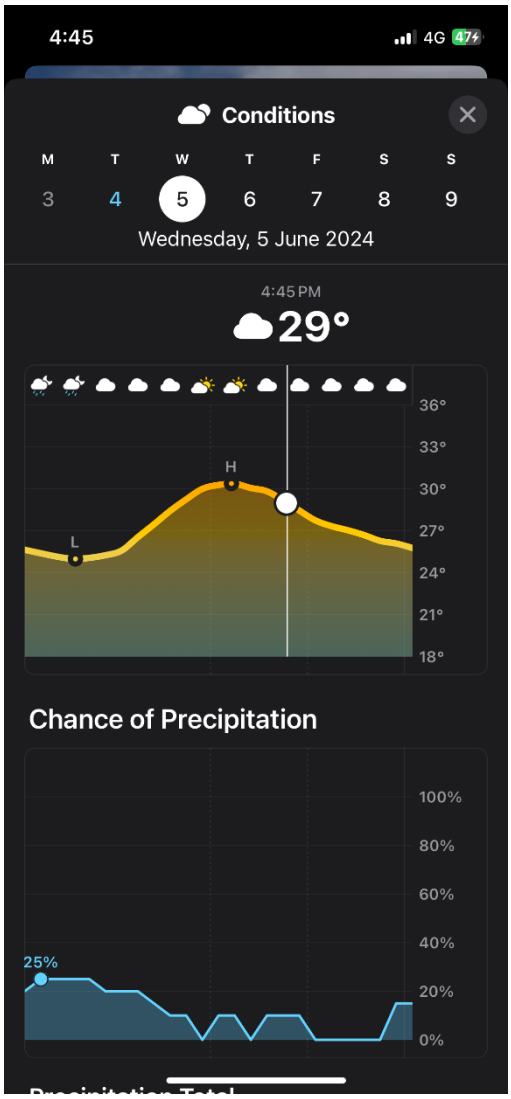

Figure 20: One Day Prediction in Weather app

```
Live Readings from ThingSpeak:
 Temperature (°C)  Humidity (%)  Pressure (hPa)  Soil Moisture (%)  hour  dayofweek  month
             32.0          80.7      1006.60822                0.2  16.0        1.0    6.0

Predicted Weather Code (Next Hour): 0
Weather Description (Next Hour): Cloud development not observed or not observable

Predicted Weather Code (Next Day): 1
Weather Description (Next Day): Clouds generally dissolving or becoming less developed
```

Figure 21: Predicted Output from Google Colab

5.5.3 Analysis of Classification Reports

The classification reports highlight that the model performs well for certain weather codes (e.g., code 1) while struggling with others. This suggests that the model might be biased towards the more frequent weather patterns in the training data and could benefit from a more balanced dataset. Additionally, the relatively low F1-scores for many codes indicate that the model often struggles to balance precision and recall, either misclassifying some instances or missing others altogether.

```
Next Hour Prediction Accuracy: 0.5024184900408405
Next Hour Classification Report:
              precision    recall  f1-score   support

         0.0       0.63      0.55      0.59     11123
         1.0       0.50      0.64      0.57     21288
         2.0       0.45      0.14      0.21     11286
         3.0       0.51      0.07      0.13      4914
        51.0       0.49      0.81      0.61     26116
        53.0       0.27      0.07      0.11      5866
        55.0       0.17      0.00      0.01      1736
        61.0       0.26      0.09      0.14      3194
        63.0       0.39      0.19      0.26      1990
        65.0       0.79      0.08      0.14       145

    accuracy                           0.50     87658
   macro avg       0.45      0.27      0.28     87658
weighted avg       0.48      0.50      0.45     87658


Next Day Prediction Accuracy: 0.4736475849323507
Next Day Classification Report:
              precision    recall  f1-score   support

         0.0       0.59      0.51      0.54     11133
         1.0       0.46      0.61      0.53     21161
         2.0       0.40      0.11      0.17     11225
         3.0       0.49      0.05      0.09      4961
        51.0       0.47      0.80      0.59     26279
        53.0       0.24      0.03      0.05      5896
        55.0       0.27      0.01      0.01      1720
        61.0       0.25      0.03      0.05      3198
        63.0       0.35      0.06      0.11      1945
        65.0       0.71      0.04      0.07       140

    accuracy                           0.47     87658
   macro avg       0.42      0.22      0.22     87658
weighted avg       0.45      0.47      0.41     87658
```

Figure 22: Classification Report for the Predictions

5.6 Overall System Evaluation

While the testing and analysis phase highlighted the challenges of evaluating the IoT-based weather station's accuracy in a rural setting, the project successfully demonstrated its potential to address the problem of unreliable weather forecasts in agricultural areas. The incorporation of diurnal and seasonal features led to a slight improvement in the model's accuracy, showcasing the effectiveness of incorporating domain-specific knowledge.

The weather station consistently collected and presented localized weather data, empowering farmers with valuable information for decision-making regarding irrigation, planting, and crop management. Although discrepancies were observed between the model's predictions and those of established weather apps, these differences highlight the complexities of microclimates and the need for further model refinement.

Despite these challenges, the integration of machine learning algorithms demonstrates the system's potential to provide tailored weather forecasts for specific locations and conditions. With continued development and refinement, this IoT-based weather station can significantly enhance agricultural practices in Sri Lanka by providing farmers with accurate and localized weather information.

Chapter 6: Conclusion and Further Development

6.1 Conclusion

The development of the IoT-based weather station aimed to address the challenges faced by farmers in rural areas of Sri Lanka due to unreliable weather forecasts. By leveraging the power of Internet of Things (IoT) technology and machine learning algorithms, this project sought to provide localized and accurate weather data tailored specifically for agricultural applications.

Through a comprehensive literature review and extensive research, the project identified the critical components and functionalities required for an effective weather monitoring system. The implementation phase involved the integration of various hardware components, including the ESP32 microcontroller, DHT22 temperature and humidity sensor, BMP280 atmospheric pressure sensor, and a soil moisture sensor.

The software implementation played a crucial role in enabling data collection, processing, and visualization. The Arduino programming environment facilitated the seamless integration of sensors and components, while machine learning algorithms were incorporated to analyse historical and real-time weather data, enabling accurate weather forecasting.

To ensure user-friendliness and accessibility, the Adafruit SSD1306 OLED display was implemented, providing farmers with a clear and intuitive interface to monitor current environmental conditions and access predicted weather information. Additionally, the integration with ThingSpeak, a popular IoT platform, allowed for data storage, visualization, and analysis, fostering collaboration and knowledge sharing among stakeholders.

The testing and analysis phase revealed both strengths and areas for improvement in the IoT-based weather station. While the system successfully collected and presented localized weather data, enabling informed decision-making for agricultural practices, the accuracy and reliability of weather forecasts were impacted by factors such as limited training data and the inherent complexity of weather patterns in rural areas.

Feedback from users highlighted the potential benefits of the weather station in supporting sustainable agricultural practices but also emphasized the need for continued improvement in forecast accuracy and reliability.

Overall, the IoT-based weather station project demonstrated its novelty and potential impact in addressing the challenges faced by farmers in rural areas due to unreliable weather forecasts. By providing localized weather information and forecasting capabilities, the system empowers farmers to make informed decisions regarding crop management, irrigation scheduling, and overall agricultural practices, contributing to improved productivity and resource management.

6.2 Further Development

While the IoT-based weather station has laid a solid foundation, there are several areas for further development and improvement:

- **Expanding the Sensor Network:** Deploying additional weather stations across a wider geographical area would enhance the accuracy and granularity of weather data collection. This would enable more localized forecasting and better representation of microclimatic variations in rural regions.
- **Integrating Additional Data Sources:** Incorporating data from external sources, such as satellite imagery, radar data, or government meteorological agencies, could improve the accuracy and reliability of weather forecasts. This would provide a more comprehensive dataset for training the machine learning models.
- **Refining Machine Learning Models:** Continuous refinement and optimization of the machine learning algorithms, including the exploration of advanced techniques like deep learning, could further enhance the accuracy of weather predictions. Additionally, incorporating ensemble methods or combining multiple models could lead to more robust and reliable forecasts.
- **Improving User Interface and Accessibility:** Enhancing the user interface and exploring alternative methods of presenting weather data, such as mobile applications or voice-based interfaces, could increase accessibility and user adoption among farmers, particularly those with limited technological familiarity.
- **Exploring Alternative Power Sources:** Investigating the integration of renewable energy sources, such as solar panels or wind turbines, could ensure the long-term sustainability and independence of the weather station, especially in areas with limited access to conventional power sources.

- **Collaboration and Knowledge Sharing:** Fostering collaboration with research institutions, agricultural organizations, and other stakeholders could facilitate knowledge sharing, data exchange, and the development of best practices for leveraging IoT-based weather monitoring systems in agriculture.
- **Scalability and Commercialization:** Exploring scalability options and potential commercialization avenues could enable the widespread adoption of the IoT-based weather station, empowering more farmers across Sri Lanka and potentially extending to other regions with similar challenges.

By continuously improving and expanding the capabilities of the IoT-based weather station, this project can contribute to the development of sustainable and efficient agricultural practices, supporting food security and economic growth in rural communities.

References

Abidin, S. Z. Y. N. Z. &. A. H., 2020. *Material selection for IoT devices: A review.* [Online] Available at: https://link.springer.com/chapter/10.1007/978-3-030-04164-9_123

Adegboye, O. L. A. I. O. &. A. A. R., 2019. Development and performance evaluation of low-cost automatic weather station for greenhouse applications. *Journal of Applied and Emerging Research,* Issue 4(2), pp. 1-6.

Adekoya, A. A. R. &. A. A., 2019. *Design and development of a low-cost, solar-powered, weather monitoring system for agricultural extension services.* [Online] Available at: https://www.sciencedirect.com/science/article/pii/S2468067222000414

Aghrir, M. A. S. E. &. O. F., 2019. An IoT-based system for agriculture monitoring and control. *In International Conference on Digital Technologies and Applications,* pp. 654-665.

Akter, M. S. R. M. M. &. A.-A.-W. M., 2018. *Development of a low-cost weather station for agricultural applications. Sensors.* [Online] Available at: https://www.sciencedirect.com/science/article/pii/S2468067222000414

Amazon, n.d. *AWS IoT Core – Get Started Guide.* [Online] Available at: https://aws.amazon.com/iot-core/getting-started/

Aregba, A. S. A. A. P. O. O. O. &. A. B. E., 2020. Design and Implementation of an Arduino-Based Weather Monitoring and Forecasting System. *International Journal of Scientific & Engineering Research,* Issue 11(1), pp. 149-156.

Aregba, A. W. O. E. O. &. M. S., 2020. Development of a low-cost IoT-based weather station for crop monitoring. *In International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM),* pp. 335-347.

Board, W. R., 2022. *Annual Report 2022,* s.l.: Water Resources Board, Sri Lanka.

Espressif, 2020. *ESP32 Series Datasheet.* [Online] Available at: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/index.html

Gonzalez-de-Santos, P. e. a., 2019. Fleets of robots for environmentally-safe pest control in agriculture. *Precision Agriculture,* Issue 20(4), pp. 659-678.

Jayasundara, C. T. M. &. P. I., 2020. *IoT-Based Smart Agriculture: A Review. In 2020 Moratuwa Engineering Research Conference.* s.l., IEEE, pp. 1-6.

Jayawardena, S., 2020. Impact of Weather Uncertainty on Rice Cultivation in Sri Lanka. *Agricultural Economics Review,* Issue 8(1), pp. 45-62.

Jones, H. G., 2005. Monitoring plant and soil water status: established and novel methods revisited and their relevance to studies of drought tolerance. *Journal of Experimental Botany,* Issue 57(13), pp. 3543-3460.

Kim, H. K. Y. &. P. S., 2015. A low-power energy-efficient wireless weather station for environmental monitoring. *Sensors,* Issue 15(10), pp. 26054-26074.

Kodali, R. K. &. M. S., 2016. An IoT based weather monitoring system using Arduino. *International Journal of Scientific Research and Management,* Issue 5(5), pp. 1802-1805.

Kumar, V. &. S. Y., 2020. Development of an IoT-based smart weather station for real-time weather monitoring and forecasting. *Procedia Computer Science,* Issue 171, pp. 1555-1562.

Meteorology, D. o., 2022. *Annual Report 2022,* s.l.: Department of Meteorology, Sri Lanka..

News, D., 2021. *Farmers Suffer Losses due to Unseasonal Rains. Daily News.* [Online] Available at: https://www.dailynews.lk/2021/10/15/local/259242/farmers-suffer-losses-due-unseasonal-rains

Park, Y. J. H. J. H. &. P. S. J., 2020. Security considerations for IoT-based smart farm and implementation using blockchain technology. *International Journal of Distributed Sensor Networks.*

Perera, J. &. D. D., 2021. Impact of Weather Variability on Agricultural Productivity in Sri Lanka. *Journal of Agricultural Science,* Issue 16(2), pp. 123-138.

Shahla, N. A. Z. &. H. S., 2021. Low-cost IoT-based smart weather monitoring system. *International Journal of Research and Application (IJRA),* Issue 12(3), pp. 342-351.

Sharma, N. K. S. S. G. A. K. V. &. K. A., 2018. A systematic literature review on machine learning applications for sustainable agriculture supply chain performance. *Computers and Electronics in Agriculture,* Issue 155, pp. 200-220.

Gantt Chart

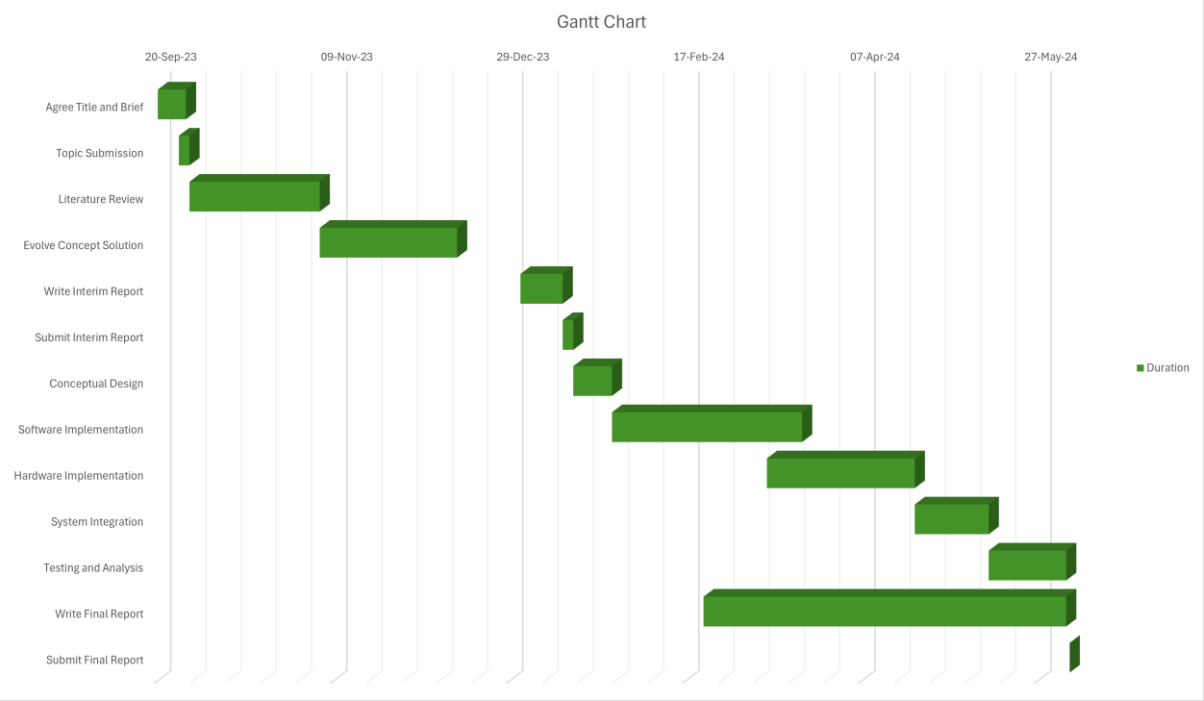

Figure 23: Gantt Chart

Appendix

Arduino Code

```
#include <WiFi.h>
#include <DHT.h>
#include <ThingSpeak.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_SSD1306.h>

// WiFi credentials
const char* ssid = "Dialog 4G 776";
const char* password = "********";

// ThingSpeak channel and API keys
unsigned long channelID = 2468268;
const char* readAPIKey = "JZU34XGC41T0AJCV";
const char* writeAPIKey = "7AP9CBEU9WJ59IO8";

// Sensor pin definitions
#define DHTPIN 19     // DHT22 temperature/humidity sensor
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define SOIL_MOISTURE_PIN 34 // Soil moisture sensor

// Display setup
Adafruit_BMP280 bmp; // BMP280 pressure sensor
#define SCREEN_WIDTH 128 // OLED display
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

WiFiClient client;

// Function to get WMO code description (simplified)
String getWmoDescription(int wmoCode) {
   switch (wmoCode) {
      case 0:  return "Cloud dev. not observed";
      case 1:  return "Clouds dissolving";
      case 2:  return "State of sky unchanged";
      case 3:  return "Clouds forming";
      case 51: return "Cont. drizzle (slight)";
      case 53: return "Cont. drizzle (moderate)";
      case 55: return "Cont. drizzle (heavy)";
      case 61: return "Cont. rain (slight)";
      case 63: return "Cont. rain (moderate)";
      case 65: return "Cont. rain (heavy)";
      default: return "Unknown code";
   }
```

```
}

void setup() {
  Serial.begin(115200);
  dht.begin();
  bmp.begin(0x76);    // Initialize pressure sensor
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // Initialize display
  display.clearDisplay();

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  ThingSpeak.begin(client);
}

void loop() {
  // Read sensor values
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  int soilMoistureRaw = analogRead(SOIL_MOISTURE_PIN);
  int soilMoisture = map(soilMoistureRaw, 0, 4095, 100, 0);
  float pressure = bmp.readPressure() / 100.0F;

  // Check for sensor errors
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Send data to ThingSpeak
  ThingSpeak.setField(1, temperature);
  ThingSpeak.setField(2, humidity);
  ThingSpeak.setField(4, soilMoisture);
  ThingSpeak.setField(3, pressure);

  int statusCode = ThingSpeak.writeFields(channelID, writeAPIKey);
  if (statusCode == 200) {
    Serial.println("Channel update successful");
  } else {
    Serial.println("Problem updating channel. HTTP error code " + String(statusCode));
  }

  // Read predictions from ThingSpeak
  long nextHourCode = ThingSpeak.readLongField(channelID, 5, readAPIKey);
  long nextDayCode = ThingSpeak.readLongField(channelID, 6, readAPIKey);
```

```
    // Display on OLED
    if (nextHourCode != -1 && nextDayCode != -1) {
        String nextHourDescription = getWmoDescription(nextHourCode);
        String nextDayDescription = getWmoDescription(nextDayCode);

        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);

        // Sensor readings
        display.setCursor(0, 0);
        display.println("Temperature: " + String(temperature) + " C");
        // ... (rest of the sensor readings)

        // Predictions
        display.setCursor(0, 32);
        display.println("Next Hour: " + nextHourDescription);
        display.println("Next Day:  " + nextDayDescription);
    } else {
        // Handle read errors
        display.clearDisplay();
        display.println("Error reading forecast");
    }
    display.display();
    delay(60000); // Wait 60 seconds
}
```

Google Colab Code

```
!pip install thingspeak
!pip install tzdata

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.multioutput import MultiOutputClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.impute import SimpleImputer
import thingspeak
import requests
from google.colab import drive
from datetime import timedelta, datetime

drive.mount('/content/drive')

# Specify the path to your CSV file within Google Drive
data_path = '/content/drive/MyDrive/Weather Dataset/Srilanka_Weather_Dataset.csv'

# Load the CSV data into a pandas DataFrame
df = pd.read_csv(data_path)

# Check for missing values
print(df.isnull().sum())

# Create a dictionary mapping WMO codes to descriptions
wmo_code_descriptions = {
    0: "Cloud development not observed or not observable",
    1: "Clouds generally dissolving or becoming less developed",
    2: "State of sky on the whole unchanged",
    3: "Clouds generally forming or developing",
    4: "Visibility reduced by smoke",
    5: "Haze",
    6: "Widespread dust in suspension in the air, not raised by wind at or near the station at the
time of observation",
    7: "Dust or sand raised by wind at or near the station at the time of observation",
    8: "Well developed dust whirl(s) or sand whirl(s)",
    9: "Duststorm or sandstorm",
    10: "Mist",
    11: "Patches of shallow fog or ice fog at the station",
    12: "More or less continuous shallow fog or ice fog at the station",
    13: "Lightning visible, no thunder heard",
    14: "Precipitation within sight, not reaching the ground or the surface of the sea",
    15: "Precipitation within sight, reaching the ground or the surface of the sea",
```

16: "Precipitation within sight, reaching the ground or the surface of the sea, near to station",
17: "Thunderstorm, but no precipitation at the time of observation",
18: "Squalls at or within sight of the station during the preceding hour or at the time of observation",
19: "Funnel cloud(s)",
20: "Drizzle (not freezing) or snow grains not falling as shower(s)",
21: "Rain (not freezing) not falling as shower(s)",
22: "Snow not falling as shower(s)",
23: "Rain and snow or ice pellets, type (a) not falling as shower(s)",
24: "Freezing drizzle or freezing rain not falling as shower(s)",
25: "Shower(s) of rain",
26: "Shower(s) of snow, or of rain and snow",
27: "Shower(s) of hail, or of rain and hail",
28: "Fog or ice fog",
29: "Thunderstorm (with or without precipitation)",
30: "Slight or moderate duststorm or sandstorm - has decreased during the preceding hour",
31: "Slight or moderate duststorm or sandstorm - no appreciable change during the preceding hour",
32: "Slight or moderate duststorm or sandstorm - has begun or has increased during the preceding hour",
33: "Severe duststorm or sandstorm - has decreased during the preceding hour",
34: "Severe duststorm or sandstorm - no appreciable change during the preceding hour",
35: "Severe duststorm or sandstorm - has begun or has increased during the preceding hour",
36: "Slight or moderate blowing snow generally low (below eye level)",
37: "Heavy drifting snow generally low (below eye level)",
38: "Slight or moderate blowing snow generally high (above eye level)",
39: "Heavy drifting snow generally high (above eye level)",
40: "Fog or ice fog at a distance at the time of observation, but not at the station during the preceding hour, the fog or ice fog extending to a level above that of the observer",
41: "Fog or ice fog in patches",
42: "Fog or ice fog, sky visible has become thinner during the preceding hour",
43: "Fog or ice fog, sky invisible has become thinner during the preceding hour",
44: "Fog or ice fog, sky visible no appreciable change during the preceding hour",
45: "Fog or ice fog, sky invisible no appreciable change during the preceding hour",
46: "Fog or ice fog, sky visible has begun or has become thicker during the preceding hour",
47: "Fog or ice fog, sky invisible has begun or has become thicker during the preceding hour",
48: "Fog, depositing rime, sky visible",
49: "Fog, depositing rime, sky invisible",
50: "Drizzle, not freezing, intermittent slight at time of observation",
51: "Drizzle, not freezing, continuous slight at time of observation",
52: "Drizzle, not freezing, intermittent moderate at time of observation",
53: "Drizzle, not freezing, continuous moderate at time of observation",
54: "Drizzle, not freezing, intermittent heavy (dence) at time of observation",
55: "Drizzle, not freezing, continuous heavy (dence) at time of observation",
56: "Drizzle, freezing, slight",
57: "Drizzle, freezing, moderate or heavy (dence)",
58: "Drizzle and rain, slight",
59: "Drizzle and rain, moderate or heavy",
60: "Rain, not freezing, intermittent slight at time of observation",

61: "Rain, not freezing, continous slight at time of observation",
62: "Rain, not freezing, intermittent moderate at time of observation",
63: "Rain, not freezing, continuous moderate at time of observation",
64: "Rain, not freezing, intermittent heavy at time of observation",
65: "Rain, not freezing, continuous heavy at time of observation",
66: "Rain, freezing, slight",
67: "Rain, freezing, moderate or heavy",
68: "Rain, or drizzle and snow, slight",
69: "Rain, or drizzle and snow, moderate or heavy",
70: "Intermittent fall of snow flakes slight at time of observation",
71: "Continuous fall of snow flakes slight at time of observation",
72: "Intermittent fall of snow flakes moderate at time of observation",
73: "Continuous fall of snow flakes moderate at time of observation",
74: "Intermittent fall of snow flakes heavy at time of observation",
75: "Continuous fall of snow flakes heavy at time of observation",
76: "Ice prisms (with or without fog)",
77: "Snow grains (with or without fog)",
78: "Isolated starlike snow crystals (with or without fog)",
79: "Ice pellets, type (a)",
80: "Rain shower(s), slight",
81: "Rain shower(s), moderate or heavy",
82: "Rain shower(s), violent",
83: "Shower(s) of rain and snow mixed, slight",
84: "Shower(s) of rain and snow mixed, moderate or heavy",
85: "Snow shower(s), slight",
86: "Snow shower(s), moderate or heavy",
87: "Shower(s) of snow pellets or ice pellets, type (b), with or without rain or rain and snow mixed - slight",
88: "Shower(s) of snow pellets or ice pellets, type (b), with or without rain or rain and snow mixed - moderate or heavy",
89: "Shower(s) of hail*, with or without rain or rain and snow mixed, not associated with thunder - slight",
90: "Shower(s) of hail*, with or without rain or rain and snow mixed, not associated with thunder - moderate or heavy",
91: "Slight rain at time of observation - thunderstorm during the preceding hour but not at time of observation",
92: "Moderate or heavy rain at time of observation - thunderstorm during the preceding hour but not at time of observation",
93: "Slight snow, or rain and snow mixed or hail** at time of observation - thunderstorm during the preceding hour but not at time of observation",
94: "Moderate or heavy snow, or rain and snow mixed or hail** at time of observation - thunderstorm during the preceding hour but not at time of observation",
95: "Thunderstorm, slight or moderate, without hail**, but with rain and/or snow at time of observation - thunderstorm at time of observation",
96: "Thunderstorm, slight or moderate, with hail** at time of observation - thunderstorm at time of observation",
97: "Thunderstorm, heavy, without hail**, but with rain and/or snow at time of observation - thunderstorm at time of observation",
98: "Thunderstorm combined with duststorm or sandstorm at time of observation - thunderstorm at time of observation",

```
  99: "Thunderstorm, heavy, with hail** at time of observation - thunderstorm at time of observation"
}

# Convert 'time' column to datetime format (using ISO 8601 format)
df['time'] = pd.to_datetime(df['time'], format='%Y-%m-%dT%H:%M')
df['time'] = df['time'].dt.tz_localize('GMT').dt.tz_convert('Asia/Colombo')

# Extract time-related features
df['hour'] = df['time'].dt.hour
df['dayofweek'] = df['time'].dt.dayofweek
df['month'] = df['time'].dt.month

# Diurnal Cycle Features
df['time_of_day'] = pd.cut(df['hour'], bins=[0, 6, 12, 18, 24], labels=['Night', 'Morning', 'Afternoon', 'Evening'])
# One-Hot Encode the time of day feature
df = pd.get_dummies(df, columns=['time_of_day'],prefix='',prefix_sep='')

# Sri Lankan Seasonal Features
def get_season(month):
  if month in [12, 1, 2]:
    return 'Maha'  # Northeast Monsoon
  elif month in [3, 4]:
    return 'Inter-Monsoon 1'  # First Inter-Monsoon
  elif month in [5, 6, 7, 8, 9]:
    return 'Yala'   # Southwest Monsoon
  else:
    return 'Inter-Monsoon 2'  # Second Inter-Monsoon

df['season'] = df['month'].apply(get_season)
# One-Hot Encode the season feature
df = pd.get_dummies(df, columns=['season'],prefix='',prefix_sep='')

# Lag features for next hour prediction
df['temperature_2m_prev_hour'] = df['temperature_2m (°C)'].shift(1)
df['relative_humidity_2m_prev_hour'] = df['relative_humidity_2m (%)'].shift(1)
df['surface_pressure_prev_hour'] = df['surface_pressure (hPa)'].shift(1)
df['soil_moisture_prev_hour'] = df['soil_moisture_0_to_7cm (m³/m³)'].shift(1)

# Lag features for next 24-hour prediction
df['temperature_2m_prev_day'] = df['temperature_2m (°C)'].shift(24)
df['relative_humidity_2m_prev_day'] = df['relative_humidity_2m (%)'].shift(24)
df['surface_pressure_prev_day'] = df['surface_pressure (hPa)'].shift(24)
df['soil_moisture_prev_day'] = df['soil_moisture_0_to_7cm (m³/m³)'].shift(24)

# Target variables
df['weather_code_next_hour'] = df['weather_code (wmo code)'].shift(-1)
df['weather_code_next_day'] = df['weather_code (wmo code)'].shift(-24)
```

```
# Drop rows with NaN values (created by the shifts)
df.dropna(inplace=True)

# Updated features list
features = ['temperature_2m (°C)', 'relative_humidity_2m (%)', 'surface_pressure (hPa)',
        'soil_moisture_0_to_7cm (m³/m³)', 'hour', 'dayofweek', 'month',
        'temperature_2m_prev_hour', 'relative_humidity_2m_prev_hour',
'surface_pressure_prev_hour', 'soil_moisture_prev_hour',
        'temperature_2m_prev_day', 'relative_humidity_2m_prev_day',
'surface_pressure_prev_day',
'soil_moisture_prev_day','Afternoon','Evening','Maha','Morning','Night','Yala']

X = df[features]
y = df[['weather_code_next_hour', 'weather_code_next_day']]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Impute missing values with the mean of each column
imputer = SimpleImputer(strategy='mean')

X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Create a multi-output model
multi_output_model = MultiOutputClassifier(RandomForestClassifier(random_state=42))

# Train the ensemble model (this is usually very fast)
multi_output_model.fit(X_train, y_train)

#Make predictions on the test set
y_pred = multi_output_model.predict(X_test)

#Evaluate the model performance
accuracy_next_hour = accuracy_score(y_test['weather_code_next_hour'], y_pred[:,0])
report_next_hour = classification_report(y_test['weather_code_next_hour'], y_pred[:,0])

accuracy_next_day = accuracy_score(y_test['weather_code_next_day'], y_pred[:,1])
report_next_day = classification_report(y_test['weather_code_next_day'], y_pred[:,1])

print("\nNext Hour Prediction Accuracy:", accuracy_next_hour)
print("Next Hour Classification Report:\n", report_next_hour)
print("\nNext Day Prediction Accuracy:", accuracy_next_day)
print("Next Day Classification Report:\n", report_next_day)

# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler to the training data
scaler.fit(X_train)
```

```
# Thingspeak Channel ID, Read and Write API Keys
channel_id = '2468268'
read_api_key = 'JZU34XGC41T0AJCV'
write_api_key='7AP9CBEU9WJ59IO8'

# Fetch latest data from ThingSpeak
url=f"https://api.thingspeak.com/channels/{channel_id}/feeds.json?api_key={read_api_key}
&results=1"
response = requests.get(url)

# Check if the request was successful
if response.status_code == 200:
    feed = response.json()['feeds'][0]  # Get the latest feed entry

   # Prepare live data for prediction
    new_data = pd.DataFrame({
        'temperature_2m (°C)': [feed['field1']],
        'relative_humidity_2m (%)': [feed['field2']],
        'surface_pressure (hPa)': [feed['field3']],
        'soil_moisture_0_to_7cm (m³/m³)': [float(feed['field4'])/100],
        'hour': [pd.to_datetime(feed['created_at']).tz_convert('Asia/Colombo').hour],
        'dayofweek': [pd.to_datetime(feed['created_at']).tz_convert('Asia/Colombo').dayofweek],
        'month': [pd.to_datetime(feed['created_at']).tz_convert('Asia/Colombo').month]
    })
    # Ensure data types are correct
    new_data = new_data.astype(float)

# Standardize new data
    scaler_new_data = StandardScaler()
    new_data_scaled = scaler_new_data.fit_transform(new_data)
    new_data_scaled = pd.DataFrame(new_data_scaled, columns = new_data.columns)

    # Add the columns that are not in the new_data
    for col in features:
        if col not in new_data_scaled.columns:
            new_data_scaled[col] = 0

    # Make predictions
    predictions = multi_output_model.predict(new_data_scaled[features]) #use the correct
features here

    next_hour_prediction = int(predictions[0][0])
    next_day_prediction = int(predictions[0][1])

    # Get descriptions
    next_hour_desc = wmo_code_descriptions.get(next_hour_prediction, "Unknown")
    next_day_desc = wmo_code_descriptions.get(next_day_prediction, "Unknown")

    # Rename columns for display
```

```
    new_data_display = new_data.rename(columns={
        'temperature_2m (°C)': 'Temperature (°C)',
        'relative_humidity_2m (%)': 'Humidity (%)',
        'surface_pressure (hPa)': 'Pressure (hPa)',
        'soil_moisture_0_to_7cm (m³/m³)': 'Soil Moisture (%)'
    })

    # Print live readings and prediction
    print("Live Readings from ThingSpeak:")
    print(new_data_display.to_string(index=False))  # Display renamed readings
    print("\nPredicted Weather Code (Next Hour):", next_hour_prediction)
    print("Weather Description (Next Hour):", next_hour_desc)
    print("\nPredicted Weather Code (Next Day):", next_day_prediction)
    print("Weather Description (Next Day):", next_day_desc)
    # Write the predicted weather code to ThingSpeak
    channel = thingspeak.Channel(id=channel_id, api_key=write_api_key)

    try:
        response = channel.update({
            'field5': next_hour_prediction,
            'field6': next_day_prediction
        })
        if response == 200:
            print("Predicted weather codes written to ThingSpeak successfully! Entry ID:",
response.json()['entry_id'])
        else:
            print("Error writing to ThingSpeak:", response.json())
    except Exception as e:
        print("An error occurred while writing to ThingSpeak:", e)

else:
    print("Error fetching data from ThingSpeak:", response.status_code)
```