

“SmartPlant: An IoT-Based System for Indoor Plant Monitoring and Assistance”

submitted by

SUDIP GHOSH

Class Roll No: 002210503019

Examination Roll No: MCA244018

Registration No: 163611 of 2022-2023

Master of Computer Application

Under the supervision of

Prof. (Dr.) CHANDREYEE CHOWDHURY

in the

Department of Computer Science & Engineering

Faculty of Engineering and Technology

Jadavpur University, Kolkata-700032

2024

Faculty of Engineering and Technology

Jadavpur University

CERTIFICATE OF RECOMMENDATION

To Whom It May Concern

I hereby recommend that the project report entitled “SmartPlant: An IoT-Based System for Indoor Plant Monitoring and Assistance” has been carried out by Sudip Ghosh (Reg. No.: 163611 of 2022-23, Roll No.: 002210503019, Exam Roll: MCA244018), under my guidance and supervision and be accepted in partial fulfilment of the requirement for the degree of Master of COMPUTER APPLICATION in DEPARTMENT of COMPUTER SCIENCE and ENGINEERING, JADAVPUR UNIVERSITY during the academic year 2023-2024.

Prof. (Dr.) Chandreyee Chowdhury
Project Supervisor
Dept. of Computer Science and Engineering
Jadavpur University

Prof. (Dr.) Nandini Mukhopadhyay
Head of the Department
Dept. of Computer Science & Engineering
Jadavpur University

Prof. Dipak Laha
Dean, Faculty Council
of Engineering & Technology
Jadavpur University, Kolkata-032

Faculty of Engineering and Technology
Jadavpur University

CERTIFICATE OF APPROVAL

This is to certify that the project report entitled “**SmartPlant: An IoT-Based System for Indoor Plant Monitoring and Assistance**” is a Bonafide record of work carried out by SUDIP GHOSH, in fulfillment of the requirements for the award of the degree of Master of Computer Application in the Department of Computer Science and Engineering, Jadavpur University. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the project report only for the purpose for which it has been submitted.

Signature of Examiner 1

Date:

Signature of Examiner 2

Date:

Faculty of Engineering and Technology
Jadavpur University

**DECLARATION OF COMPLIANCE WITH ACADEMIC
ETHICS**

I hereby declare that this project report entitled "SmartPlant: An IoT-Based System for Indoor Plant Monitoring and Assistance" is an outcome of my genuine and independent work, as part of my Master of Computer Application program.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, all materials and results that are not original to this work have been fully cited and referenced.

Sudip Ghosh

Registration No.: 163611 of 2022-23

Examination Roll No.: MCA244018

Thesis Title: - SmartPlant: An IoT-Based System for Indoor Plant Monitoring and Assistance

Signature with Date

ACKNOWLEDGEMENT

With my most sincere gratitude, I would like to thank Dr. Chandreyee Chowdhury, Department of Computer Science & Engineering, my supervisor, for her overwhelming support throughout the duration of the project. Her motivation always gave me the necessary inputs and momentum to continue with my work, without which the project would not have taken its current shape. Her valuable suggestions and numerous discussions have always inspired new ways of thinking. I feel deeply honored to have had the opportunity to work under her guidance.

I would also like to express my sincere thanks to all my teachers for providing a sound knowledge base and their cooperation.

Furthermore, I extend my heartfelt thanks to all the faculty members of the Department of Computer Science & Engineering at Jadavpur University for their continuous support.

Last, but not least, I would like to thank my batchmates for staying by my side when I needed them.

Sudip Ghosh

Class Roll No.: 002210503019

Examination Roll No.: MCA244018

Department of Computer Science & Engineering

Jadavpur University

Table of Contents

1. Introduction.....	1
1.1 Overview	1
1.2 Motivation	2
1.3 Objectives	2
1.4 Significance	2
1.5 Contribution.....	3
2. Background and Related works	4
2.1 Background.....	4
2.2 IC & Sensors	4
2.3 Previous work.....	5
3. Experiment Design.....	7
3.1 Overview	7
3.2 Materials	8
3.3 Methodology	9
4. Proposed Work.....	11
4.1 Overview	11
4.2 System Architecture	11
4.3 IoT Device (Hardware Architecture)	11
4.4 Cloud Infrastructure	16
4.5 Mobile Application.....	19
5. Experiment Results	23
5.1 Experimental Setup	23
5.2 Power Consumption Analysis	26
6. Conclusions and Future work.....	31
6.1 Conclusion.....	31
6.2 Future Work.....	31
7. References	32

1. Introduction

1.1 Overview

Indoor gardening, as a practice, has transcended its traditional role as a hobby to become a crucial component of contemporary urban living. With the exponential growth of urban populations and the increasing scarcity of outdoor green spaces, indoor gardening has emerged as a viable solution to reconnect individuals with nature within the confines of urban environments. This resurgence of interest in indoor gardening can be attributed to a multitude of factors, including the desire for sustainable living, the pursuit of healthier lifestyles, and the aesthetic appeal of bringing greenery indoors.

Urbanization, characterized by the rapid expansion of cities and the densification of populations, has fundamentally altered the way people interact with their surroundings. As urban areas continue to expand, access to traditional outdoor gardening spaces becomes increasingly limited. High-rise apartments, condominiums, and compact living spaces have become the norm in many urban centers, leaving little room for traditional outdoor gardens. In response to this urban shift, indoor gardening has gained popularity as a practical and convenient alternative for urban dwellers seeking to cultivate plants within their living spaces.

Furthermore, the growing awareness of environmental issues and the importance of sustainable practices have prompted individuals to seek ways to reduce their carbon footprint and live more eco-friendly lifestyles. Indoor gardening offers a sustainable means of producing fresh herbs, vegetables, and ornamental plants without the need for extensive land use or transportation. By growing plants indoors, individuals can minimize their reliance on commercially grown produce, thereby reducing greenhouse gas emissions associated with transportation and agriculture.

Moreover, indoor gardening provides numerous health benefits that contribute to overall well-being. Studies have shown that indoor plants can improve air quality by removing toxins and pollutants, thereby creating a healthier indoor environment. Additionally, caring for indoor plants can have therapeutic effects, reducing stress levels and promoting relaxation. The act of nurturing plants fosters a sense of connection with nature, even in the heart of urban landscapes, enhancing mental and emotional well-being.

According to a report by Grand View Research [1], the global indoor farming market size was valued at USD 106.6 billion in 2020 and is projected to reach USD 140.4 billion by 2027, exhibiting a CAGR of 4.2% during the forecast period. The increasing urbanization trend is driving demand for indoor gardening solutions, with an estimated 68% of the world's population projected to live in urban areas by 2050, according to the United Nations Department of Economic and Social Affairs [2].

However, despite these advancements, there remains a need for comprehensive and user-friendly solutions that cater specifically to the unique requirements of indoor gardening. Existing smart plant monitoring systems often lack integration, scalability, and user-centric design, limiting their effectiveness in indoor environments. Addressing these limitations requires the development of a tailored smart plant monitoring system that combines

hardware, software, and cloud-based infrastructure to provide indoor gardeners with a seamless and intuitive means of monitoring and managing their plants.

1.2 Motivation

While existing studies have made significant contributions to smart agriculture and indoor plant monitoring, there are areas that remain underexplored or insufficiently addressed. For instance, existing systems often lack comprehensive integration, scalability, and user-centric design, limiting their effectiveness in diverse indoor gardening environments. Additionally, energy efficiency and sustainability, crucial for long-term deployment of IoT devices, are not always adequately prioritized. Our project aims to build on these foundations by addressing the specific needs of indoor gardening with a focus on user-friendliness, scalability, and energy efficiency. The primary objectives include developing a robust hardware platform for monitoring environmental variables pertinent to indoor plants, implementing energy-efficient data transmission protocols, and designing an intuitive mobile application interface for real-time access to plant health data and actionable insights. By prioritizing these objectives, our smart plant monitoring system aims to fill the gaps identified in previous research, offering a more holistic and user-centric solution for indoor gardening enthusiasts.

1.3 Objectives

The primary objective of this project is to design, develop, and evaluate a smart plant monitoring system tailored specifically for indoor gardening plants. The system will utilize NodeMCU for computational tasks and internet connectivity, enabling remote data transmission to a centralized database. Key objectives include:

1. Develop a robust plant monitoring device tailored for indoor plants, prioritizing sensitivity to ambient conditions to ensure optimal plant health and growth.
2. Implement clustering capabilities to enable seamless connection and monitoring of multiple plants simultaneously, enhancing efficiency and scalability of the system.
3. Design a user-friendly and intuitive interface that provides comprehensive plant data to users, empowering them to make informed decisions and take necessary actions for plant care and maintenance.
4. Engineer a sustainable and cost-effective solution with low power consumption to prolong battery life, ensuring long-term usability and environmental responsibility while maintaining affordability for users.

1.4 Significance

The proposed smart plant monitoring system tailored for indoor gardening plants holds immense significance in addressing the evolving needs of urban dwellers and advancing sustainable living practices. By providing a seamless integration of IoT technology, cloud computing, and user-centric design, the system offers indoor gardening enthusiasts an innovative solution to monitor and manage their plants with ease. This system not only enhances the accessibility and convenience of indoor gardening but also contributes to environmental sustainability by promoting the cultivation of fresh produce within urban spaces. Furthermore, by fostering a deeper connection between individuals and nature, the system has the potential to improve overall well-being and quality of life in urban environments. Through its scalability, efficiency, and user-friendly interface, the smart plant

monitoring system represents a significant step towards redefining indoor gardening practices and empowering individuals to lead healthier, greener lifestyles in an increasingly urbanized world.

1.5 Contribution

This thesis contributes to the field of indoor gardening by presenting a novel smart plant monitoring system designed specifically for indoor gardening plants. By integrating IoT technology, cloud, and user-centric design principles, the system offers a comprehensive solution to address the unique challenges associated with plant care in indoor environments. The development of a robust hardware platform capable of monitoring multiple environmental variables, coupled with energy-efficient data transmission protocols, represents a significant technological advancement in the field. Moreover, the design of an intuitive mobile application interface empowers users with real-time access to plant health data and actionable insights for optimal plant care and maintenance.

Additionally, this research contributes to advancing sustainable living practices by promoting indoor gardening as a viable means of producing fresh produce in urban settings. By reducing reliance on commercially grown produce and minimizing carbon emissions associated with transportation and agriculture, the smart plant monitoring system encourages environmental stewardship and supports efforts towards mitigating climate change. Furthermore, by fostering a deeper connection between individuals and nature, the system promotes mental and emotional well-being, contributing to overall quality of life in urban environments. Through its innovative approach and practical applications, this research makes a significant contribution to the advancement of indoor gardening practices and sustainable urban living.

2. Background and Related works

2.1 Background

2.1.1 Internet of Things

The Internet of Things (IoT) revolutionizes connectivity by enabling everyday objects to collect, exchange, and analyze data. In the context of smart plant monitoring, IoT devices equipped with sensors facilitate real-time monitoring of environmental variables crucial for plant health, such as soil moisture, light intensity, temperature, and humidity. This interconnectedness empowers users to remotely monitor and manage their indoor gardens, optimizing plant care and fostering sustainable practices. According to recent statistics, the global IoT in agriculture market is projected to reach USD 28.56 billion by 2027, growing at a CAGR of 10.1% from 2020 to 2027, highlighting the increasing adoption of IoT technologies in agriculture [3].

2.1.2 Firebase

Firebase, a cloud-based platform developed by Google, offers a suite of tools and services for app development, including real-time database capabilities. In the context of smart plant monitoring systems, Firebase provides a reliable and scalable infrastructure for storing and accessing plant data securely. By leveraging Firebase's real-time database features, users can access up-to-date plant health information from any device with internet connectivity, facilitating seamless monitoring and management of indoor gardens. Firebase's robust security features and scalability make it a popular choice for IoT applications, with over 1.5 million apps actively using Firebase services [4].

2.1.3 Flutter

Flutter is an open-source UI software development kit (SDK) developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It offers a rich set of pre-designed widgets and tools that enable developers to create beautiful and responsive user interfaces with ease. Flutter's hot reload feature allows for real-time code changes and instant updates, streamlining the development process and enhancing productivity. With its fast performance, expressive UI, and cross-platform compatibility, Flutter has gained widespread adoption among developers for creating high-quality mobile applications, including those for smart plant monitoring systems. As of 2023, Flutter has been used by more than 500,000 developers worldwide, reflecting its growing popularity and versatility [5].

2.2 IC & Sensors

2.2.1 NodeMCU

The NodeMCU is an open-source development board based on the ESP8266 Wi-Fi module. It serves as the central hub for the system, facilitating wireless communication and data processing for various sensors. Its integrated Wi-Fi connectivity and extensive GPIO pins make it ideal for IoT projects [6].

2.2.2 AHT10

The AHT10 is a high-precision digital sensor that measures temperature and humidity. It utilizes MEMS technology for accurate and reliable readings, crucial for monitoring the environmental conditions of indoor plants [7].

2.2.3 BH1750

The BH1750 is a digital ambient light sensor capable of measuring light intensity. Its high sensitivity and wide dynamic range make it suitable for ensuring optimal light conditions for plant growth [8].

2.2.4 SSD1306 OLED Display

This OLED display module offers high contrast and clear visibility, making it ideal for displaying real-time data from the sensors. It enhances user experience by providing immediate visual feedback on environmental conditions [9].

2.2.5 Soil Moisture Sensor

This sensor measures the water content in the soil, providing essential data to prevent over- or under-watering of plants. It employs either resistive or capacitive sensing principles to detect soil moisture levels [10].

2.3 Previous work

Numerous studies have explored the integration of technology, particularly IoT and data analytics, in the field of agriculture, focusing on enhancing productivity, efficiency, and sustainability. Palandea *et al.* introduced a fully automated hydroponic system for indoor plant growth, leveraging microcontrollers, sensors, and IoT technology to regulate environmental parameters essential for plant growth. Their study demonstrated successful plant growth within the system, highlighting the potential of automation in optimizing plant cultivation independent of external climate conditions [11].

Elijah *et al.* provided an overview of IoT and data analytics in agriculture, emphasizing the benefits and challenges of adopting smart farming practices. Their analysis highlighted the role of IoT ecosystems, incorporating wireless sensor networks and cloud computing, in revolutionizing agricultural processes and decision-making [12]. Similarly, Ayaz *et al.* discussed the transformation of traditional farming practices into smart farming 4.0 through the integration of IoT-based technologies. They identified challenges and opportunities in leveraging wireless sensors and IoT devices for various agricultural applications, emphasizing the need for further research to address these challenges and capitalize on emerging opportunities [13].

Furthermore, Ezhilazhahi and Bhuvaneswari focused on IoT-enabled soil moisture monitoring using wireless sensor networks, proposing the Exponential Weighted Moving Average algorithm to enhance network lifetime. Their study aimed to improve plant health monitoring through advanced technology, addressing challenges in agriculture and providing practical insights for real-time implementation [14].

In summary, while previous work has significantly good smart agriculture and indoor plant monitoring, our project addresses the key gaps by focusing on comprehensive integration, user-centric design, and energy efficiency. By building on these foundations, we aim to provide a more effective and sustainable solution for indoor gardening enthusiasts.

3. Experiment Design

3.1 Overview

The primary goal of this project is to develop an advanced and comprehensive smart plant monitoring system specifically designed for indoor gardening. This system will leverage IoT technology and cloud computing to provide real-time monitoring and management of indoor plants, ensuring their optimal health and growth. The detailed objectives are as follows:

3.1.1 Develop a Robust Plant Monitoring Device:

Purpose: To create a reliable and efficient device tailored for indoor plants, which are particularly sensitive to ambient conditions such as light, temperature, humidity, and soil moisture.

Approach: Utilize high-precision sensors (AHT10 for temperature and humidity, BH1750 for light intensity, and soil moisture sensors) integrated with a NodeMCU microcontroller to continuously monitor these critical environmental parameters. This ensures that the system can accurately track and respond to the varying needs of indoor plants, promoting their health and growth.

3.1.2 Implement Clustering Capabilities:

Purpose: To enable seamless connection and monitoring of multiple plants simultaneously, enhancing the efficiency and scalability of the system.

Approach: Each IoT device can support multiple soil moisture sensors, allowing it to monitor multiple plants within a cluster. This is efficient because other environmental sensors such as light, humidity, and temperature are shared among the plants in the cluster. Therefore, a single IoT device can efficiently manage and monitor multiple plants by utilizing shared data from these sensors, enhancing both the scalability and efficiency of the system.

3.1.3 Design a User-Friendly Interface:

Purpose: To provide users with an intuitive and comprehensive interface that displays all necessary plant data, enabling them to make informed decisions and take appropriate actions for plant care and maintenance.

Approach: Create a mobile application using Flutter that interfaces with Firebase to retrieve real-time data on plant conditions. The app will feature a user-friendly design with clear visualizations (graphs and charts), alerts, and actionable recommendations, ensuring that users can easily monitor their plants and respond to any issues promptly.

3.1.4 Engineer a Sustainable and Cost-Effective Solution:

Purpose: To design a system that is both environmentally responsible and affordable for users, with low power consumption to prolong battery life and ensure long-term usability.

Approach: We will utilize the deep sleep feature in the NodeMCU to minimize energy consumption. The device will enter deep sleep mode, waking up at predetermined intervals to collect and transmit data before returning to sleep. This significantly reduces power usage, extending battery life and enhancing sustainability. Additionally, we are making the system cost-effective by allowing one NodeMCU device to monitor multiple plants using multiple

soil moisture sensors. This means users can manage several plants without the need to purchase multiple devices, reducing overall costs.

By achieving these objectives, the smart plant monitoring system aims to provide a comprehensive, scalable, user-centric, and sustainable solution for indoor gardening, addressing the specific needs of indoor plants and their caretakers.

3.2 Materials

The materials used in the development of the smart plant monitoring system can be broadly categorized into hardware components and software tools. Each element plays a critical role in ensuring the system's functionality, reliability, and user-friendliness.

Hardware Components:

3.2.1 NodeMCU

Description: The NodeMCU is an open-source IoT platform. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems and hardware which is based on the ESP-12 module.

Function: This microcontroller serves as the central hub for the system, collecting data from various sensors, processing the data, and transmitting it to the cloud via Wi-Fi.

3.2.2 AHT10

Description: The AHT10 is a high-precision, fully calibrated digital sensor for measuring temperature and humidity.

Function: This sensor provides accurate readings of ambient temperature and humidity levels, which are crucial for maintaining optimal conditions for indoor plants.

3.2.3 BH1750

Description: The BH1750 is a digital light intensity sensor. It measures the intensity of light in lux units.

Function: This sensor monitors the light levels in the plant's environment, ensuring they receive adequate light for photosynthesis and growth.

3.2.4 Soil Moisture Sensor

Description: A sensor designed to measure the moisture content in the soil.

Function: This sensor helps monitor the water needs of the plants, preventing over or under-watering by providing real-time soil moisture data.

3.2.5 128x64 OLED SSD1306 Display

Description: A compact OLED display module with a resolution of 128x64 pixels.

Function: This display shows real-time data from the sensors locally, providing a quick overview of environmental conditions without needing to access the mobile app.

3.2.6 ADS1115

Description: The ADS1115 is a 16-bit analog-to-digital converter (ADC). It provides high precision for reading analog signals.

Function: This ADC enhances the accuracy of sensor data, particularly useful for sensors with analog outputs like the soil moisture sensor.

Software Tools

3.2.7 Firebase

Description: Firebase is a platform developed by Google for creating mobile and web applications. It provides a real-time NoSQL database.

Function: In this project, Firebase is used to store and retrieve sensor data. Its real-time capabilities ensure that the data is updated instantly on the mobile application.

3.2.8 Flutter

Description: Flutter is an open-source UI software development kit created by Google. It is used for developing applications for Android, iOS, Linux, macOS, Windows, and the web from a single codebase.

Function: The mobile application for this project is developed using Flutter. It allows users to access real-time data, receive notifications, and interact with the system seamlessly across different platforms.

3.2.9 Python (for Web Server)

Description: Python is a high-level, interpreted programming language. It is widely used for web development, data analysis, artificial intelligence, and more.

Function: Python is used to set up a web server that acts as an intermediary between the NodeMCU and Firebase. It handles data requests, processes them, and ensures secure and efficient data transmission to the cloud.

By integrating these hardware components and software tools, the smart plant monitoring system is able to provide comprehensive monitoring, real-time data access, and actionable insights to users, ensuring optimal indoor plant health and maintenance.

3.3 Methodology

The methodology section details the step-by-step approach for setting up and conducting the experiment using the smart plant monitoring system. Below is a detailed description:

3.3.1 Setup

The experimental setup involves the following hardware components: NodeMCU (ESP8266) as the primary IoT device, AHT10 for temperature and humidity sensing, BH1750 for light intensity measurement, a soil moisture sensor for soil condition monitoring, 128x64 OLED SSD1306 for local display, and ADS1115 for analog-to-digital conversion. The system architecture is illustrated in *Figure 1*.

The NodeMCU collects data from multiple plants at regular intervals. Each sensor is connected to the NodeMCU according to their respective pin configurations. The NodeMCU is programmed to read sensor data and transmit it to the cloud using Wi-Fi connectivity.

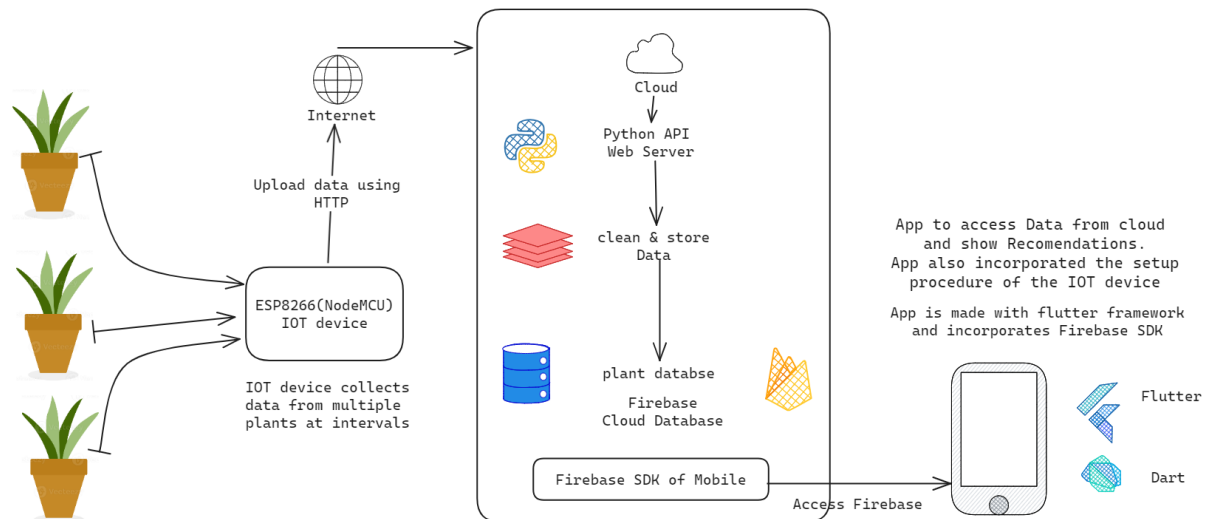


Figure 1: Overall Architecture of the Smart Plant Monitoring System

The provided image (inserted above) illustrates the overall architecture of the system, showing how data is collected by the IoT device, transmitted to the cloud, processed by the Python API, stored in the Firebase database, and accessed by the mobile application.

3.3.2 Data Collection

Data from the sensors (temperature, humidity, light intensity, and soil moisture) is collected at predefined intervals. The NodeMCU gathers this data and temporarily stores it. Each data point includes a timestamp to record when the measurement was taken.

3.3.3 Data Transmission

Collected data is transmitted to the cloud using the HTTP protocol. The NodeMCU connects to a Wi-Fi network and uploads the data to a Python API Web Server hosted on the cloud. The Python API processes, cleans, and stores the data in the Firebase Cloud Database. This approach ensures that data integrity is maintained and that the system remains scalable and reliable for monitoring multiple plants.

To conserve battery life and make the system energy efficient, the NodeMCU utilizes the deep sleep feature. The device enters deep sleep mode after each data transmission, significantly reducing power consumption. It wakes up at one-hour intervals to collect and transmit new data, ensuring that the energy usage is minimized without compromising on data collection frequency.

3.3.4 User Interface

A mobile application developed using the Flutter framework allows users to interact with the system. The app is integrated with Firebase SDK to access and display real-time data from the cloud. Users can view environmental parameters, receive notifications, and get recommendations for optimal plant health. Additionally, the app includes a setup procedure for configuring the IoT device.

4. Proposed Work

4.1 Overview

The proposed work focuses on developing an IoT-based smart plant monitoring system for indoor gardening, leveraging ambient sensors, NodeMCU (ESP8266), and cloud computing for optimal plant health. The system continuously monitors temperature, humidity, light intensity, and soil moisture, transmitting data to a Firebase cloud database via a Python API. Users access real-time data through a Flutter-based mobile app, enabling informed plant care decisions. Designed for scalability and user-friendliness, the system supports monitoring multiple plants and utilizes the NodeMCU's deep sleep feature for energy efficiency. This section details the system architecture, data flow, and strategies for ensuring scalability, reliability, and energy efficiency, addressing the limitations of existing solutions with a more comprehensive approach to indoor gardening.

4.2 System Architecture

The system architecture of the smart plant monitoring system is composed of three main components: the IoT device, the cloud infrastructure, and the mobile application. Each component plays a crucial role in ensuring the efficient monitoring and management of indoor plants. Below is a detailed explanation of each part along with the corresponding flow diagrams.

4.3 IoT Device (Hardware Architecture)

The IoT device, based on the NodeMCU (ESP8266), is responsible for collecting data from various sensors and transmitting it to the cloud. The device is equipped with several sensors: AHT10 for temperature and humidity, BH1750 for light intensity, a soil moisture sensor, and ADS1115 for analog-to-digital conversion. The system utilizes an OLED SSD1306 display for local data visualization.

The NodeMCU (ESP8266) microcontroller is a low-cost Wi-Fi-enabled microcontroller with built-in TCP/IP protocol stack, which is essential for connecting the device to the internet and transmitting data. The device follows a structured operational flow to ensure reliable data collection and energy efficiency.

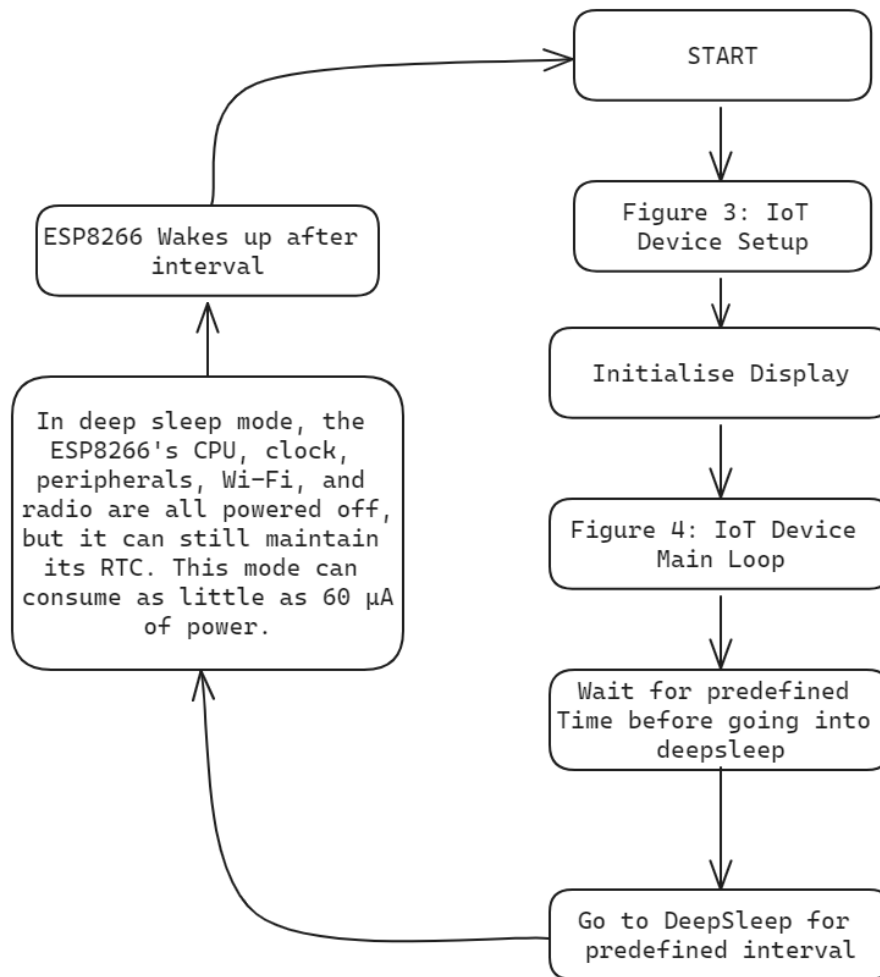


Figure 2: IoT Device Initialization and Deep Sleep Flow

In the initial setup phase, the NodeMCU initializes the display and other peripherals. It then enters the main loop where it waits for a predefined time before going into deep sleep. The deep sleep mode significantly reduces power consumption by shutting down most of the device's components while maintaining the real-time clock (RTC). This mode can consume as little as 60 μA of power. The device wakes up after an interval to collect and transmit new data, ensuring energy efficiency.

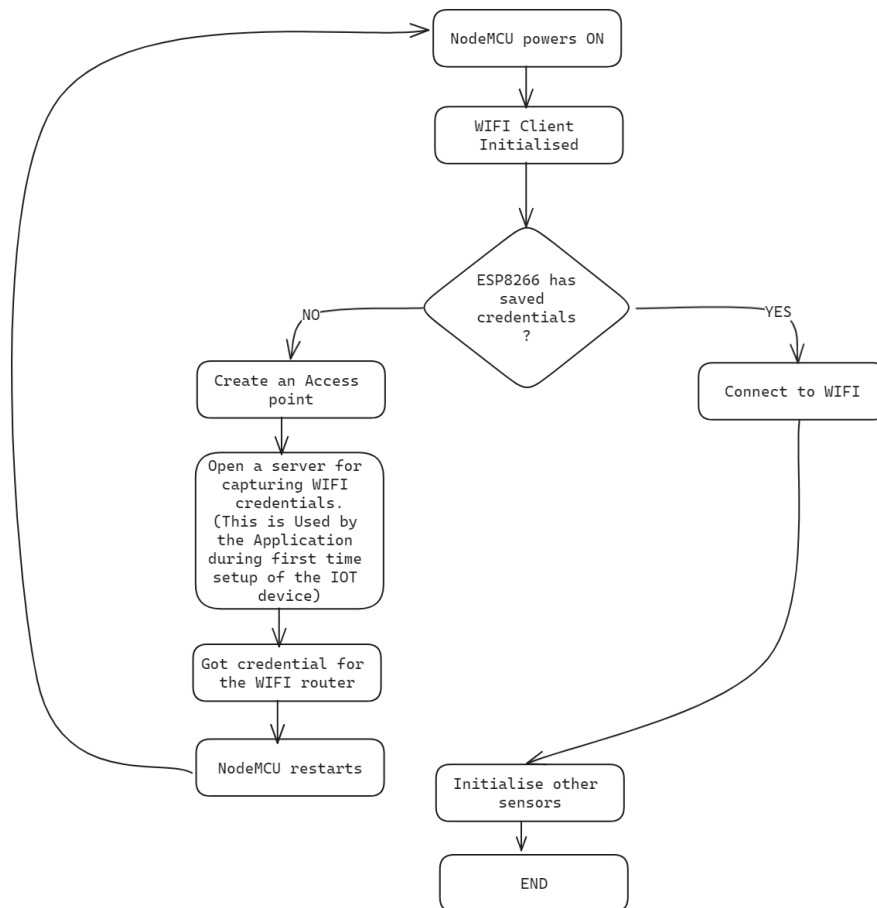


Figure 3: IoT Device Setup

During the setup procedure, the NodeMCU powers on and initializes the Wi-Fi client. It checks for previously saved Wi-Fi credentials in its memory. If credentials are found, it connects to the Wi-Fi network and proceeds to initialize other sensors. If no credentials are found, the NodeMCU creates an access point and opens a server to capture Wi-Fi credentials from the user. Once the credentials are received, the NodeMCU saves them, restarts, and then connects to the Wi-Fi network to continue initializing the other sensors.

The setup also includes setting up the deep sleep parameters. The deep sleep mode significantly reduces power consumption by shutting down most of the device's components while maintaining the real-time clock (RTC). This mode can consume as little as 60 μ A of power, allowing the device to operate on battery power for extended periods.

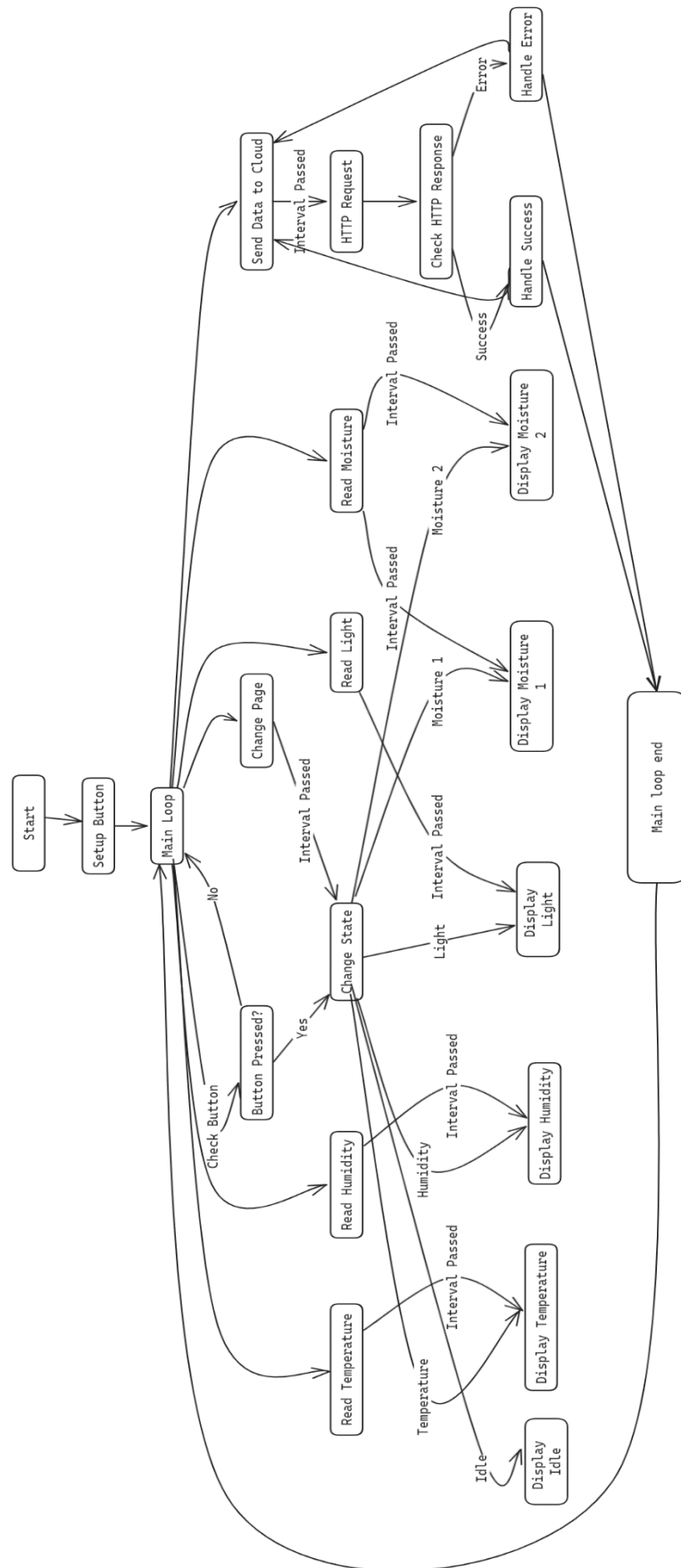


Figure 4: IoT Device Main Loop

The main loop of the IoT device orchestrates the continuous monitoring and data handling processes. After the setup button is initialized, the system enters the main loop:

- **Check Button:** Regularly checks if the setup button is pressed.
 - **Button Pressed?:** If the button is pressed, it triggers a state change to either change the display page or update the display with the latest sensor data.
 - **Change State:** Alternates the display between different sensor readings (temperature, humidity, light, soil moisture).
- **Read Temperature:** Periodically reads the temperature sensor.
 - **Display Temperature:** Updates the OLED display with the current temperature reading.
- **Read Humidity:** Periodically reads the humidity sensor.
 - **Display Humidity:** Updates the OLED display with the current humidity reading.
- **Read Light:** Periodically reads the light intensity sensor.
 - **Display Light:** Updates the OLED display with the current light intensity reading.
- **Read Moisture:** Periodically reads the soil moisture sensors for multiple plants.
 - **Display Moisture 1:** Updates the OLED display with the moisture level of the first plant.
 - **Display Moisture 2:** Updates the OLED display with the moisture level of the second plant.
- **Send Data to Cloud:** Periodically sends collected data to the cloud.
 - **HTTP Request:** Initiates a request to transmit data.
 - **Check HTTP Response:** Evaluates the server's response to confirm successful data transmission.
 - **Handle Success:** Manages successful data transmission.
 - **Handle Error:** Handles any errors encountered during data transmission.

The loop continuously cycles through these operations, ensuring that sensor data is accurately read, displayed, and transmitted, thereby maintaining efficient plant monitoring and management.

The device periodically wakes up from deep sleep to collect and transmit new data. The wake-up interval is configurable, but for this project, it is set to one hour. Upon waking up, the device performs the following steps:

- **Initialization:** The device initializes its peripherals, including the display and sensors.
- **Sensor Data Collection:** The device reads data from the AHT10 (temperature and humidity), BH1750 (light intensity), and soil moisture sensor. The ADS1115 is used to convert the analog signals from the soil moisture sensor to digital values.
- **Data Processing:** The collected data is processed and formatted for transmission. This involves converting raw sensor readings into meaningful values (e.g., converting the ADC value to soil moisture percentage).
- **Data Transmission:** The processed data is sent to the cloud using the HTTP protocol. The NodeMCU connects to the Wi-Fi network and uploads the data to the Python web server, which then updates the Firebase database.
- **Deep Sleep:** After transmitting the data, the device goes back into deep sleep mode to conserve energy.

4.4 Cloud Infrastructure

The cloud infrastructure is built using Firebase[4] and a Python-based web server. The NodeMCU sends data to the cloud where it is processed and stored. The server is responsible for handling incoming data, validating it, and updating the Firebase database.

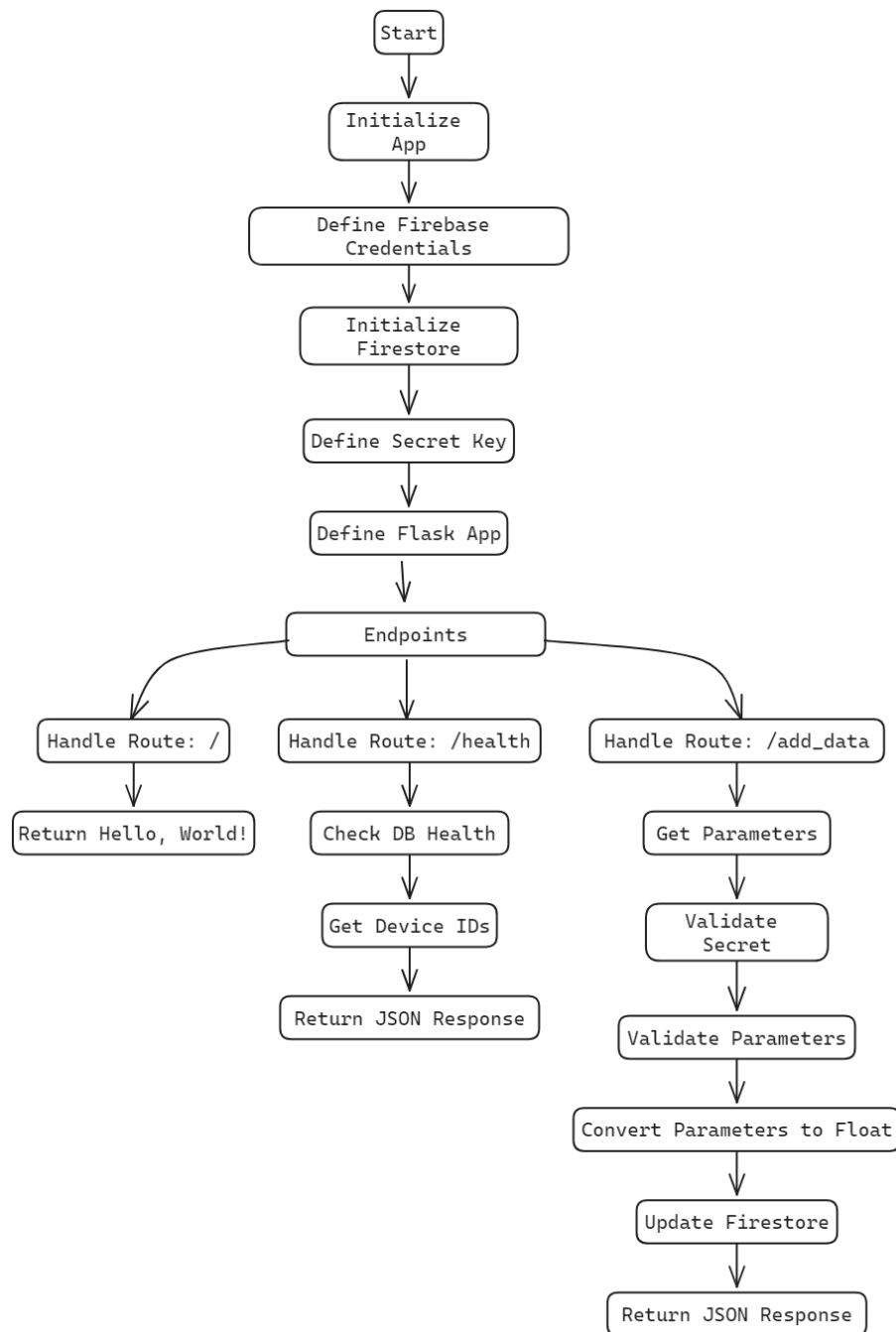


Figure 5: Python Web Server Architecture

The Python web server architecture is designed to facilitate communication between the IoT device and the cloud, ensuring data is efficiently processed and stored in Firebase. The server initialization process begins with setting up the application and defining Firebase credentials. These credentials are crucial for securely connecting to Firebase services. Once the

credentials are in place, Firestore is initialized to handle the database operations.

A secret key is then defined to ensure secure communication between the server and its clients. This key is used to validate incoming requests, adding a layer of security to the data handling process. The Flask application is set up next, providing the framework for defining various endpoints that the server will handle.

The server is equipped with multiple endpoints to manage different types of requests:

- The root endpoint (/) handles basic requests and returns a simple "Hello, World!" message. This is primarily used to verify that the server is running correctly.
- The **/health** endpoint is designed to check the health of the database. When this endpoint is accessed, the server retrieves the device IDs from Firestore and returns a JSON response indicating the status of the database. This allows for monitoring the connectivity and health of the database.
- The **/add_data** endpoint is more complex and handles incoming data from the IoT device. When a request is made to this endpoint, the server first retrieves the parameters from the request. It then validates a secret key to ensure that the request is authorized. After the secret key is validated, the server proceeds to validate the parameters received. The parameters are then converted to float values to ensure they are in the correct format for storage. Finally, the server updates Firestore with the new data and returns a JSON response confirming that the data has been successfully updated.

This structured approach ensures that the Python web server can efficiently manage incoming data, validate it, and update the Firebase database, supporting the overall functionality and reliability of the smart plant monitoring system. The architecture is designed to be robust and secure, ensuring that data integrity and security are maintained throughout the process.

4.4.1 Database Structure

The database in Firebase is structured into three main collections: “encyclopedias”, “smartplants”, and “users”. Each collection serves a unique purpose in the overall system, ensuring that the data is organized and accessible for the application's functionality.

Encyclopedias Collection

The **encyclopedias** collection is dedicated to storing detailed information about various indoor plants. Each document within this collection represents a different plant and contains several fields. For instance, a document might look like this:

```
{
  "common_name": "Money Plant Golden",
  "ideal_humidity": {
    "max": 80,
    "min": 50
  },
  "ideal_moisture": {
    "max": 70,
    "min": 50
  },
  "ideal_temperature": {
    "max": 30,
    "min": 20
  },
  "image": "https://i.ibb.co/m0TFPt8/Money-Plant-Golden.png",
  "insight_humidity": "allowing the top layer to dry out before watering again",
  "insight_sunlight": "Needs Bright Indirect Sunlight",
  "insight_water": "Water Twice A Week",
  "scientific_name": "Epipremnum aureum 'Golden'"
}
```

In this document, fields such as **common_name**, **scientific_name**, and **image** are strings that provide basic identification and visual reference for the plant. The **ideal_humidity**, **ideal_moisture**, and **ideal_temperature** fields are maps containing numerical values that define the optimal environmental conditions for the plant. Additional fields like **insight_humidity**, **insight_sunlight**, and **insight_water** provide practical care tips.

Smartplants Collection

The **smartplants** collection contains data about all the active IoT devices. Each document in this collection corresponds to a specific device, identified by its device ID. An example document might look like this:

```
{
  "id": "2401",
  "last_data": {
    "hum": 40.63,
    "light": 63.33,
    "soil": 0,
    "soil2": 0,
    "temp": 33.2
  },
  "last_updated": 1714932636930,
  "raw_data": [
    {
      "data": {
        "hum": 54,
        "light": 26,
        "soil": 0,
        "soil2": 0,
        "temp": 32,
        "timestamp": 1713530753668
      }
    }
  ]
}
```

In this document, the **id** field is a string that uniquely identifies the device. The **last_data**

field is a map that stores the most recent sensor readings, including humidity (**hum**), light intensity (**light**), soil moisture (**soil** and **soil2**), and temperature (**temp**). The **last_updated** field is a numerical timestamp indicating the last time the data was updated. Additionally, the **raw_data** field is an array of historical sensor data, each entry containing a map of readings and a timestamp.

Users Collection

The **users** collection stores information related to users and includes an inner collection called **smartplants** that details the IoT devices associated with each user and the specific plants they are monitoring. An example structure might look like this:

```
{
  "enc_id": "6U36tRc1ugwXo37IfxLK",
  "id": "2401",
  "name": "my plant",
  "plant_id": "2401-1",
  "pot_no": "1"
}
```

In this inner collection, each document includes an **enc_id** that links to the plant's encyclopedia entry, an **id** that corresponds to the device ID, a **name** for the plant as chosen by the user, a **plant_id** that uniquely identifies the plant within the user's collection, and a **pot_no** indicating the pot number. This structure allows the system to relate user-specific plant data with the general plant information stored in the **encyclopedias** collection.

Overall, this database structure ensures efficient organization and retrieval of data, supporting the application's functionality in managing plant care through IoT devices.

4.5 Mobile Application

The mobile application, developed using Flutter, allows users to interact with the system, access real-time data, and receive recommendations for optimal plant care. The app communicates with the Firebase database to fetch data and display it in a user-friendly interface.

4.5.1 IOT Device Pairing Flow

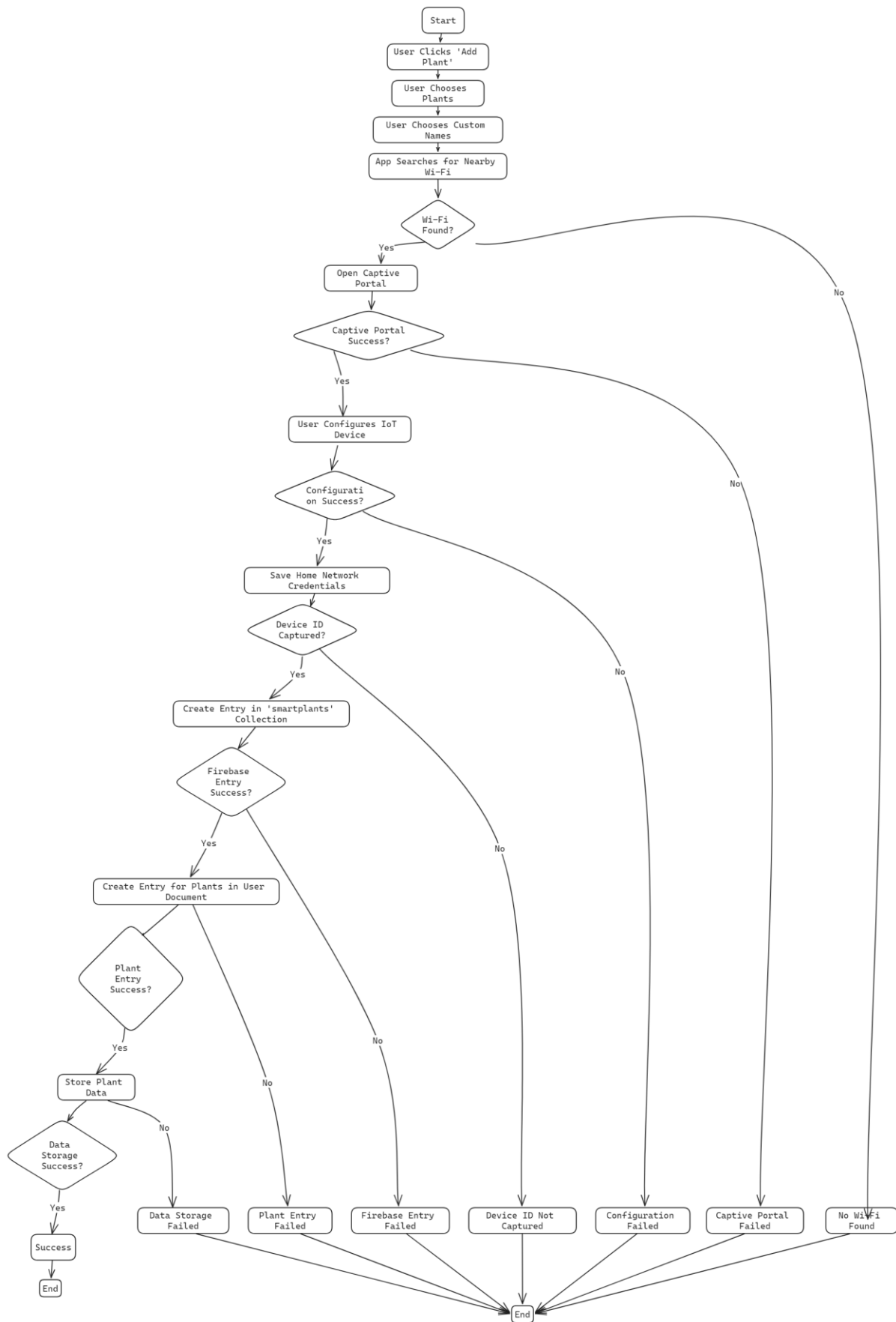


Figure 6: IOT device setup App Flow

This App flow outlines the process within the mobile application for setting up a new IoT device, specifically designed for plant care. The user initiates the process by clicking "Add Plant" and choosing the desired plants, followed by assigning custom names to each plant. The app then searches for nearby Wi-Fi networks.

If a Wi-Fi network is found, the app attempts to open a captive portal to facilitate the configuration of the IoT device. Upon successful opening of the captive portal, the user configures the IoT device. The app verifies the success of the configuration process before saving the home network credentials.

Next, the app checks if the device ID has been captured. If successful, an entry is created in the 'smartplants' collection in Firebase. The success of this Firebase entry is then verified. If successful, an entry for the plant is created in the user's document.

Finally, the app attempts to store plant data. If this data storage is successful, the setup process is complete, and the user is notified of the success. If any step fails (such as no Wi-Fi found, captive portal failure, configuration failure, device ID not captured, Firebase entry failure, or plant entry failure), the process ends with an appropriate error notification.

After completing the setup flow for configuring a new IoT device, users can access a variety of features within the mobile application. The app, developed using Flutter, offers a user-friendly interface that allows users to interact with their plant monitoring system, view real-time data, and receive personalized care recommendations.

4.5.2 Authentication

The app supports user authentication through login and sign-up methods using email. This ensures that each user's data is securely stored and accessible only to them. Users can create a new account or log in to an existing one to access their plant data and personalized recommendations.

4.5.3 Home Screen

The home screen presents a carousel of plants that the user owns. This carousel is populated by fetching plant data from the Firebase database. Each plant card in the carousel displays the plant's name and a thumbnail image, providing a quick overview for the user.

4.5.4 Plant Details

By clicking on a plant in the carousel, users are taken to a detailed view of that specific plant. This screen displays real-time data collected by the IoT device, such as:

- **Humidity**
- **Light Intensity**
- **Soil Moisture**
- **Temperature**

In addition to real-time data, the app fetches the ideal conditions for the plant from the **encyclopedias** collection in the Firebase database. It then compares the current readings with these ideal conditions and provides visual indicators to show how the plant's environment measures up. For example, if the current humidity is below the ideal range, the app might highlight this in red to alert the user.

4.5.5 Data Visualization

Scrolling down from the real-time data section, users can view graphical representations of various environmental factors over time. These graphs display historical data points for:

- **Temperature**
- **Humidity**
- **Light Intensity**
- **Soil Moisture**

These graphs allow users to track trends and monitor the health of their plants over time, providing valuable insights into their care routine.

4.5.6 AI-Generated Recommendations

For users seeking personalized advice, the app offers AI-generated recommendations. This feature sends the past 10 data points to the OpenAI API (GPT-3.5) and receives a response formatted in JSON. The app then processes this response and displays it to the user in a clear and actionable format. These recommendations might include suggestions on watering frequency, light adjustments, and other care tips tailored to the specific needs of the plant based on the historical data.

5. Experiment Results

5.1 Experimental Setup

The experimental setup section describes the configuration and deployment of the smart plant monitoring system. This section includes a detailed account of the hardware assembly, schematic diagrams, and the mobile application interface that enables user interaction with the system.

5.1.1 IoT Device Setup:

Below are the diagrams illustrating the breadboard connections and the schematic of the IoT device setup.

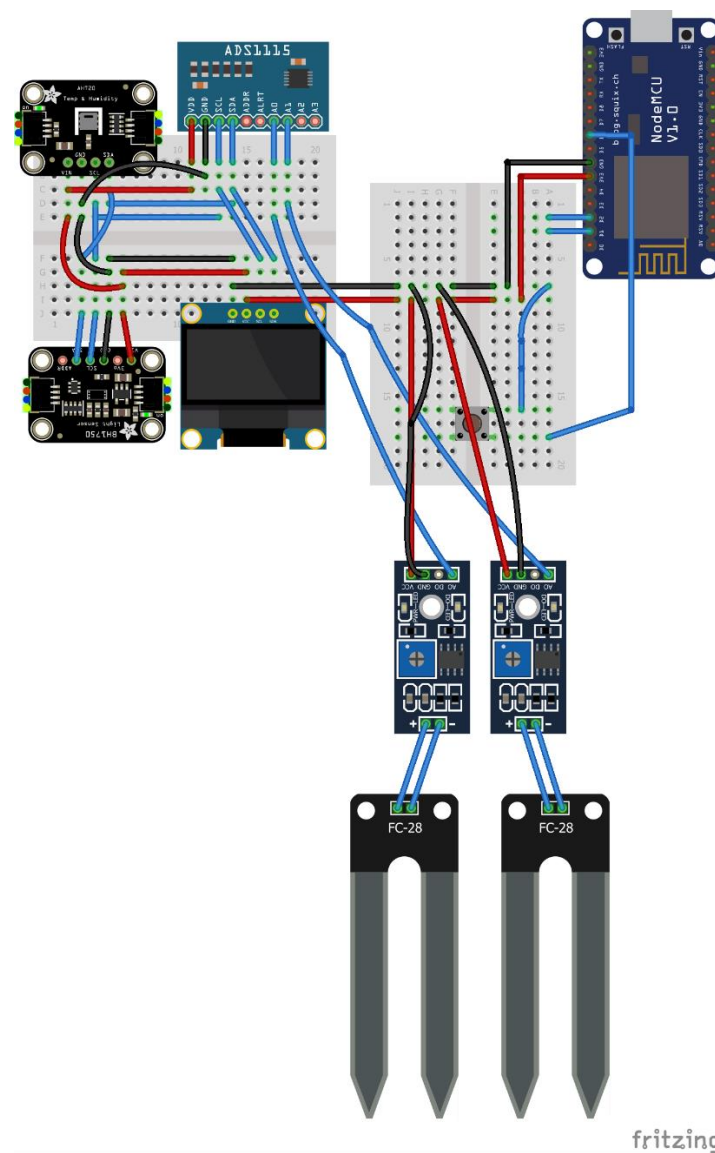


Figure 7: Breadboard Diagram of the IoT Device Setup

This breadboard diagram shows the connections between the NodeMCU and various sensors, including the AHT10, BH1750, soil moisture sensors, ADS1115 ADC, and the OLED

display. Each component is connected via the I2C bus to streamline data collection and ensure efficient communication between devices.

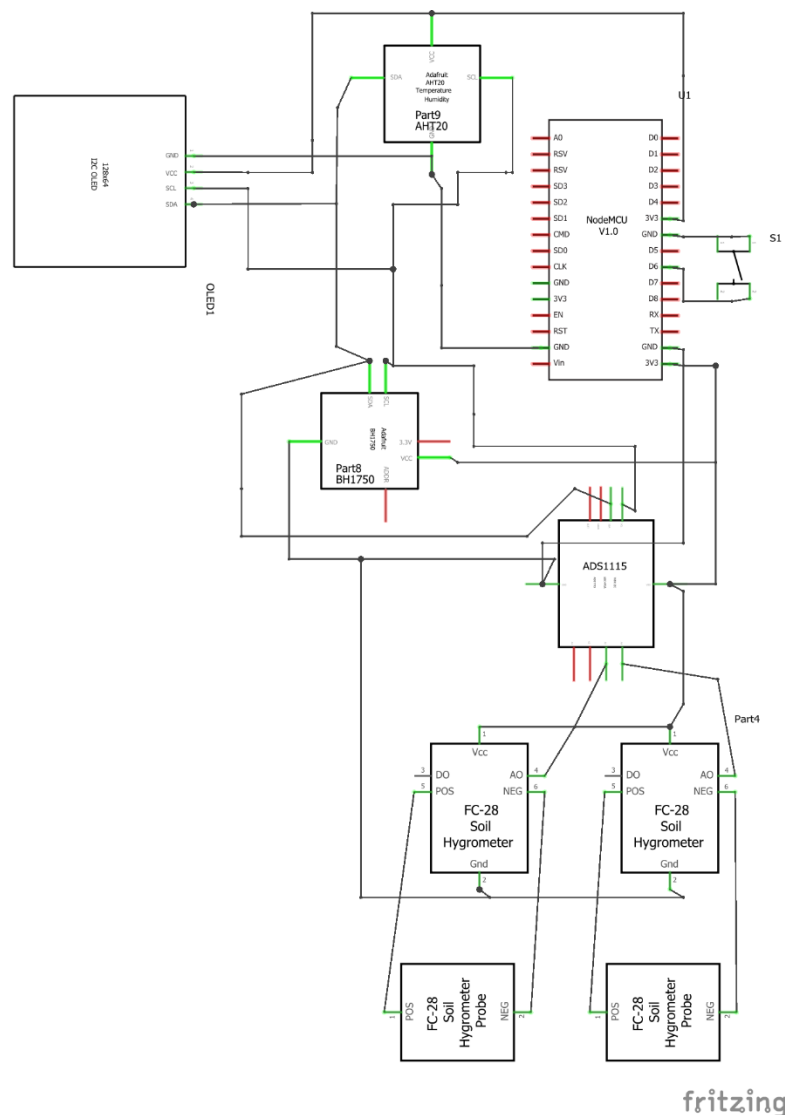
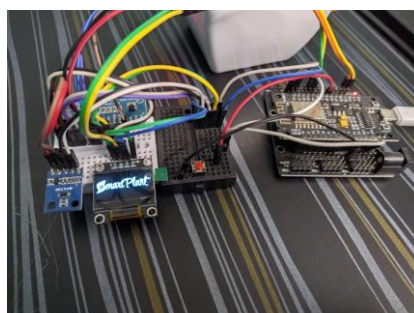


Figure 8: Schematic Diagram of the IoT Device Setup

This schematic diagram provides a detailed view of the circuit connections for the IoT device setup. It includes the wiring for the AHT10 temperature and humidity sensor, BH1750 light intensity sensor, soil moisture sensors, ADS1115 ADC, OLED display, and the NodeMCU. The schematic highlights the integration of the components using the I2C bus, which facilitates efficient data transmission and communication.



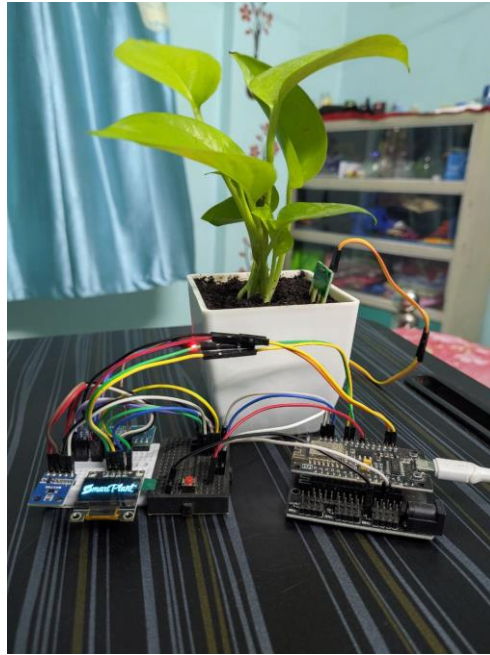


Figure 9: Photograph IoT Device Setup

The experimental setup ensures accurate data collection and efficient power management, enabling continuous monitoring and maintenance of optimal conditions for indoor gardening plants.

5.1.2 Mobile App Screenshot

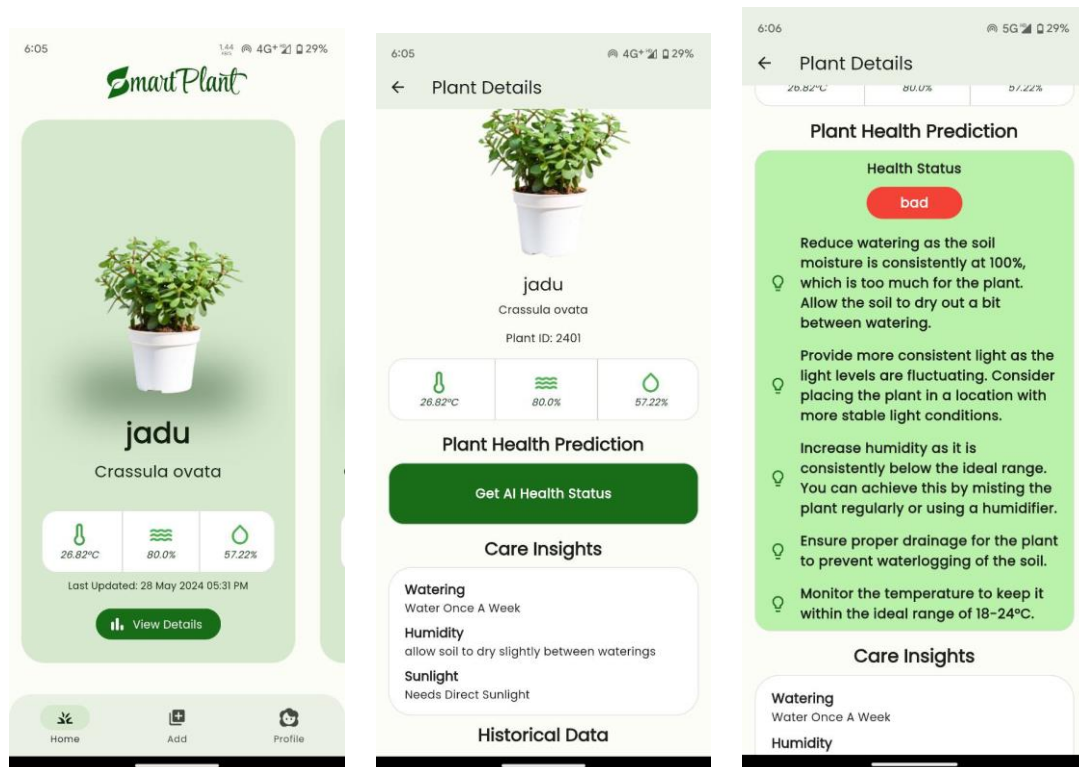


Figure 11: Screenshot of the Mobile Application

5.2 Environmental Data Analysis

In this section, we present and analyze the environmental data collected by the smart plant monitoring system on 29/5/2024. The graphs below illustrate the variations in humidity, temperature, and soil moisture over time, providing insights into the environmental conditions experienced by the indoor plants.

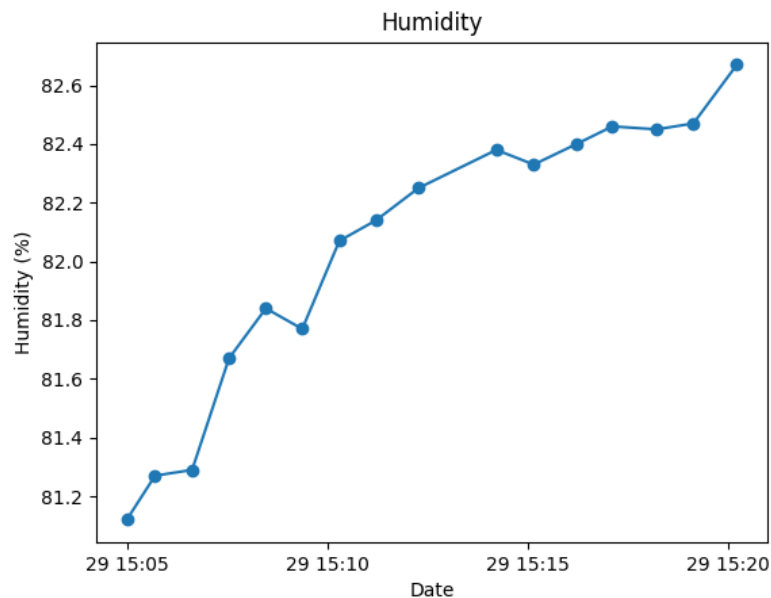


Figure 12: Humidity Levels on 29/5/2024 Monitored by the AHT10 Sensor

Humidity Graph: This graph displays the changes in humidity levels measured by the AHT10 sensor, highlighting the system's ability to track and respond to fluctuations in the indoor environment.

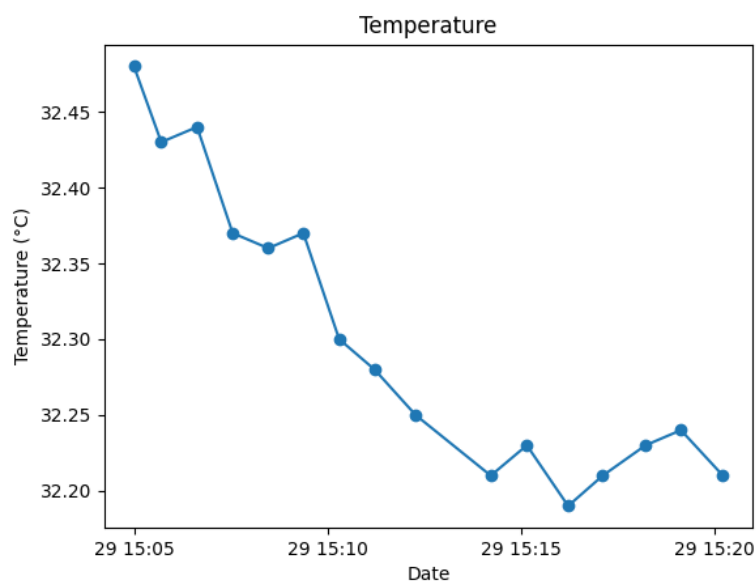


Figure 13: Temperature Readings on 29/5/2024 Monitored by the AHT10 Sensor

Temperature Graph: The temperature graph shows the readings from the AHT10 sensor, demonstrating the system's capability to monitor and maintain optimal temperature conditions for plant health.

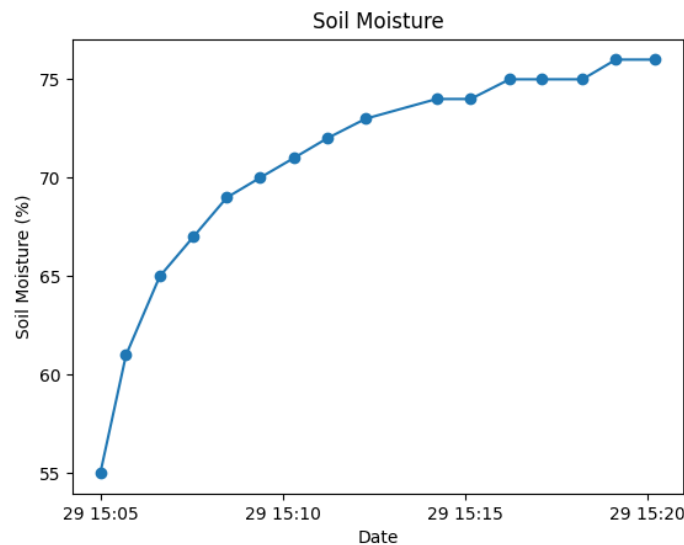


Figure 14: Soil Moisture Levels on 29/5/2024 Monitored by the Soil Moisture Sensor

Soil Moisture Graph: The soil moisture graph captures the moisture levels detected by the soil moisture sensor, indicating the effectiveness of the system in ensuring adequate water supply to the plants.

5.3 Power Consumption Analysis

The power consumption of the IoT device is a critical factor for ensuring its sustainability and effectiveness in long-term indoor gardening applications. The NodeMCU (ESP8266) microcontroller, equipped with the deep sleep feature, is used to minimize power usage when the device is inactive.

<i>Time (s)</i>	<i>NodeMCU (mA)</i>	<i>BH1750 (mA)</i>	<i>Soil Moisture (mA)</i>	<i>AHT10 (mA)</i>	<i>Display (mA)</i>	<i>ADS1115 (mA)</i>	<i>Total Current (mA)</i>
0	152	0.11	97.39	0.0012	10.05	0.14	259.6812
1	71	0.12	96.89	0.00122	10.1	0.15	178.16122
2	69	0.13	96.29	0.00124	10.15	0.16	175.73124
3	151	0.14	97.19	0.00126	10.2	0.17	258.70126
4	0.024	0.02	0.02	0.02	0.02	0.02	0.024
5	0.026	0.02	0.02	0.02	0.02	0.02	0.026
6	153	0.15	98.09	0.00128	10.25	0.18	261.68128
7	72	0.16	97.59	0.0013	10.3	0.19	180.2413
8	68	0.17	96.99	0.00132	10.35	0.2	175.71132
9	149	0.18	97.89	0.00134	10.4	0.21	257.67134
10	0.027	0.02	0.01	0.01	0.01	0.01	0.027
11	0.028	0.00	0.01	0.00	0.02	0.02	0.028

Table 1: Current Consumption of IoT Components in Various States

This table represents the power consumption of each component at different times (in seconds). The “Total Current” column is the sum of the currents of all components at a given time.

- At time 0s, 6s, the NodeMCU is starting up and all sensors are reading data. The display is on.
- At time 1s, 2s, 7s, 8s, the NodeMCU is idle, all sensors are still reading data, and the display is on.
- At time 3s, 9s, the NodeMCU is transferring data through WiFi, all sensors are reading data, and the display is on.
- At time 4s, 5s, 10s, 11s, the NodeMCU is in deep sleep, all sensors are off, and the display is off.

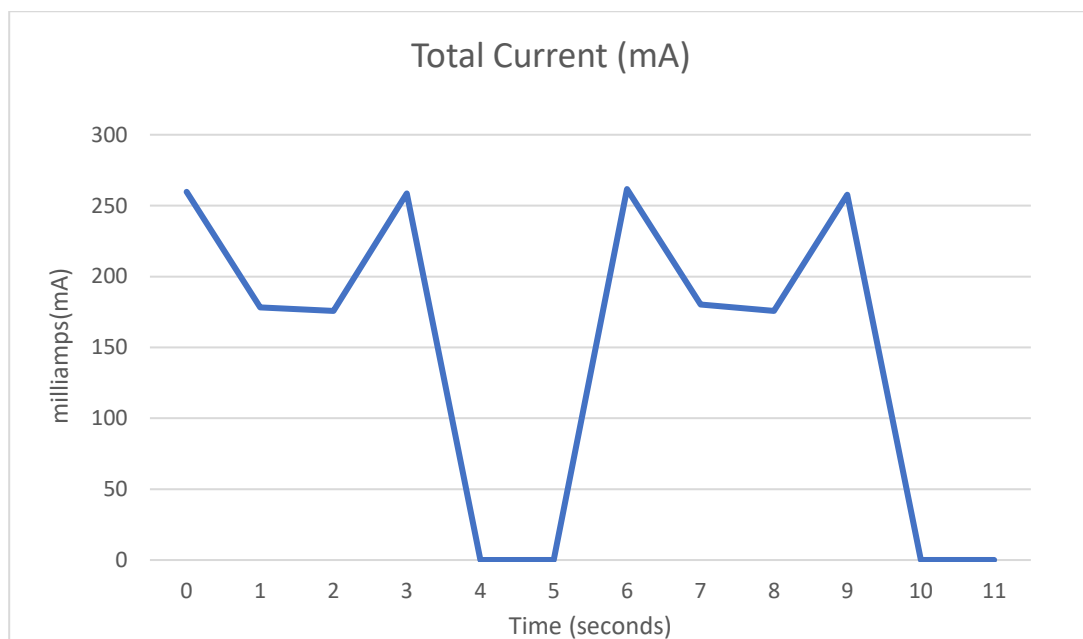


Figure 11: chart showing Current Consumption of IoT Components in Various States

This setup simulates the various states the IoT device goes through, including startup, idle, data transmission, and deep sleep. In reality, for power consumption calculations, the device is programmed to wake up for **30 seconds**, perform its tasks, and then go into deep sleep mode for **60 minutes**. This cycle is repeated to ensure energy efficiency while providing periodic updates on the plant's conditions.

5.3.1 Deep Sleep Mode

The deep sleep mode significantly reduces power consumption by turning off the majority of the NodeMCU's components and only maintaining the necessary circuitry to keep the real-time clock running. During this mode, the device consumes only 60 microamps of current, which is a substantial reduction compared to its active state.

By averaging the current consumption across deepsleep states, we can estimate the typical sleep mode current usage. The average current usage during active mode is approximately

0.025 mA.

5.3.2 Active Mode Consumption

In active mode, the IoT device cycles through several states, each with distinct power consumption characteristics. These states are:

- **Startup:** The NodeMCU initializes, and all sensors begin reading data. The display is turned on.
- **Idle:** The NodeMCU remains on but is not performing any significant computations. All sensors continue to read data, and the display remains on.
- **Data Transmission:** The NodeMCU sends the collected data via WiFi to the cloud server. All sensors are still reading data, and the display is on.

In active mode, the NodeMCU and the connected sensors consume more power as they perform data collection, processing, and transmission tasks. Each sensor has its own power requirements, and the combined consumption is taken into account to determine the overall power usage during the active phase.

By averaging the current consumption across all active states (startup, idle, and data transmission), we can estimate the typical active mode current usage. The average current usage during active mode is approximately **250 mA**.

5.3.3 Power Consumption Cycle

To estimate the power consumption per cycle, we consider both the active and deep sleep modes of the NodeMCU. For battery life estimation, we use a battery with a capacity of 1400 mAh.

Let's denote the following variables:

- I_{wake} : Average current during the wake – up period (250 mA)
- T_{wake} : Duration of the wake – up period (30 seconds)
- I_{sleep} : Current during the deep sleep period (0.025 mA)
- T_{sleep} : Duration of the deep sleep period (3600 seconds)
- **Battery_capacity**: Capacity of the battery (1400 mAh)

The total current during wake in 1 cycle is:

$$I_{total_wake} = \frac{I_{wake} \times T_{wake}}{3600}$$

Substituting the given values:

$$I_{total_wake} = \frac{250 \text{ mA} \times 30 \text{ s/cycle}}{3600} \approx 2.08 \text{ mAh}$$

The total current during deepsleep in 1 cycle is:

$$I_{\text{total_sleep}} = \frac{I_{\text{sleep}} \times T_{\text{sleep}}}{3600}$$

Substituting the given values:

$$I_{\text{total_sleep}} = \frac{0.025 \text{ mA} \times 3600 \text{ s/cycle}}{3600} \approx 0.025 \text{ mAh}$$

Therefore the total consumption in 1 cycle is given by:

$$I_{\text{total}} = I_{\text{total_wake}} + I_{\text{total_sleep}} \approx 2.105 \text{ mAh}$$

5.3.4 Battery Life Estimation

To estimate the battery life, the power consumption in both active and deep sleep modes is calculated. By understanding the duration and frequency of each mode, the total energy consumed over a specific period can be determined. This information is used to estimate how long a particular battery will last before needing a recharge or replacement.

Now, let's calculate how many cycles the device will last for a 1400mAh battery:

$$\text{cycles} = \frac{\text{Battery_capacity}}{I_{\text{total}}}$$

Substituting the given values:

$$\text{cycles} = \frac{1400 \text{ mAh}}{2.105 \text{ mAh}} \approx 665 \text{ cycles}$$

$$1 \text{ cycle consists of } 3600 + 30 = 3630 \text{ seconds} \approx 1 \text{ hour}$$

$$\text{Battery life} = 665 \text{ hours} = 27.708 \text{ days} \approx \mathbf{27 \text{ days}}$$

Given the calculations, a single cycle of the NodeMCU (30 seconds active, 60 minutes deep sleep) results in an average hourly power consumption of approximately 2.105 mAh. With a 1400 mAh battery, the estimated battery life for the IoT device is approximately **27 days**. This estimation assumes consistent power consumption across the specified active and deep sleep periods.

These findings highlight the importance of power-efficient design in our IoT device, emphasizing the need for optimizing both active and sleep states to extend battery life. This estimation provides a practical insight into the operational longevity of the device under typical usage conditions.

6. Conclusions and Future work

6.1 Conclusion

The development of a smart plant monitoring system tailored for indoor gardening integrates IoT technology and cloud computing to provide a robust, scalable, and user-friendly solution for plant care. High-precision sensors with the NodeMCU microcontroller ensure accurate monitoring of temperature, humidity, light intensity, and soil moisture, promoting plant health and growth. The system efficiently monitors multiple plants, while the mobile app offers real-time data and recommendations, enhancing user decision-making. The NodeMCU's deep sleep feature reduces power consumption, making the system environmentally responsible and cost-effective. Overall, this project overcomes existing limitations, offering a comprehensive and efficient approach to indoor gardening.

6.2 Future Work

Building on the current success of the smart plant monitoring system, several avenues for future development can further enhance its functionality and user experience:

1. **Enhanced Sensor Integration:**

- Future iterations could incorporate additional sensors to monitor parameters such as CO₂ levels and pH, providing a more comprehensive overview of the plant environment.

2. **Enhanced Mobile Application Features:**

- Expanding the mobile application to include features such as push notifications for critical alerts, detailed plant care guides, and integration with virtual assistants (e.g., Alexa, Google Assistant) could significantly enhance user engagement and convenience.

By pursuing these future directions, the smart plant monitoring system can evolve into a more advanced, comprehensive, and user-friendly solution, continuing to push the boundaries of indoor gardening technology and contributing to the well-being of plants and their caretakers.

7. References

- [1] Grand View Research. (2021). "Indoor Farming Market Size, Share & Trends Analysis Report By Facility Type (Greenhouses, Indoor Vertical Farms), By Component (Hardware, Software), By Crop Category, By Region, And Segment Forecasts, 2021 - 2028." Retrieved from <https://www.grandviewresearch.com/industry-analysis/indoor-farming-market>
- [2] United Nations Department of Economic and Social Affairs. (2018). "68% of the world population projected to live in urban areas by 2050, says UN." Retrieved from <https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html>
- [3] Market Research Future, "Global IoT in Agriculture Market", [Online]. Available: <https://www.marketresearchfuture.com/reports/iot-in-agriculture-market-5642>
- [4] Firebase, "Firebase Documentation", [Online]. Available: <https://firebase.google.com/docs>
- [5] Google Developers, "Flutter: Beautiful native apps in record time", [Online]. Available: <https://flutter.dev/>
- [6] NodeMCU, "NodeMCU Documentation", [Online]. Available: <https://nodemcu.readthedocs.io/>
- [7] Aosong Electronics, "AHT10: Temperature & Humidity Sensor", [Online]. Available: <https://www.aosong.com/en/products-40.html>
- [8] Adafruit, "BH1750: Digital Ambient Light Sensor", [Online]. Available: <https://learn.adafruit.com/adafruit-bh1750-ambient-light-sensor/overview>
- [9] SparkFun, "Soil Moisture Sensor", [Online]. Available: <https://www.sparkfun.com/products/13322>
- [10] Adafruit, "128x64 OLED SSD1306", [Online]. Available: <https://www.adafruit.com/product/326>
- [11] Palande, Vaibhav and Zaheer, Adam and George, Kiran, Fully automated hydroponic system for indoor plant growth, Procedia Computer Science, Vol-129, pg-482-488,2018,publ-Elsevier
- [12] Elijah, Olakunle and Rahman, Tharek Abdul and Orikumhi, Igbafe and Leow, Chee Yen and Hindia, MHD Nour, An overview of Internet of Things (IoT) and data analytics in agriculture: Benefits and challenges, IEEE Internet of things Journal, vol-5, pg-3758—3773,2018,publ-IEEE
- [13] Ayaz, Muhammad and Ammad-Uddin, Mohammad and Sharif, Zubair and Mansour, Ali and Aggoune, El-Hadi M, Internet-of-Things (IoT)-based smart agriculture: Toward making the fields talk, IEEE access, vol-7, Pg-129551—129583, 2019,publ-IEEE

[14] Ezhilazhahi, AM and Bhuvaneswari, PTV, IoT enabled plant soil moisture monitoring using wireless sensor networks, 2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS),pg-345-349, 2017, organization-IEEE