ESPRIT - Private Higher School of Engineering and Technology

ACADEMIC INTERNSHIP REPORT

Design and Implementation of an IoT-Based System for Agricultural Monitoring

Author:
Imed Hamdene
Class: 4SLEAM

Internship Supervisors:
Ameni Driss
Amel Bennicira

Hosting Organization: ESPRIT

Acknowledgements

First and foremost, I extend my deepest thanks to my internship supervisors, Mrs. Ameni Driss and Mrs. Amel Bennicira, for their invaluable mentorship, their insightful advice, and their constant availability throughout this project. Their expertise and encouragement were crucial in navigating the technical challenges and shaping the direction of this work.

I would also like to thank the entire team at ESPRIT for welcoming me and for creating a stimulating and collaborative work environment.

Finally, I wish to thank my family and friends for their unwavering support and encouragement during this period. *At the conclusion of this internship project, I would like to express my sincere gratitude to several individuals whose guidance and support were instrumental to my success.*

First and foremost, I extend my deepest thanks to my internship supervisors, Mrs. Ameni Driss and Mrs. Amel Bennicira, for their invaluable mentorship, their insightful advice, and their constant availability throughout this project. Their expertise and encouragement were crucial in navigating the technical challenges and shaping the direction of this work.

I would also like to thank the entire team at ESPRIT for welcoming me and for creating a stimulating and collaborative work environment.

Finally, I wish to thank my family and friends for their unwavering support and encouragement during this period.

Abstract

This report details the design, implementation, and validation of an integrated IoT system for smart agricultural applications. The primary objective was to develop a low-cost, automated solution for collecting both environmental and visual data from a target crop area. The system architecture is composed of two main data acquisition units: a stationary monitoring station and a mobile autonomous vehicle. The stationary node utilizes an ESP32 microcontroller to periodically collect data on air temperature, humidity, soil moisture, and ambient light, transmitting this information via the MQTT protocol. The autonomous vehicle, controlled by an ESP32-S3, is equipped with ultrasonic sensors for collision avoidance and an ESP-CAM for capturing and sending plant images over HTTP. All data is aggregated by a Node-RED backend, which provides a real-time web dashboard for visualization and logs all information to a MongoDB cloud database for historical analysis. The resulting prototype successfully demonstrates an effective and scalable solution for comprehensive, data-driven agricultural monitoring.

Contents

List of Figures

Chapter 1

General Context and Project Objectives

1.1 Introduction

In recent years, the agricultural sector has faced increasing pressure to enhance productivity while simultaneously promoting sustainability. The rise of Smart Agriculture, which leverages modern information and communication technologies, offers a powerful solution to these challenges. At the heart of this revolution is the Internet of Things (IoT), a network of interconnected devices and sensors that can collect and exchange data. By deploying IoT systems in an agricultural context, it becomes possible to monitor critical environmental parameters and plant health with unprecedented precision. This data-driven approach allows for the optimization of resources such as water and fertilizer, the early detection of crop stress or disease, and the automation of labor-intensive tasks, ultimately leading to improved crop yields and more sustainable farming practices.

1.2 Problem Statement

Traditional methods for monitoring agricultural fields often rely on manual inspection and periodic data collection. These methods are not only labor-intensive and time-consuming but can also lead to inconsistent and infrequent data, making it difficult to respond quickly to changing conditions. A farmer might miss the early signs of a pest infestation or apply water inefficiently due to a lack of precise soil moisture data. Therefore, there is a clear need for an automated, cost-effective system capable of gathering both comprehensive environmental data and detailed visual plant health information in a reliable and real-time manner.

1.3 Project Objectives

To address the problem statement, this project aims to design, build, and validate a complete IoT-based agricultural monitoring system. The primary objectives are defined as follows:

- *Design and construct an autonomous ground vehicle capable of navigating a designated area and capturing images of plants upon command.*
- *Develop a stationary monitoring unit equipped with sensors to continuously collect key environmental data, including air temperature, humidity, ambient light, and soil moisture.*

- *Implement a robust, dual-protocol communication system using MQTT for lightweight sensor data and HTTP for heavy image data transmission.*
- *Create a centralized, web-based dashboard for the real-time visualization of all incoming sensor data and images.*
- *Set up a scalable cloud database to log all collected data for historical analysis and future use.*

1.4 System Architecture

The system is designed with a modular, distributed architecture, as illustrated in Figure 1.1. It consists of two distinct data acquisition "nodes"—the mobile vehicle and the stationary sensor station. These nodes operate independently but communicate with a central backend server. The backend, built with Node-RED, is responsible for receiving all data, routing it to a MongoDB cloud database for storage, and presenting it to the end-user via a real-time dashboard. This architecture ensures that the system is scalable and resilient.

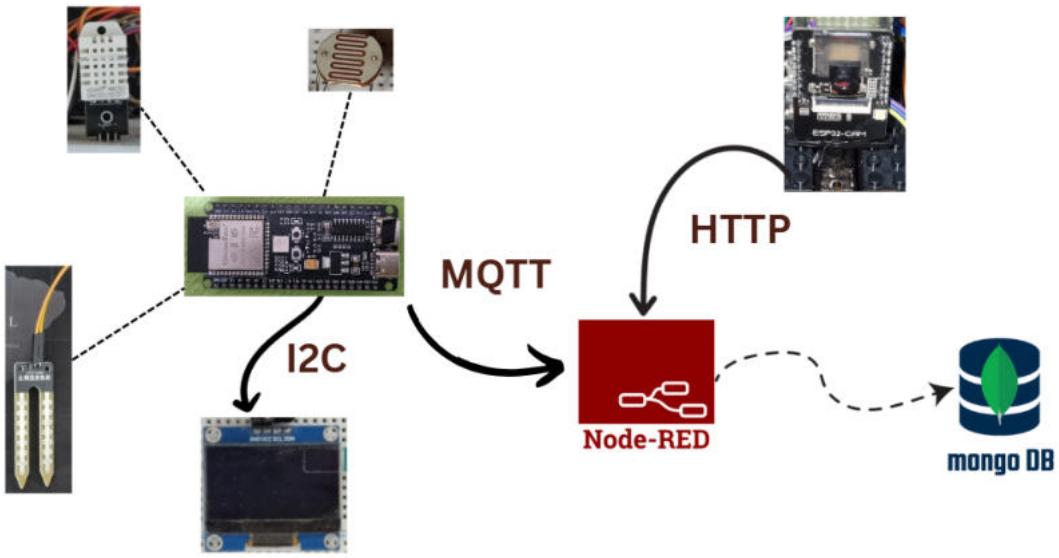

Figure 1.1: High-Level System Architecture

1.5 Project Planning (Gantt Chart)

The project was executed over a period of [e.g., eight weeks], following a structured plan to ensure timely completion of all objectives. The project timeline, broken down into key phases such as research, hardware assembly, software development, and testing, is visualized in the Gantt chart shown in Figure 1.2. This structured approach allowed for methodical progress and effective time management throughout the internship.

Phase	Task Name	Jul 1–7	Jul 8–14	Jul 15–21	Jul 22–28	Jul 29–Aug 4	Aug 5–11	Aug 12–18	Aug 19–25
Research & Design	1.Project Definition & Research	■							
	2.Component Selection & System Arch		■						
Development & Assembly	3.Monitoring Station: Circuit & Code			■					
	4.Autonomous Vehicle: Assembly & Code				■	■			
Backend & Integration	5.Node-RED & MongoDB Setup						■		
	6.Full System Integration & Testing							■	
Finalization	7.Final Validation & Report Writing								■

Figure 1.2: Project Gantt Chart

Chapter 2

State of the Art and Technology Choices

2.1 IoT in Smart Agriculture

Smart Agriculture refers to the use of modern information and communication technologies, particularly the Internet of Things (IoT), to increase the quality and quantity of agricultural products. In a world with a growing population and increasing environmental pressures, optimizing resource usage (such as water and fertilizer) is paramount. IoT enables this by deploying networks of sensors to collect granular data on environmental and crop conditions in real-time. This data allows for more precise and automated decision-making, moving from traditional farming practices to a data-driven approach. Common applications include automated irrigation systems that water plants only when needed, drone-based crop monitoring, and automated climate control in greenhouses.

2.2 Key Enabling Technologies

2.2.1 Microcontrollers for IoT

The core of any IoT device is its microcontroller. While many options exist, such as the Arduino family (known for simplicity) or the Raspberry Pi (a full-fledged single-board computer), the ESP32 family stands out for embedded IoT applications. For this project, both the standard ESP32 and the more powerful ESP32-S3 were chosen. They offer an ideal balance of low cost, high performance (with dual-core processors), sufficient GPIO for connecting multiple peripherals, and, most importantly, integrated Wi-Fi and Bluetooth connectivity. This built-in wireless capability dramatically simplifies the design and reduces the cost and complexity of connecting our devices to a network.

2.2.2 Communication Protocols

Effective communication is critical for an IoT system. Two dominant protocols were considered and implemented for this project: MQTT and HTTP. MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe protocol designed for low-bandwidth, high-latency networks, making it perfect for sending frequent, small packets of sensor data. Its pub-sub model decouples the data producers (our sensors) from the data consumers (our backend). For sending large, single-instance data like images, the ubiquitous HTTP protocol (specifically, POST requests) is more suitable. It is robust and universally supported by web servers and platforms like Node-RED.

2.2.3 Data Visualization and Backend Platforms

A backend platform is needed to aggregate, process, and visualize the incoming data. While custom web applications could be built, a low-code platform like Node-RED was selected to accelerate development. Node-RED provides a browser-based, visual flow editor that allows for the easy connection of devices, APIs, and online services. Its extensive library of community-contributed nodes (for MQTT, databases, dashboards, etc.) and its intuitive workflow make it an ideal tool for rapid prototyping and building powerful IoT applications.

2.2.4 Cloud Database Solutions

To store historical data for analysis, a database is required. NoSQL (non-relational) databases are often preferred for IoT applications due to their flexible data schema, which can easily accommodate different types of data (like sensor readings and image logs). For this project, MongoDB Atlas was chosen as the cloud database solution. It is a powerful, scalable NoSQL database-as-a-service that offers a generous free tier, a straightforward setup process, and excellent compatibility with Node-RED.

2.3 Analysis of Existing Solutions and Project Justification

2.3.1 Existing Solutions

The field of agricultural monitoring includes a wide range of existing solutions, which can be broadly categorized:

- ***Commercial, Large-Scale Systems:*** *These are high-end, integrated solutions from major agricultural technology companies. They often involve GPS-guided tractors, drone surveillance with multispectral imaging, and expensive proprietary sensor networks.*
- ***Static, Wired Sensor Networks:*** *These systems typically consist of multiple sensor nodes placed at fixed locations in a field to monitor parameters like soil moisture or temperature. Data is often collected manually or transmitted through wired connections.*
- ***DIY/Hobbyist Projects:*** *Numerous open-source projects exist, but they are often focused on a single, specific task, such as an automated watering system for a single plant or a simple weather station.*

2.3.2 Limitations of Existing Solutions

While effective in their own right, these existing solutions present several limitations that this project aims to address:

- ***High Cost and Inaccessibility:*** *Commercial systems are prohibitively expensive for small-to-medium-sized farms, research institutions, or educational purposes.*
- ***Lack of Mobility and Granularity:*** *Static sensor networks only provide data for fixed points. They cannot offer a detailed, plant-by-plant visual overview of an entire area, which is crucial for detecting localized pest infestations or nutrient deficiencies.*

- ***Limited Scope and Integration:*** *Many DIY projects are not integrated systems. They may provide a sensor reading but often lack a centralized dashboard, a historical database, or a visual component. Crucially, they typically fail to combine continuous environmental data with on-demand visual plant health data.*

2.3.3 Added Value and Justification of Our Project

This project was designed to address the aforementioned limitations by creating an integrated, low-cost, and flexible monitoring system. The primary added value of our solution lies in:

- ***Cost-Effectiveness and Accessibility:*** *By using affordable, off-the-shelf components like the ESP32 and open-source software like Node-RED, our system provides advanced monitoring capabilities at a fraction of the cost of commercial alternatives.*
- ***Integrated Data Fusion:*** *The core innovation of this project is the fusion of two complementary data collection methods: a stationary unit for continuous, time-series environmental data, and an autonomous vehicle for mobile, high-resolution visual inspection. This provides a far more complete and actionable picture of crop health than either method could alone.*
- ***Unified, Real-Time Backend:*** *Unlike fragmented DIY projects, our solution centralizes all data—both sensor readings and images—into a single cloud database (MongoDB) and visualizes it on a unified, real-time dashboard (Node-RED). This provides a holistic and immediately accessible overview of the entire monitored area.*
- ***Modularity and Scalability:*** *The use of wireless technologies (Wi-Fi, MQTT) and a modular design allows for easy expansion. More sensor stations or different types of vehicles could be added to the same backend system with minimal changes.*

2.4 Summary of Technology Stack

The following table summarizes the key technologies chosen for this project and the justification for their selection.

Table 2.1: Summary of the project's technology stack.

Category	**Component/Technology**	**Justification**
Microcontroller	ESP32 & ESP32-S3	Low cost, powerful processing, integrated Wi-Fi.
Sensor Protocol	MQTT	Lightweight, reliable, and ideal for low-power sensor networks.
Image Protocol	HTTP	Robust and universally supported for transferring large files.
Backend Logic	Node-RED	Rapid development, visual workflow, extensive community support.
Dashboard	Node-RED Dashboard	Tightly integrated with the backend, easy to create real-time UIs.
Database	MongoDB Atlas	Scalable, flexible NoSQL database with a generous free tier.

Chapter 3

Design and Implementation of the Autonomous Vehicle

3.1 Hardware Components

The autonomous vehicle is the mobile data-gathering unit of the system, designed to navigate through rows of plants and capture high-resolution images. The component selection was focused on creating a robust, low-cost, and easily controllable platform. The key hardware parts are shown in Figure 3.7 and detailed below.

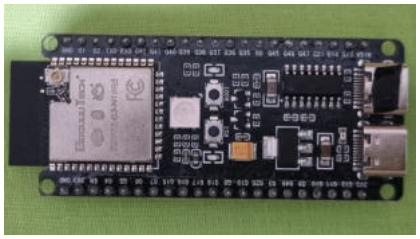

Figure 3.1: *
a) ESP32-S3 Controller

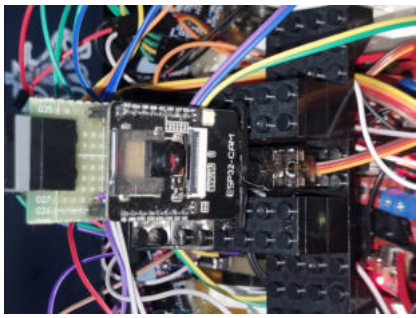

Figure 3.2: *
b) ESP-CAM Module

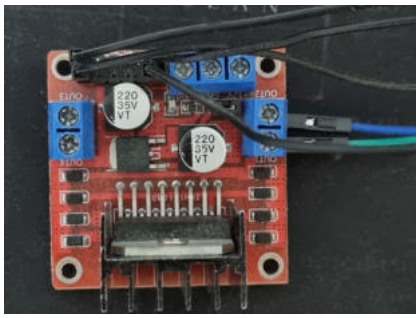

Figure 3.3: *
c) L298N Motor Driver

Figure 3.4: *
d) DC Motor

Figure 3.5: *
e) HC-SR04 Sensor

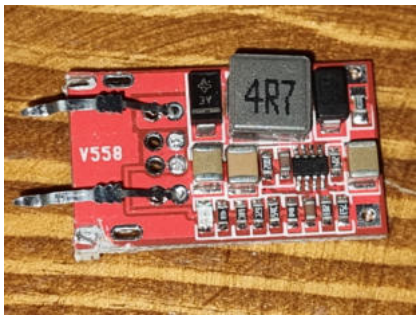

Figure 3.6: *
f) 5V DC-DC Converter

Figure 3.7: Key hardware components of the autonomous vehicle.

- ***Microcontroller (ESP32-S3 N16R8):*** *An ESP32-S3 (Figure 3.7a) was chosen as the main controller for the vehicle due to its powerful dual-core processor, ample GPIO*

pins, and native USB support. This power is necessary to manage six ultrasonic sensors, two motors, a servo, and UART communication simultaneously.

- ***Camera (AI-Thinker ESP-CAM):*** *The AI-Thinker ESP32-CAM (Figure 3.7b) was used as a dedicated, low-cost camera module. In this project, it acts as a peripheral controlled by the main ESP32-S3. It is responsible for capturing the image and handling its own Wi-Fi connection to transmit the data, offloading this task from the main controller.*
- ***Motors and Driver (2x DC Motors & L298N):*** *The vehicle uses a tank-like propulsion system with two DC motors (Figure 3.7d). This design was chosen for its robustness on uneven terrain. The motors are controlled by the L298N driver module (Figure 3.7c), which allows the ESP32-S3 to control the speed (via PWM) and direction of each motor independently.*
- ***Servo Motor (MG90S):*** *A small MG90S servo motor is used to pan the ESP-CAM left and right. This allows the vehicle to capture images from different angles without needing to turn the entire chassis, saving time and energy.*
- ***Obstacle Avoidance (6x HC-SR04):*** *To achieve autonomous movement and environmental awareness, the vehicle is equipped with six HC-SR04 ultrasonic sensors (Figure 3.7e). These sensors are placed around the chassis to provide 360-degree obstacle detection, allowing the software to make intelligent decisions to avoid collisions.*
- ***Power System (18650 Batteries & 5V Converter):*** *The vehicle is powered by two 18650 Li-ion batteries connected in series to provide approximately 7.4V. This voltage directly powers the L298N motor driver. A 5V DC-DC buck converter (Figure 3.7f) steps this voltage down to a stable 5V rail to safely power all the logic components, including the ESP32-S3, the servo, and the ultrasonic sensors.*

3.2 Circuit Design and Assembly

The vehicle's circuit is designed around two main power domains: a high-voltage domain for the motors and a regulated 5V domain for the logic components. Figure 3.12 shows multiple views of the fully assembled vehicle prototype, illustrating the placement of sensors and components.

Logic Level Shifting: *A critical design consideration was the voltage incompatibility between the 5V HC-SR04 sensors and the 3.3V ESP32-S3 microcontroller. While the ESP32-S3's 'Trig' pin can safely send a 3.3V signal to the 5V sensor, the sensor's 'Echo' pin sends back a 5V signal, which could damage the microcontroller. To solve this, a simple voltage divider circuit (Figure 3.13) using two resistors was implemented for each of the six 'Echo' pin connections. This circuit safely reduces the 5V signal to approximately 3.3V, protecting the ESP32-S3's GPIO pins.* ***Power Distribution:*** *The 7.4V output from the series-connected 18650 batteries is connected directly to the 'VMOT' input of the L298N driver to power the DC motors. The buck converter is connected to the same battery source to generate the stable 5V rail that powers the ESP32-S3 microcontroller, the servo motor, and all six HC-SR04 sensors.*

Logic Level Shifting: *A critical design consideration was the voltage incompatibility between the 5V HC-SR04 sensors and the 3.3V ESP32-S3 microcontroller. While the ESP32-S3's 'Trig' pin can safely send a 3.3V signal to the 5V sensor, the sensor's 'Echo' pin sends back a 5V signal, which could damage the microcontroller. To solve this, a simple voltage divider circuit (Figure 3.13) using two resistors was implemented for each of the six 'Echo' pin connections. This circuit safely reduces the 5V signal to approximately 3.3V, protecting the ESP32-S3's GPIO pins.*

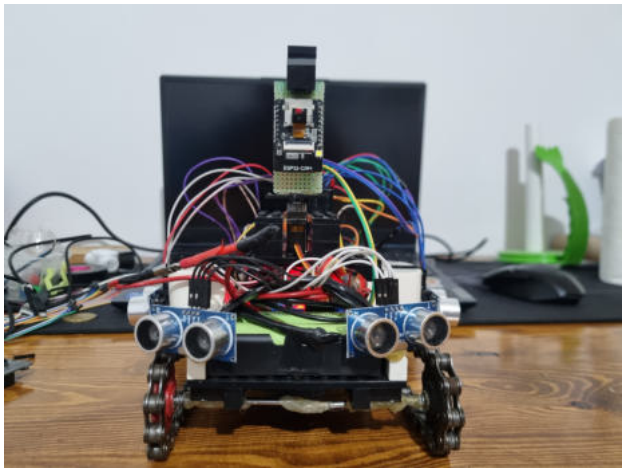

Figure 3.8: *
Front view

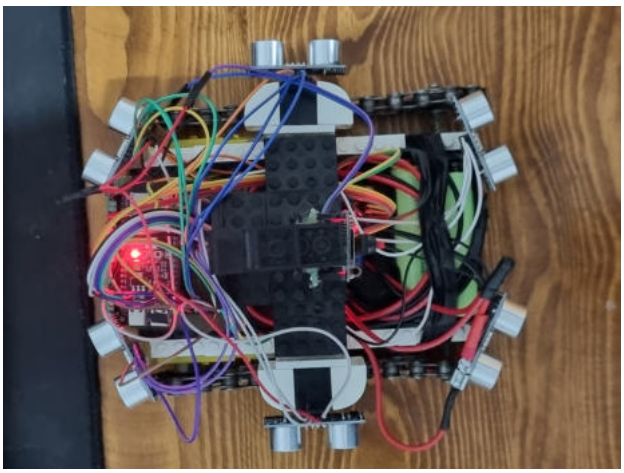

Figure 3.9: *
Top-down view

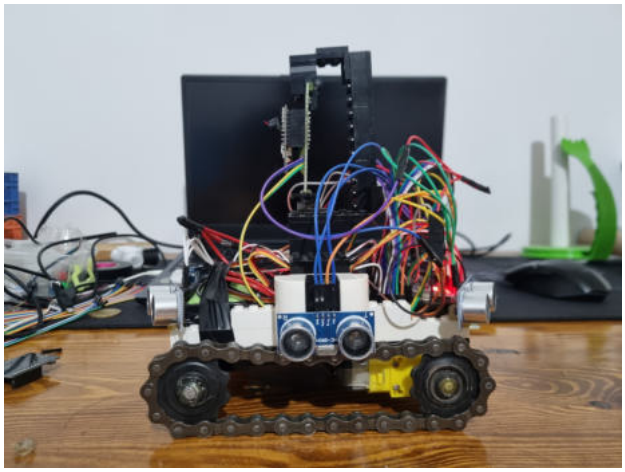

Figure 3.10: *
Left side view

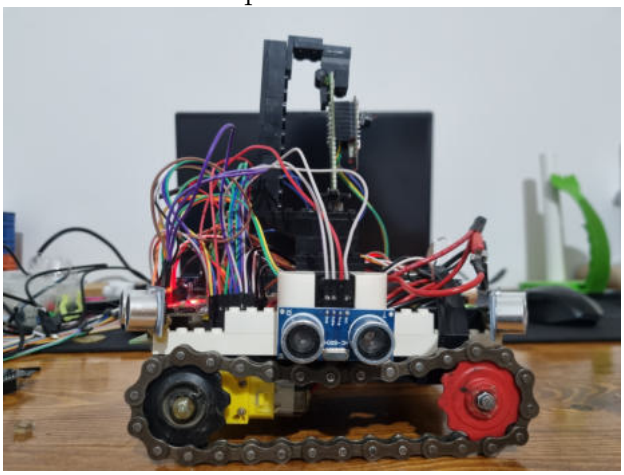

Figure 3.11: *
Right side view

Figure 3.12: Different views of the assembled autonomous vehicle.

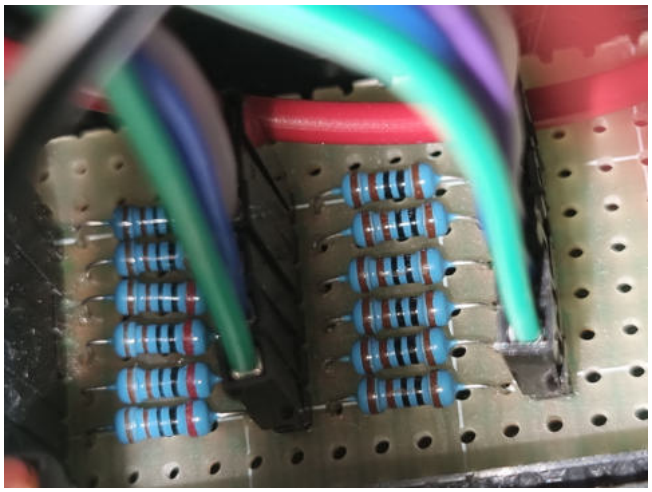

Figure 3.13: The voltage divider circuit used for each HC-SR04 Echo pin to ensure voltage compatibility with the 3.3V ESP32-S3.

3.3 Software Implementation

The vehicle's software was developed in the Arduino IDE. The main loop is responsible for navigation, while communication with the ESP-CAM is handled via UART to trigger the image capture and send process.

- ***Autonomous Navigation:*** *The core of the navigation logic involves periodically reading the distance from all six HC-SR04 sensors. The software analyzes these distances to build a simple map of its immediate surroundings. If an obstacle is detected by the front sensors, the vehicle stops and checks the side sensors to decide whether to turn left or right. This creates a basic but effective collision avoidance behavior.*

- ***Camera Control:*** *The ESP32-S3 acts as the master controller. When it is time to take a picture, it first sends a command to the servo motor to pan to a specific angle (e.g., center, left, or right). After a short delay to allow the servo to stabilize, it sends a command character (e.g., 'C') over a UART serial connection to the ESP-CAM.*

- ***Data Transmission Logic:*** *The ESP-CAM runs its own separate firmware which constantly listens for commands on its UART port. Upon receiving the 'C' command, it initiates its own internal process: it captures a high-resolution image, connects to the Wi-Fi network (the mobile hotspot), and sends the image data via an HTTP POST request directly to the Node-RED server's IP address. This distributed approach allows the main vehicle controller to continue its navigation tasks while the camera module handles the time-consuming process of Wi-Fi connection and data upload.*

Chapter 4

Design and Implementation of the Plant Monitoring Station

4.1 Hardware Components

The stationary monitoring station is the core of the environmental data collection system. It is designed to operate continuously, gathering critical data about the plant's immediate surroundings and soil conditions. The selection of components was driven by the need for accuracy, low power consumption, and ease of integration with the ESP32 microcontroller. The key hardware parts are shown in Figure 4.7 and detailed below.

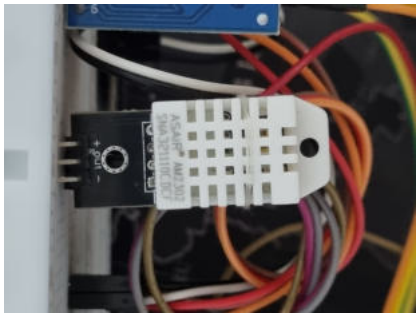

Figure 4.1: *
a) DHT22 Sensor

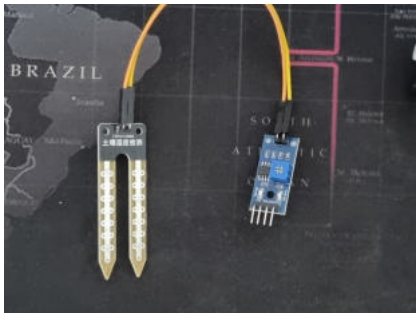

Figure 4.2: *
b) Soil Moisture Sensor

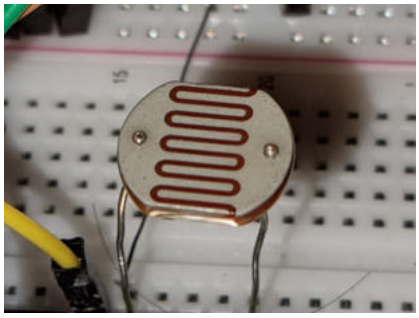

Figure 4.3: *
c) LDR Light Sensor

Figure 4.4: *
d) PIR Motion Sensor

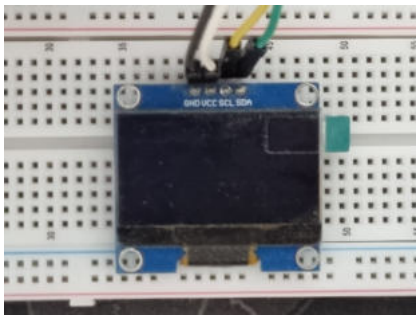

Figure 4.5: *
e) OLED Display

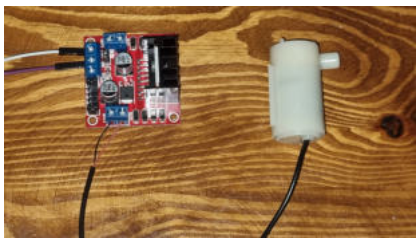

Figure 4.6: *
f) DC Water Pump

Figure 4.7: Key hardware components of the Plant Monitoring Station.

- ***Microcontroller (ESP32):*** *The standard ESP32 development board was chosen as the brain of the station. Its built-in Wi-Fi capabilities are essential for transmitting data*

wirelessly via MQTT, and its ample GPIO pins allow for easy interfacing with all the selected sensors and peripherals.

- ***Temperature and Humidity Sensor (DHT22):*** *The DHT22 (Figure 4.7a) is a digital sensor used to measure ambient temperature and relative humidity. It was selected for its good accuracy and simple one-wire digital interface.*
- ***Soil Moisture Sensor:*** *A capacitive soil moisture sensor (Figure 4.7b) was used to measure the water content in the soil. Unlike resistive sensors, the capacitive type does not corrode over time, making it more reliable for long-term deployment.*
- ***Light Dependent Resistor (LDR):*** *An LDR (Figure 4.7c) was used to gauge the ambient light intensity. This simple sensor is crucial for the automated control of the supplementary grow lights.*
- ***PIR Motion Sensor:*** *A PIR motion sensor (Figure 4.7d) was included for a practical, user-friendly feature. It detects the presence of a person, allowing a farmer with dirty or wet hands to activate the screen for a status check without needing to touch any buttons.*
- ***OLED Display (SSD1106):*** *A 1.3-inch OLED display (Figure 4.7e) provides a local, real-time readout of the sensor data. It uses the I2C protocol, simplifying wiring to just two data pins.*
- ***Actuators (Water Pump & LED Lights):*** *A small DC water pump (Figure 4.7f) and a strip of LED grow lights were integrated as actuators. They are controlled safely through 5V relay modules, which allow the low-power ESP32 to switch the high-power actuators on and off.*

4.2 Circuit Design and Assembly

The assembly of the monitoring station involved connecting all sensors and actuators to the ESP32 microcontroller and ensuring a stable power supply. The circuit was designed to be modular and easy to troubleshoot. Figure 4.8 shows the fully assembled prototype. Key aspects of the circuit include the LDR voltage divider and the use of relays for actuator control.

Relay Modules for Actuators: *The DC water pump and LED lights require more current than the ESP32's GPIO pins can safely provide. To address this, 5V relay modules were used. The ESP32 sends a simple digital signal to the relay's control pin. This low-power signal activates an electromagnet inside the relay, which closes a high-power switch, completing the circuit for the pump or lights. This electrically isolates the sensitive microcontroller from the noisy, high-power actuator circuits.* ***LDR Voltage Divider Circuit:*** *The LDR is a variable resistor, but the ESP32's ADC reads voltage, not resistance. To solve this, the LDR was connected in a voltage divider circuit with a fixed 2kΩ resistor. As the light intensity changes, the LDR's resistance changes, which in turn changes the voltage at the point between the two resistors. This varying voltage is then read by an analog pin on the ESP32.*

Relay Modules for Actuators: *The DC water pump and LED lights require more current than the ESP32's GPIO pins can safely provide. To address this, 5V relay modules were used. The ESP32 sends a simple digital signal to the relay's control pin. This low-power signal activates an electromagnet inside the relay, which closes a high-power switch, completing the circuit for the pump or lights. This electrically isolates the sensitive microcontroller from the noisy, high-power actuator circuits.*

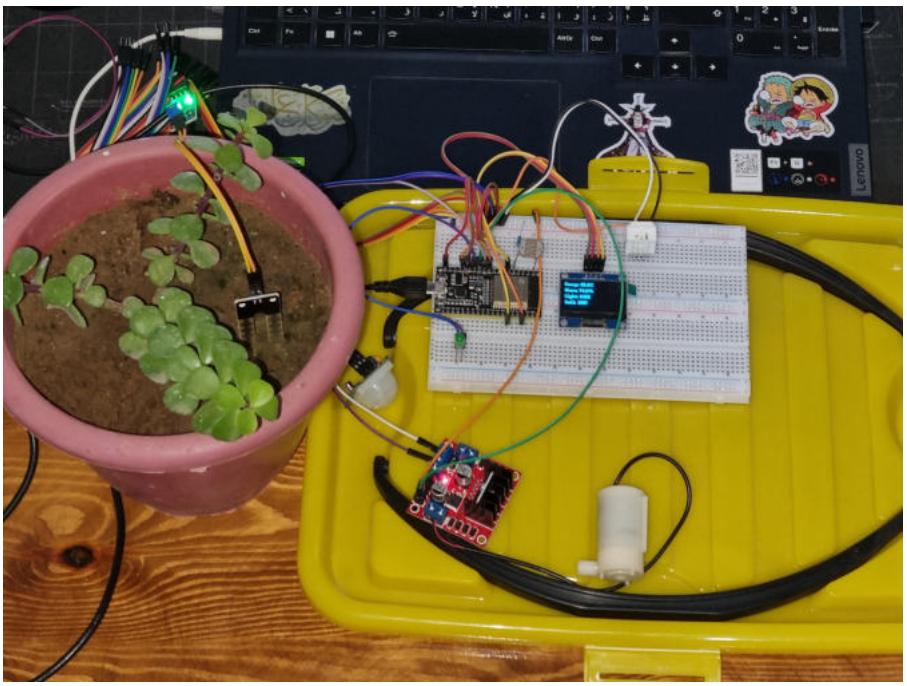

Figure 4.8: The fully assembled Plant Monitoring Station prototype.

4.3 Software Implementation

The software for the monitoring station was developed in the Arduino IDE using C++. The program is built around a non-blocking main loop that performs several key tasks periodically. The full source code is available in Appendix A.

- ***Sensor Reading Loop:*** *A timer based on the 'millis()' function is used to read all connected sensors every five seconds. This avoids the use of blocking 'delay()' functions, ensuring the code remains responsive. The raw data from the DHT22, LDR, and soil moisture sensor is read and stored in global variables.*

- ***MQTT Publishing:*** *After each sensor reading cycle, the data is published to a public MQTT broker (HiveMQ). Each sensor value is sent to its own unique topic (e.g., 'plant_monitor/temperature', 'plant_monitor/humidity'). This publish-subscribe architecture decouples the sensor node from the backend; the ESP32 does not need to know the IP address of the server, only the address of the broker.*

- ***Automated Actuator Control:*** *The software implements a simple control logic based on predefined thresholds. If the light level from the LDR falls below 'LIGHT_THRESHOLD_LOW', the ESP32 activates the relay for the grow lights. Similarly, if the soil moisture reading goes above 'SOIL_MOISTURE_THRESHOLD_LOW' (indicating dry soil, as the sensor's reading is inverted), the water pump relay is activated. The system also subscribes to MQTT topics to allow for manual override of the actuators from the Node-RED dashboard.*

- ***OLED Display and Power Saving:*** *The program constantly checks the state of the PIR motion sensor. If motion is detected, a timestamp is recorded and the OLED display is turned on, showing the most recent sensor values. This provides an immediate, hands-free data readout, which is highly practical for a user like a farmer who may not be able to interact with a screen directly. If no motion is detected for a period of 10 seconds ('screenOffDelay'), the display is automatically turned off to conserve energy, extending the operational life of the unit.*

Chapter 5

Backend System: Data Aggregation and Visualization

The backend system serves as the central hub for the entire project. It is responsible for receiving data from both the autonomous vehicle and the stationary monitoring station, processing this information, visualizing it in real-time for the user, and logging it for historical analysis. The entire backend was implemented using Node-RED, chosen for its rapid development capabilities and modular, flow-based programming model.

5.1 Node-RED Flow Design

The logic of the backend is defined by a single, comprehensive Node-RED flow, as shown in Figure 5.1. The flow is divided into three main functional branches: sensor data ingestion, image data ingestion, and database storage.

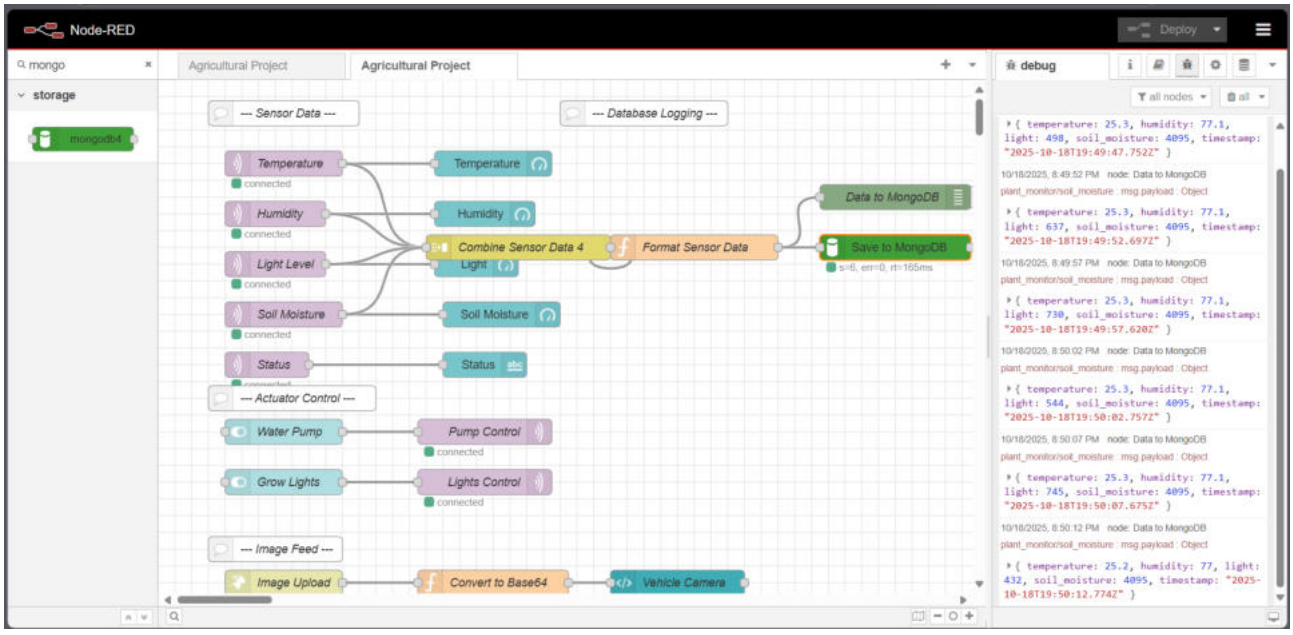

Figure 5.1: The complete Node-RED flow for data processing and storage.

Image Data Ingestion: *An 'HTTP in' node is configured to create a web endpoint at the URL '/upload/image'. This node listens for incoming HTTP POST requests. When the autonomous vehicle sends a picture, this node receives the raw binary data of the JPEG image. The*

data is then passed along two paths: one for immediate display on the dashboard and another for processing and storage. ***Sensor Data Ingestion:*** *An 'MQTT in' node subscribes to the parent topic 'plant$_m$onitor/', capturingalldatapublishedbythestationarymonitoringunit.Themessagesarether*

Image Data Ingestion: *An 'HTTP in' node is configured to create a web endpoint at the URL '/upload/image'. This node listens for incoming HTTP POST requests. When the autonomous vehicle sends a picture, this node receives the raw binary data of the JPEG image. The data is then passed along two paths: one for immediate display on the dashboard and another for processing and storage.*

5.2 Dashboard Implementation

A real-time user interface was created using the 'node-red-dashboard' nodes to provide an intuitive and immediate overview of the system's status. The dashboard, shown in Figures 5.2 and 5.3, is designed to be accessible from any web browser on the same network.

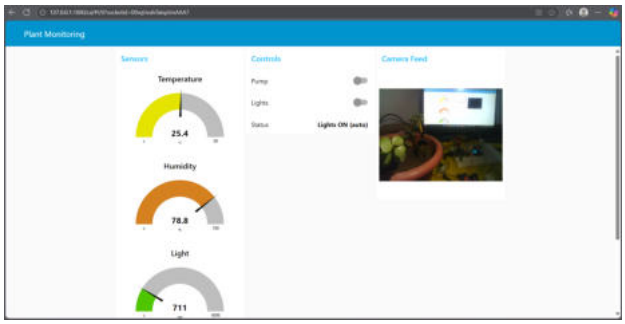

Figure 5.2: Dashboard view showing sensor data gauges.

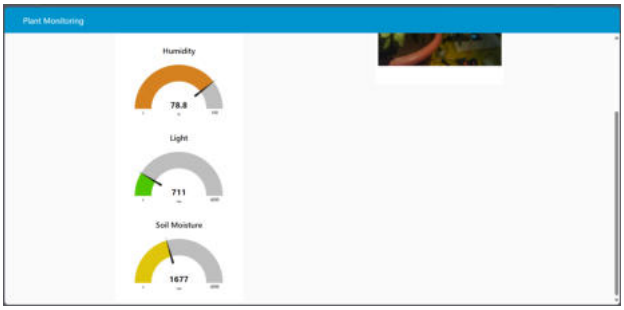

Figure 5.3: Dashboard view showing the live image feed.

The UI consists of several key elements:

- ***Gauges:*** *Four 'ui_gauge' nodes are used to display the real-time values for temperature, humidity, light intensity, and soil moisture, providing a quick visual assessment of the environmental conditions.*
- ***Image Feed:*** *A 'ui_template' node is used to display the most recent image received from the vehicle. It uses simple HTML and Angular.js to render the base64-encoded image data sent from the backend flow.*
- ***Actuator Switches:*** *Two 'ui_switch' nodes allow the user to manually override the automated control of the water pump and grow lights, providing direct control over the system's actuators.*

5.3 Cloud Database Integration with MongoDB

To ensure data persistence and enable historical analysis, all incoming data is logged to a cloud-hosted NoSQL database using MongoDB Atlas. This service was chosen for its generous free tier, straightforward setup, and robust performance. Figure 5.4 shows an example of the sensor data as it is stored in the database collection.

The data storage strategy is as follows:

- ***Sensor Data:*** *The formatted sensor data object from the monitoring station is sent to a 'mongodb4 out' node configured with the 'insertOne' operation. Each object is saved as a*

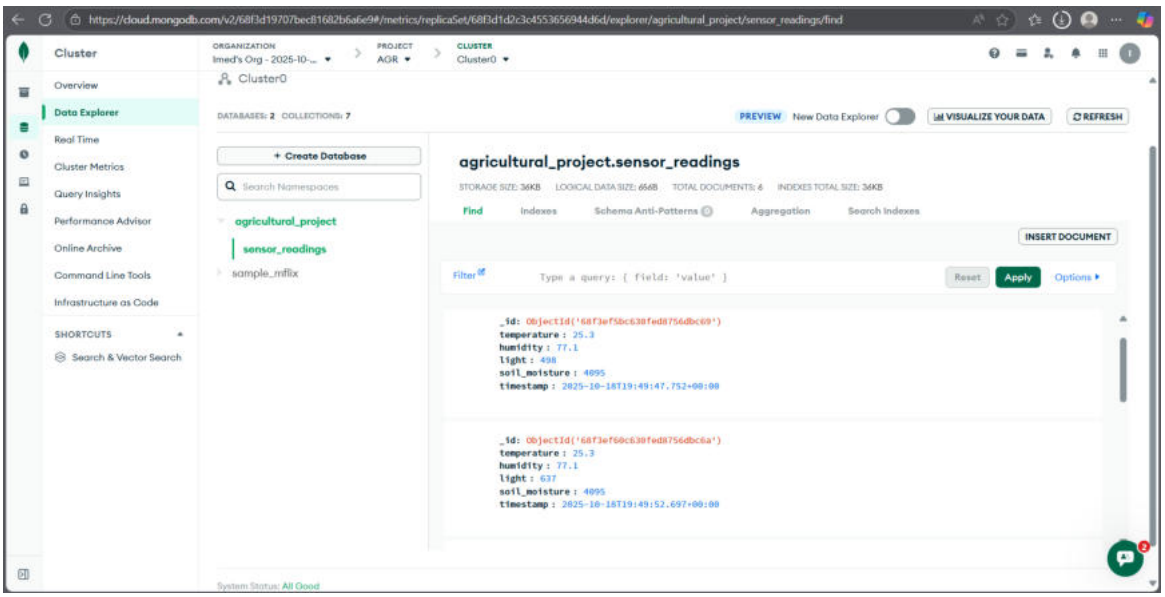

Figure 5.4: Sensor data being stored in the ‘sensor_readings‘ collection in MongoDB Atlas.

new document in a collection named ‘sensor_readings‘. Each document contains the four sensor values and a precise timestamp.

- ***Image Data (Future Work):*** *While the current flow successfully displays the live image, the next logical step, not yet implemented, would be to store the images. The professional approach, as designed in the flow, would be to upload the image file to a dedicated file service (like Firebase Storage) and then save a log containing the image URL and a timestamp to a separate ‘image*$_{l}$*ogs‘collectioninMongoDB.Thisseparateslargefilestoragefromstructure*

Chapter 6

System Integration, Testing, and Validation

After the individual development of the autonomous vehicle, the monitoring station, and the backend system, the final phase of the project involved integrating these components and conducting a series of tests to validate the functionality and performance of the entire system. This chapter outlines the testing protocols used, discusses the results and challenges encountered, and presents the limitations of the current prototype.

6.1 Test Protocols

A multi-layered testing strategy was adopted to ensure the reliability of the system, starting from individual components and progressing to the fully integrated solution.

- ***Unit Tests:*** *Each hardware component was tested in isolation to verify its basic functionality. For example, a simple sketch was used to confirm that the DHT22 sensor was providing temperature readings, that the L298N driver could spin the DC motors, and that the ESP-CAM could successfully initialize and capture an image using the standard 'CameraWebServer' example.*

- ***Integration Tests:*** *Components were then tested as integrated subsystems. For the monitoring station, this involved running the full script to ensure that sensor readings were not only taken but also correctly published to the MQTT broker. For the vehicle, tests were conducted to confirm that the ESP32-S3 could correctly read from all six ultrasonic sensors and control the motors in response to detected obstacles.*

- ***End-to-End System Test:*** *The final test involved running the entire system in a real-world scenario. The monitoring station was placed near a plant, and the autonomous vehicle was activated in the same area. The objectives were to confirm:*

    1. *That sensor data from the stationary node was appearing on the Node-RED dashboard and being logged to the MongoDB database in real-time.*
    2. *That the vehicle could successfully capture an image, transmit it over the Wi-Fi hotspot, and that the image would appear on the dashboard.*
    3. *That the system could run continuously without crashes or significant data loss.*

6.2 Results and Discussion

The end-to-end system test was successful, and the prototype met all the primary objectives defined at the start of the project. Sensor data was reliably transmitted and stored, and the vehicle successfully delivered images to the dashboard. However, the integration process was not without challenges. Several key issues were identified and resolved, demonstrating critical problem-solving during the project.

Challenge 1: Camera Initialization and Power Stability

An initial and persistent problem was the failure of the ESP-CAM module to initialize or capture images, often resulting in 'Camera Init Failed' or 'Camera Capture Failed' errors. Through systematic testing, it was determined that this was primarily a power-related issue. The camera's image sensor requires a significant inrush of current upon startup, which the standard USB port on a computer could not reliably provide. The issue was resolved by powering the ESP-CAM with a more robust, stable 5V power supply, which provided the necessary current to overcome the initial power sag.

Challenge 2: Network Connectivity with Mobile Hotspot

During development, a major networking issue was discovered when using a mobile phone hotspot. After resetting the ESP-CAM, it would fail to reconnect to the hotspot, getting stuck in an infinite connection loop. This was diagnosed as an issue with the hotspot's DHCP server, which would not correctly release the previously assigned IP address. The most robust solution, which was implemented, was to configure the ESP-CAM with a static IP address. This bypassed the faulty DHCP negotiation entirely and resulted in a fast and reliable connection every time the device was reset.

Challenge 3: Backend Connection Refusal

Even when the ESP-CAM was successfully connected to the network and taking pictures, it initially failed to send the image to the Node-RED server, receiving a 'Connection Refused' error. This indicated that the request was reaching the host computer but was being actively blocked. The root cause was identified as the Windows Defender Firewall, which, by default, blocks incoming connections on new ports. The problem was solved by creating a specific inbound rule in the firewall settings to explicitly allow traffic on port '1880', the port used by Node-RED.

6.3 Limitations and Future Improvements

While the project successfully resulted in a functional prototype, it has several limitations inherent to its design as a proof-of-concept. These also present clear opportunities for future improvements.

- ***Limitation - Network Dependency:*** *The system is entirely dependent on Wi-Fi coverage. In a large agricultural field, this would be a significant constraint. A future improvement would be to replace Wi-Fi with a long-range communication protocol like LoRaWAN.*
- ***Limitation - Battery Life:*** *The autonomous vehicle's operational time is limited by the capacity of the two 18650 batteries. For extended deployment, a more sophisticated*

power management system, including the potential for solar panel charging or a docking station, would be necessary.

- ***Limitation - Navigation System:*** *The vehicle's navigation is purely reactive, based on obstacle avoidance. It has no knowledge of its absolute position. A major improvement would be to integrate a GPS module and a compass (IMU) to enable true autonomous navigation to specific waypoints.*
- ***Future Improvement - Data Analysis:*** *The current system successfully collects and stores data. The next logical step would be to apply data analysis and machine learning techniques. For example, the sensor data could be used to predict disease risk, and the images could be processed using computer vision algorithms to automatically detect signs of plant stress or pest damage.*

General Conclusion

Summary of Achievements

Briefly summarize what you accomplished, referencing your initial objectives. "The project successfully resulted in a functional prototype of an IoT agricultural monitoring system..."

Acquired Skills

List the technical and soft skills you learned or improved during the internship (e.g., embedded systems programming, network protocols, cloud database management, problem-solving, project management).

Future Perspectives

Discuss the potential real-world application of this project and how it could be expanded into a more robust, commercial product.

General Conclusion

This internship project set out to address the need for modern, data-driven monitoring in agriculture by designing and building an integrated IoT system from the ground up. By combining a mobile imaging unit with a stationary environmental sensor node, the project successfully created a holistic data collection platform. The development process spanned hardware selection, circuit design, embedded software programming, and backend development, culminating in a functional prototype that met all primary objectives.

Summary of Achievements

Over the course of the internship, the initial concept was transformed into a tangible and working system. The key achievements are directly aligned with the project's objectives:

- *A fully autonomous ground vehicle was constructed, capable of navigating its environment and wirelessly transmitting images.*
- *A stationary monitoring station was successfully built, providing reliable, real-time data on key environmental variables via MQTT.*
- *A robust backend system using Node-RED was established, providing a user-friendly dashboard for immediate data visualization.*
- *A cloud-based data persistence solution was implemented using MongoDB, ensuring that all collected data is logged for historical analysis.*
- *Critical real-world challenges related to power stability, network protocols, and system security (firewalls) were systematically identified and solved, demonstrating a practical approach to engineering.*

The final integrated system stands as a successful proof-of-concept for a low-cost, scalable agricultural monitoring solution.

Acquired Skills

This project served as a significant learning experience, allowing me to develop and refine a wide range of technical and soft skills. On the technical side, I gained hands-on experience in:

- *Embedded systems programming with the ESP32 family in a C++ environment.*
- *Hardware integration, circuit design, and practical electronics, including power management and logic level shifting. - Network communication protocols, specifically the practical implementation of MQTT and HTTP for IoT applications.*
- *Backend development and data flow management using Node-RED.*

- *Cloud database integration and management with a NoSQL database (MongoDB Atlas).*

Beyond the technical skills, this internship also strengthened my abilities in project management, systematic problem-solving, and the critical process of documenting and reporting on a complex engineering project.

Future Perspectives

The completed prototype serves as a strong foundation for a much more advanced and commercially viable system. The modularity of the design opens up numerous avenues for future development. The vehicle could be enhanced with GPS for waypoint navigation, the backend could incorporate machine learning algorithms to analyze the incoming images for automated disease detection, and the entire system could be adapted to use long-range communication protocols like LoRaWAN for deployment in large, remote fields. This project not only met its goals but also paved the way for future innovation in the field of smart agriculture.

Bibliography

List any datasheets, articles, tutorials, or books you referenced.

[1] *Espressif Systems, "ESP32 Series Datasheet", Version 3.3, 2021. [Online]. Available:* https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[2] *Node-RED, "Node-RED Documentation". [Online]. Available:* https://nodered.org/docs/

[3] *MongoDB, Inc., "MongoDB Atlas Documentation". [Online]. Available:* https://docs.atlas.mongodb.com/