ROTTEN FRUIT DETECTION USING IOT

NUR SAFI SYUHRAH BINTI ABDUL RANI

52224122591

UNIVERSITI KUALA LUMPUR

MARCH 2025

ROTTEN FRUIT DETECTION USING IOT

NUR SAFI SYUHRAH BINTI ABDUL RANI

52224122591

Report Submitted to Fulfilment of the Requirements for the Bachelor in Information Technology (Honours) (Internet of Things)

UNIVERSITI KUALA LUMPUR

MARCH 2025

DECLARATION

I declare that this report is my original work, and all references have been cited adequately as required by the university.

Date: 23th June 2025

Signature: 

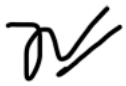
Full Name: Nur Safi Syuhrah Binti Abdul Rani

ID Number: 52224122591

APPROVAL PAGE

This report has been supervised, examined and verified that it meets the program and university's requirements for the Bachelor in Information Technology (Honours) (Internet of Things).

Date: 23th June 2025

Signature: 

Supervisor: Assoc. Prof. Ts. Dr. Haidawati binti Mohamad Nasir

Official Stamp:

Date: _____

Signature: ……

Co-Supervisor (if any):

Official Stamp:

COPYRIGHT PAGE

ACKNOWLEDGEMENT

In the name of Allah, the Most Beneficent, the Most Merciful. It is with deep gratitude and appreciation that I express my sincere thanks to my supervisor, Assoc. Prof. Ts. Dr. Haidawati binti Mohamad Nasir, for her unwavering guidance, support, and encouragement throughout the course of my Final Year Project. Despite her demanding schedule, she consistently provided valuable insights and constructive feedback, which played a crucial role in the successful completion of this project. I would also like to extend my appreciation to Ts. Faridah binti Yahya and Sir Azmi bin Ahmad for their efforts in coordinating the Final Year Project and providing the necessary guidelines and assistance throughout the process. My heartfelt thanks also go to my beloved parents and family. Their endless support, patience, and encouragement have been the backbone of my journey. Whenever I felt overwhelmed or discouraged, their presence and motivation gave me the strength to carry on. This project would not have been possible without the help and support of all these individuals, and I am deeply grateful to each of them.

ABSTRACT

The primary goal of this project is to develop an IoT-based system that detects spoiled fruits in real time to reduce food waste and maintain freshness. The system uses an MQ4 gas sensor to detect spoilage-related gas emissions and a DHT22 sensor to monitor temperature and humidity. A Wemos D1 ESP8266 microcontroller collects and sends the data to Node-RED, where it is analyzed using a prediction model. If the readings indicate spoilage, a Telegram alert is sent to notify the user. This system provides a cost-effective and efficient solution for households to monitor fruit condition and take timely action.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation		Descriptions
CSV	-	Comma-Separated Values
CNN	-	Convolutional Neural Network
IDE	-	Integrated Development Environment
IoT	-	Internet of Things
LCD	-	Liquid Crystal Display
ML	-	Machine Learning
MQTT	-	Message Queuing Telemetry Transport
UV	-	Ultraviolet
YOLO	-	You Only Look Once

CHAPTER 1: INTRODUCTION

1.1 Introduction

Every year, millions of tons of food are wasted globally, with fruits being among the most discarded items due to their perishable nature and improper storage. Forgotten fruits left in refrigerators often spoil before they can be consumed, contributing to significant food waste and environmental harm (Salonen, 2022). The aim of this project is to develop an IoT-based system that detects early signs of fruit spoilage and notifies users before the fruits completely rot. The objectives of this project include integrating a gas sensor to monitor spoilage-related emissions and an environmental sensor to track temperature and humidity (Gawas et al., 2021). The system is designed to provide real-time alerts, enabling users to take timely action to prevent fruit spoilage. The expected outcome is a reliable and user-friendly solution that helps reduce food waste, lower household expenses, and promote sustainable living practices.

1.2 Project Theme

For this project, the design structure focuses on integrating IoT technology to automate detect in fruit freshness monitoring in household settings. The hardware will communicate with a software dashboard to track important parameters like gas emissions, temperature, and humidity levels, along with visual signs of spoilage. The initial proposed hardware and sensors include:

- MQ4 Gas Sensor
  This sensor specifically monitors methane levels, which tend to rise significantly as fruits enter advanced stages of decay. This helps the system detect higher levels of spoilage, adding an additional layer of monitoring.

- DHT22 Sensor
  This sensor records temperature and humidity levels, which are crucial environmental factors in fruit spoilage. By maintaining optimal conditions, the system helps slow down the decay process, providing insights into storage conditions.

All proposed hardware components will be managed using an ESP8266 microcontroller with built-in wireless communication. The sensor data will be processed and visualized on a mobile or web dashboard, allowing users to monitor their produce remotely and receive timely alerts.

1.3 Problem Statement

Many households face the common problem of forgetting fruits stored in the fridge, often leading to spoilage and unnecessary food waste (Schanes et al., 2018). Without a reliable system to monitor their freshness, users are unaware of early signs of spoilage, resulting in fruits being discarded once they are completely rotten. This project addresses this issue by providing an IoT-based system that detects early indicators of fruit spoilage, such as gas emissions and environmental changes, and promptly notifies users. By enabling timely action, this system helps reduce food waste, ensures better produce management, and promotes sustainability in daily household practices.

1.4 Objectives

This project's objectives are:

- To use MQ4 gas sensors to get gas reading for rotten fruits by setting baseline readings in a fresh, non-spoiling environment.
- To develop an IoT-based system using sensors to continuously monitor the condition of fruits and detect early signs of spoilage, such as changes in gas emissions or other environmental factors.
- To create an alert system that notifies users when fruits are at risk of rotting, based on the data gathered from IoT sensors.

1.5 Project Scopes

The scope of this project is to develop an IoT-based fruit spoilage detection system specifically designed for household users, including those who may not have the time or ability to regularly check their stored produce. The system will monitor environmental factors such as gas emissions, temperature, and humidity using sensors and provide real-time alerts through a user-friendly interface. By automating the detection process, the system ensures that users can easily manage their fruit storage without needing technical knowledge or constant attention, making it accessible for all, including busy individuals or those with physical or cognitive limitations. This solution aims to reduce food waste, improve convenience, and promote sustainable practices in everyday household routines.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Food waste is a critical global issue, with millions of tons of food, particularly fruits, discarded annually due to spoilage and improper storage practices. This waste not only represents a significant economic loss but also exacerbates environmental problems, contributing to greenhouse gas emissions and resource depletion. Traditional methods of monitoring fruit freshness are often inadequate, relying on subjective visual inspections or expiration dates that may not accurately reflect the food's actual condition. To combat this issue, the proposed project aims to develop an IoT-based system that detects early signs of fruit spoilage using advanced sensor technology and provides timely notifications to users. By integrating gas sensors, environmental monitoring tools, and visual inspection capabilities, this system seeks to minimize food waste and promote sustainable consumption practices.

2.2 Related Works

2.2.1 IOT Based Food Freshness Detection Using Deep Learning Technique

This study introduces an IoT-powered food freshness detection system that combines sensor data with deep learning techniques. The system incorporates CNN for image classification, YOLO for detecting defects on food surfaces, and MQ2 and MQ135 sensors to monitor gases emitted during spoilage. These technologies collectively enhance the accuracy and efficiency of spoilage detection. The integration of visual data and sensor inputs represents a significant step forward in freshness monitoring for households and industries.

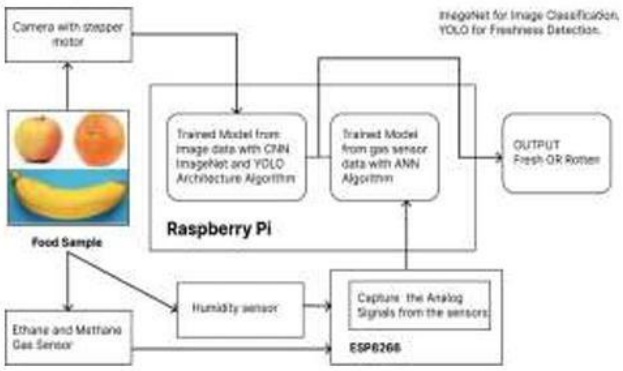


Figure 2.1 Architecture Diagram of The Project

The methodology involves capturing 360-degree images of food using a rotating camera, analysing gas emissions using sensors, and processing data through Raspberry Pi and ESP8266 microcontrollers. Sensor and image data are combined to generate comprehensive insights into the freshness of food items. Alerts and real-time results are displayed on a browser-based application for user convenience.

The system offers several advantages, including real-time monitoring, cost-effectiveness, and high detection accuracy. By integrating multiple parameters, it overcomes the limitations of traditional single-parameter freshness detection methods. However, challenges include environmental sensitivity and delays when processing large datasets, which could hinder industrial applications.

In conclusion, this research demonstrates the potential of combining IoT and deep learning to enhance food quality management. The system's ability to reduce waste and improve monitoring processes positions it as a valuable tool for households and food industries alike.

2.2.2 IoT based Fruit Quality Inspection and Lifespan Detection System

This paper proposes an IoT-based system to monitor fruit quality and predict lifespan. The system uses DHT11 sensors for temperature and humidity monitoring and MQ4 sensors for methane gas detection, managed by Raspberry Pi Pico microcontrollers. Real-time data is analyzed with machine learning algorithms to estimate fruit spoilage stages and predict expiration dates, making it suitable for warehouses and cold storage applications.

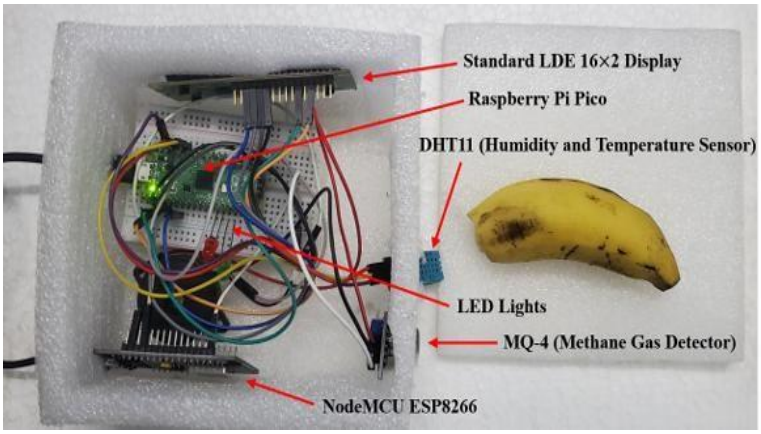


Figure 2.2 Prototype of The Project

The methodology integrates IoT sensors with a web-based interface for visualization and alerts. Data is transmitted via NodeMCU ESP8266, stored in a cloud database, and analyzed using custom algorithms that calculate fruit lifespan based on spoilage indicators. The system provides insights into environmental conditions and enables users to take preventive measures.

Advantages include cost-effectiveness, real-time monitoring, and the system's scalability for small-scale storage or household use. However, challenges such as environmental dependency and limited scalability for industrial operations are noted. These limitations indicate the need for further development to enhance its robustness.

Overall, this study highlights the potential of IoT in fruit quality monitoring and spoilage prevention. By combining data-driven analysis with real-time notifications, the system addresses food waste concerns and ensures better produce management.

2.2.3 IoT-Based Food Spoilage Detection System with UV Sterilization

This paper combines IoT technology with UV sterilization to detect and mitigate food spoilage. Using MQ135 and DHT11 sensors, the system tracks gas emissions, temperature, and humidity levels. A NodeMCU microcontroller processes the data, and UV sterilization is activated upon detecting spoilage. This integration aims to prolong food shelf life while ensuring safety and reducing waste.

The methodology involves collecting sensor data and transmitting it to a mobile application for real-time monitoring. When spoilage indicators exceed predefined thresholds, the system triggers UV sterilization to eliminate microorganisms. The application also provides users with alerts and a dashboard for visualization.

The system offers advantages such as enhanced food safety, reduced waste, and real-time spoilage detection. However, limitations include the energy demands of UV sterilization and sensor sensitivity to environmental noise. These challenges highlight areas for improvement in future iterations of the system.

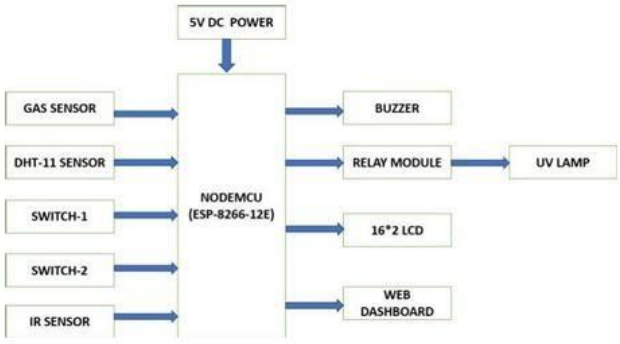


Figure 2.3 Block Diagram

In conclusion, this research presents a practical solution for food safety and preservation, emphasizing the role of IoT and sterilization technology in minimizing waste and improving storage conditions.

2.2.4 Fruit Freshness Detection and Monitoring using IOT

This paper proposes an IoT-based system to detect fruit freshness in real time. It uses MQ2 and MQ4 gas sensors to monitor spoilage-related emissions and an Arduino microcontroller for data processing. The system generates alerts when spoilage thresholds are exceeded, allowing users to take timely actions to prevent further decay.

The methodology integrates sensors to collect data on gas emissions, processed by an Arduino and visualized on a user-friendly dashboard. The system's non-invasive approach preserves the quality of the fruits while providing accurate monitoring. Data thresholds are calibrated to detect early spoilage signs effectively.

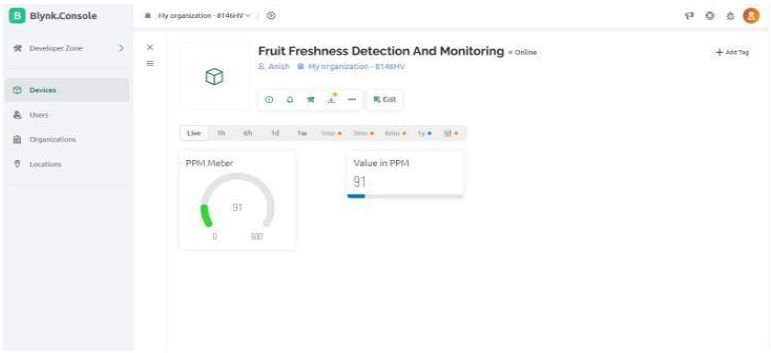


Figure 2.4 Blynk Interface

Advantages of the system include affordability, real-time usability, and the ability to minimize household food waste. However, limitations such as sensor sensitivity and dependence on stable environmental conditions restrict its wider application. Regular calibration is also necessary for accurate performance.

Overall, this study demonstrates the potential of IoT to enhance household produce management. By providing real-time insights, the system promotes sustainable practices and reduces food waste in daily routines.

2.2.5 Developing an IoT and ML-driven Platform for Fruit Ripeness Evaluation and Spoilage Detection: A Case Study on Bananas

This paper presents a novel IoT, and machine learning platform designed to monitor fruit ripeness and detect spoilage. By leveraging IoT sensors like SHT40 (for temperature and humidity) and SGP30 (for gas emissions), the system gathers real-time data to predict ripeness stages. Machine learning models, such as CatBoost, are employed for classification, achieving high accuracy levels in ripeness detection. The proposed system aims to reduce food waste and enhance the supply chain's efficiency by providing timely insights into fruit quality.

Real-time data collection is facilitated through ESP32 microcontrollers, with information stored and visualized on the ThingSpeak cloud platform. Data preprocessing and analysis are performed using Python libraries like NumPy and Pandas, and machine learning models are trained to predict stages of ripeness. The methodology ensures a user-friendly experience and scalable design for monitoring fruits in retail or cold storage environments.

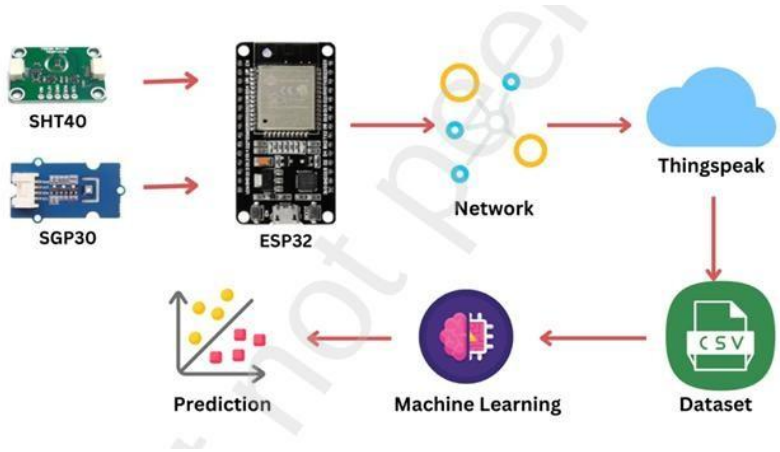


Figure 2.5 Overview of the System

The study identifies significant advantages, including improved accuracy, scalability, and the ability to integrate IoT with ML for automated quality control. However, limitations such as sensitivity to environmental noise and the system's narrow focus on bananas pose challenges. These drawbacks suggest a need for further research to generalize the framework for other fruits and applications.

Overall, this research showcases how IoT and machine learning integration can automate fruit ripeness detection, reduce post-harvest losses, and improve agricultural supply chain management. By addressing food security and minimizing waste, the platform contributes to sustainability goals and enhances operational efficiency.

2.3 Comparison

Title	Author(s) and Year	Technology Used	Methodology	Sensor, Microcontroller and Cloud
IOT Based Food Freshness Detection Using Deep Learning Techniques	Karthickeyan P, Nikesh.V, Sanjay.V, Dr. K. Devi, M.E., 2022	IoT, Deep Learning Algorithms (CNN, YOLO, ANN) Rotational Mechanism	Agile	-Raspberry Pi 4 Model B - ESP8266 - Web Camera - MQ2 Sensor - MQ135 Sensor
IoT based Fruit Quality Inspection and Lifespan Detection System	Ashim Saha, Lubaina Ali, Rudrita Rahman, Toushif Hossain, Shah Jalal Mohammad Bin Shamim, Md. Al Abid Supto, Mohammad Shidujaman, Mohammad Rejwan Uddin and Mahady Hasan 2024	IoT, Machine Learning, Data Visulization	Waterfall	- Raspberry Pi Pico - NodeMCU ESP8266 - MQ4 Sensor - DHT11 -ThingSpeak Cloud
IoT-Based Food Spoilage Detection System with UV Sterilization	P. Manjulamna, G.Sai Sangeetha, Sk. Thaseen,B. Meghana,K.N.Veena Madhuri, K.Babitha 2024	IoT, UV Sterilization, Mobile Application	Agile	- NodeMCU ESP8266 - Gas Sensor - DHT11 - IR Sensor
Fruit Freshness Detection and Monitoring using IOT	Prof. Dipti A. Gaikwad, Anish D. Pophale, Sumeet D. Palande, Keshav S. Baviskar, Nabil J. Pirjade 2024	IoT, Data Communication	Waterfall	- ESP8266 - Arduino UNO - MQ3 Sensor - Blynk
Developing an IoT and ML-driven Platform for Fruit Ripeness Evaluation and Spoilage Detection: A Case Study on Bananas	Rajini M, V Persis 2024	IoT, Machine Learning, Cloud Storage, Data Transmission	Waterfall	- ESP32 - SGP30 - SHT40 - ThingSpeak Cloud

Table 2.1 The Comparison

2.4 The Findings

The findings from all the reviewed journals collectively highlight the effectiveness of IoT and sensor-based systems in monitoring fruit freshness and spoilage. By leveraging gas sensors such MQ2, MQ4, and MQ135, environmental sensor, and advanced data analysis techniques like machine learning such as CNN, YOLO, and CatBoost, these studies demonstrate significant improvements in detecting spoilage at early stages. Key outcomes include high accuracy in predictions, real-time monitoring capabilities, and reduced food waste through timely user alerts and interventions.

While each study focused on different aspects, such as lifespan prediction, visual inspection, or additional measures like UV sterilization, all emphasized the importance of integrating IoT technology for cost-effective and scalable solutions. These systems collectively address critical challenges in food waste management, household convenience, and sustainability, paving the way for more intelligent and user-friendly produce monitoring applications.

2.5 Conclusion

In conclusion, the IoT-based fruit freshness detection system you are developing stands as a significant advancement in the realm of food quality monitoring. By integrating various sensors and visual data analysis, this project not only aligns with the methodologies outlined in the reviewed journals but also enhances them through practical applications tailored for household use. The continuous data collection and real-time notifications empower users to manage their produce effectively, thereby reducing food waste and promoting sustainability. Furthermore, by implementing suggested improvements from the literature, such as combining gas and visual data for enhanced accuracy, this system is poised to meet diverse user needs while contributing positively to environmental goals. Overall, this project exemplifies a forward-thinking approach to leveraging technology for better food management in everyday life.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This section explains the methodology used to achieve the objectives of the Rotten Fruit Detection project. A clear and well-structured approach is required to develop and implement a complete monitoring system for detecting spoiled fruits. In this project, a Wemos D1 ESP8266 microcontroller with a built-in Wi-Fi module is used to collect and sent in real time to the Node-RED platform via MQTT, where the data is processed and displayed. This section describes the key steps, including sensor setup, data transmission, prediction handling, and alert notification.

3.2 Research Methodology

Agile methodology is an ideal choice for this project because it emphasizes iterative development, flexibility, and continuous feedback, which are crucial for integrating hardware and software components in an IoT-based system. The iterative approach allows for early testing and calibration of sensors, ensuring accurate detection of spoilage indicators like gas emissions and environmental changes. Frequent iterations facilitate timely adjustments based on sensor performance, user feedback, and evolving requirements, reducing the risk of significant setbacks. Additionally, Agile’s focus on collaboration ensures that the final product meets user expectations, delivering a reliable, user-friendly system tailored to household needs. This adaptability and user-centric approach make Agile the most suitable methodology for achieving the project's objectives effectively.

Figure 3.1 Agile Methodology

3.3 Plan Phase

In the Planning Phase, the initial step involved conducting extensive research to identify a suitable title for the project that aligns with its objectives and scope. This was followed by in-depth research to gather information on similar projects, available technologies, and methodologies that could be utilized. After gathering sufficient information, the hardware components Wemos D1 ESP8266, MQ4 sensor, DHT22 sensor needed for the project were carefully selected. A detailed proposal outlining the project's goals, requirements, and implementation plan was then prepared. Finally, the proposal was submitted to the supervisor for review and approval to ensure the project was feasible and aligned with the desired outcomes.

3.3.1 Budgeting

No	Component	Unit	Cost
1	Wemos D1 ESP8266	1	RM 25.00
2	MQ4 Gas Sensor	1	RM 15.00
3	DHT22	1	RM 13.80
4	Breadboard	1	RM 3.90
5	Jumper Wire	1	RM 4.60
Total			RM 62.30

Table 3.1 Budget

3.3.2 Gantt Chart

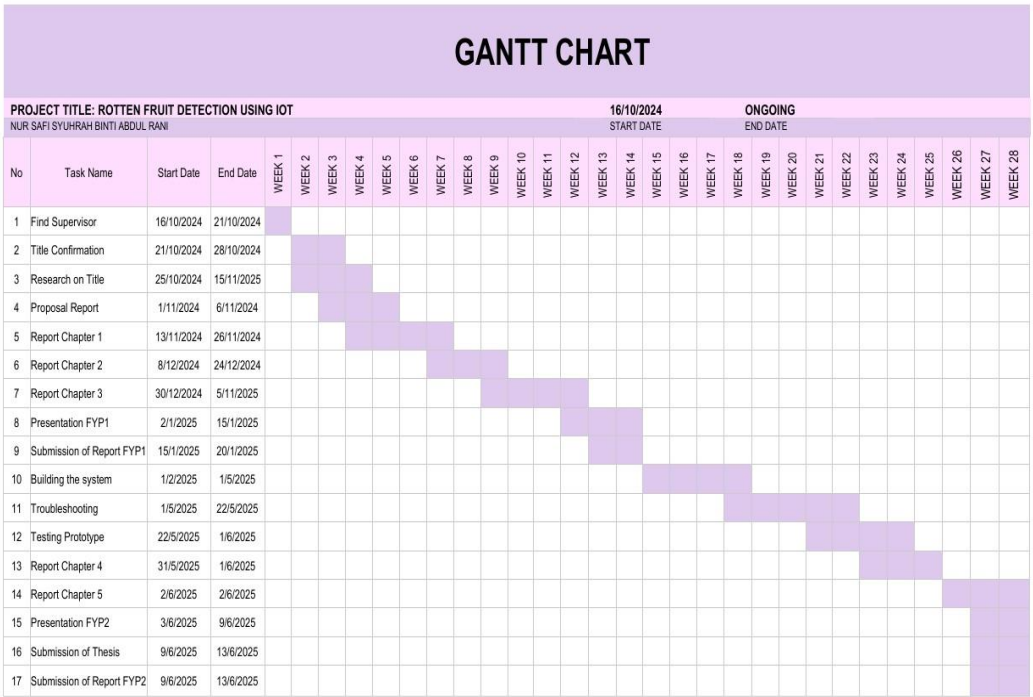


Figure 3.2 Gantt Chart

3.4 Design Phase

The design phase creates detailed specifications for the features identified in the planning phase which includes architectural design, user interface design, and any other necessary design documentation. In this phase, a block diagram and circuit diagram are designed. These designs will be helpful for the development phase. The designs will prevent any issue arise when developing the project's prototype.

3.4.1 Block Diagram

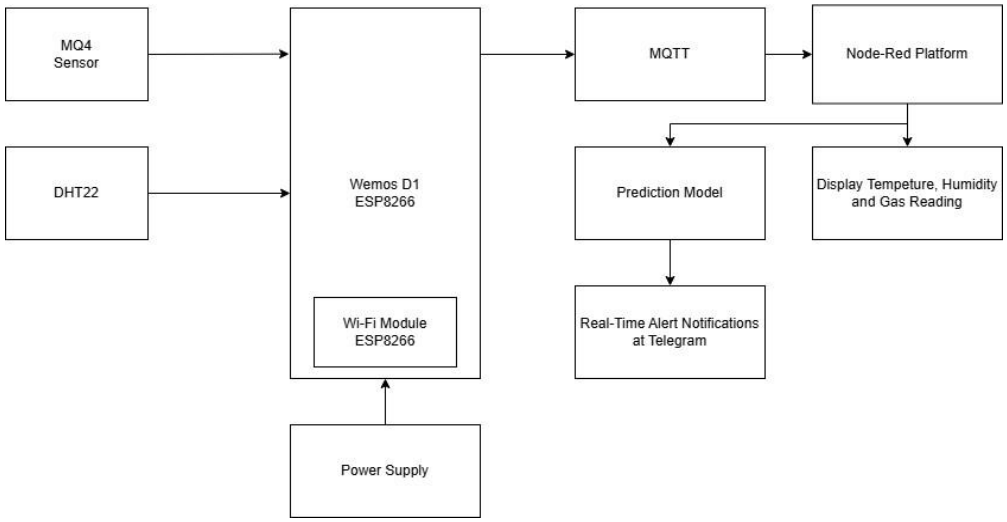


Figure 3.3 Block Diagram

This block diagram illustrates the working principle of the Rotten Fruit Detection System using IoT. The system starts with two sensors: the MQ4 gas sensor, which detects methane gas levels released from fruits, and the DHT22 sensor, which measures the surrounding temperature and humidity. These sensors are connected to the Wemos D1 ESP8266 microcontroller, which acts as the main control unit. The ESP8266 reads the data from both sensors and uses its built-in Wi-Fi module to transmit the sensor readings wirelessly using the MQTT protocol. The data is then received by the Node-RED platform, where the information is processed and visualized. On the dashboard, users can monitor real-

time values of temperature, humidity, and gas readings. Additionally, a prediction model integrated within NodeRED analyses the gas reading to predict the ripeness status of the fruit. If the prediction model detects that the fruit is rotten or overripe, the system will immediately send a real-time alert notification to the user via Telegram.

3.4.2 Flowchart

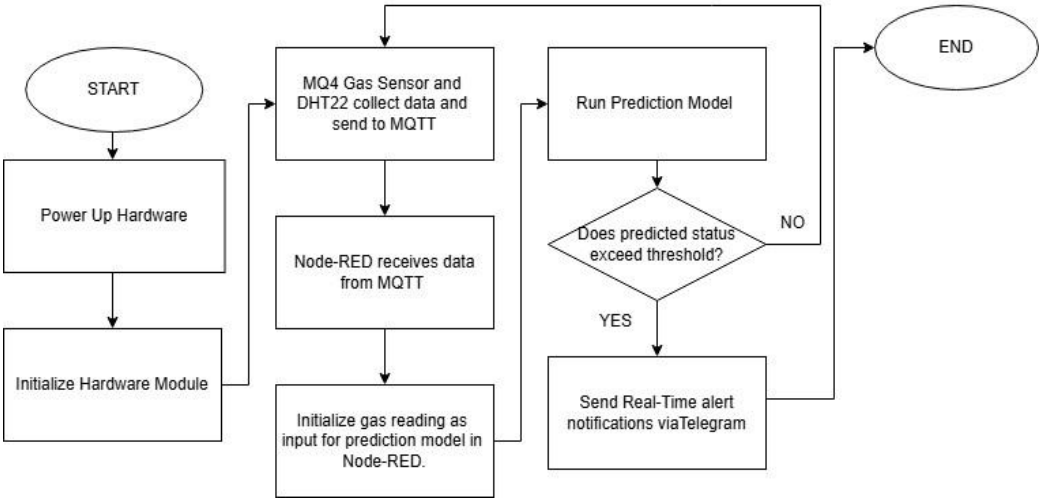


Figure 3.4 Flowchart

The flowchart describes the operational steps of the Rotten Fruit Detection using IoT. The system begins by powering up the hardware and initializing the MQ4 sensor and DHT22. Once initialized, the sensors begin collecting environmental data which is gas emission, temperature, and humidity. These readings are then sent wirelessly to the ESP8266 module via the MQTT protocol. The Node-RED platform receives the data and processes it. Within Node-RED, a prediction model is triggered, which uses the gas readings to predict the status of the fruit. All sensor values are displayed on a dashboard, allowing the user to monitor the environment in real time. If the model determines that the fruit is rotten or about to rot, Node-RED will send a real-time Telegram notification to alert the user. If no rot is detected, the system continues to monitor as normal.

3.5 Develop Phase

The hardware setup begins by connecting the Wemos D1 ESP8266 microcontroller to the MQ4 gas sensor and the DHT22 temperature and humidity sensor for data collection. Once the connections are established, the Wemos D1 is configured to send sensor data wirelessly using the MQTT protocol to enable real-time monitoring and data processing and display the data on Node-RED.

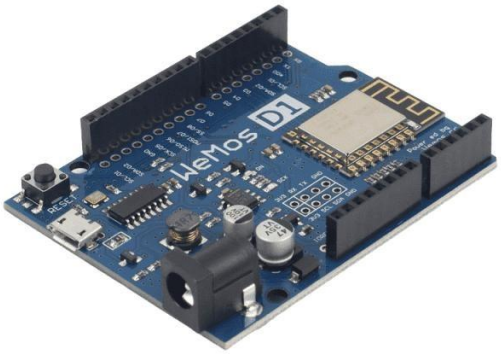


Figure 3.5 Wemos D1 ESP8266

The MQTT protocol to facilitate efficient and reliable wireless communication between the Wemos D1 microcontroller and the Node-RED platform using Mosquitto. Its lightweight messaging system ensures continuous real-time data transmission with minimal bandwidth consumption.

Figure 3.6 MQTT Protocol and Mosquitto

The software development focuses on creating a smart system that integrates sensor data to detect and monitor fruit spoilage conditions. Node-RED is used as the central platform to receive sensor readings, run the prediction model, and display the results on a real-time dashboard. The system automatically analyses the data using a model trained in RapidMiner and executed through Node-RED, providing alerts when spoilage is detected and allowing users to monitor fruit storage conditions effectively.

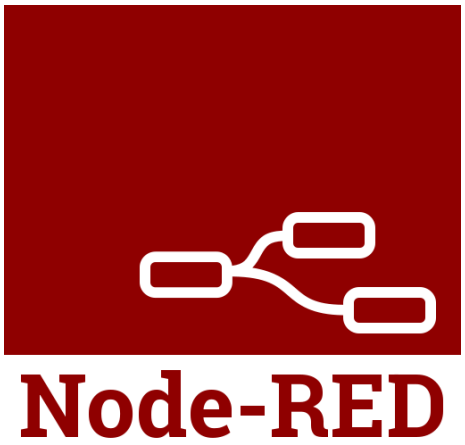


Figure 3.7 Node-RED

The MQ4 gas sensor detects methane gas, which increases significantly during the advanced stages of fruit spoilage. This sensor provides critical information on gas emissions, helping the system identify early signs of fruit decay with high sensitivity and accuracy.

Figure 3.8 MQ4-Gas Sensor

The DHT22 sensor measures ambient temperature and humidity, which are important environmental factors affecting fruit freshness. The accurate data from the DHT22 helps improve the precision of spoilage detection by factoring in storage conditions.

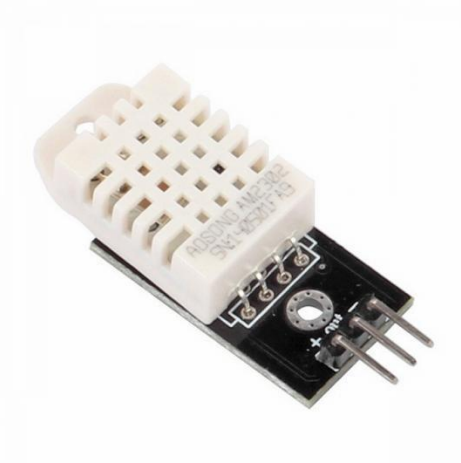


Figure 3.9 DHT22

RapidMiner is used to develop the prediction model for this project. Based on the collected sensor data specifically methane gas level. RapidMiner trains a machine learning model to classify the fruit condition as fresh, overripe, or rotten. This model is then applied within the system to automatically analyse new incoming data and provide accurate spoilage predictions in real time.

Figure 3.10 Altair RapidMiner

Telegram is used to send real-time notifications to users based on the sensor data and prediction results. When the fruit condition is predicted as overripe or rotten, a message is automatically sent via Telegram. This alert includes the current fruit status and the gas reading, allowing users to take immediate action. The integration with Node-RED ensures that Telegram notifications are triggered efficiently and reliably as part of the smart monitoring system.

Figure 3.11 Telegram

The software development integrates the connected sensors into a smart monitoring system that uses Node-RED to visualize and process real-time data. Through a webbased dashboard, users can monitor methane gas levels, temperature, and humidity with ease. When sensor readings exceed predefined thresholds or when the prediction model detects spoilage, automated alerts are

sent via Telegram to notify users, including the current fruit status and gas reading. The MQTT protocol ensures secure and efficient communication between the Wemos D1 ESP8266 and the Node-RED server. This setup delivers a smooth and reliable user experience, combining accurate sensor input, intelligent prediction through RapidMiner, and real-time notifications to help reduce fruit spoilage and food waste

3.5.1 The Schematic Diagram

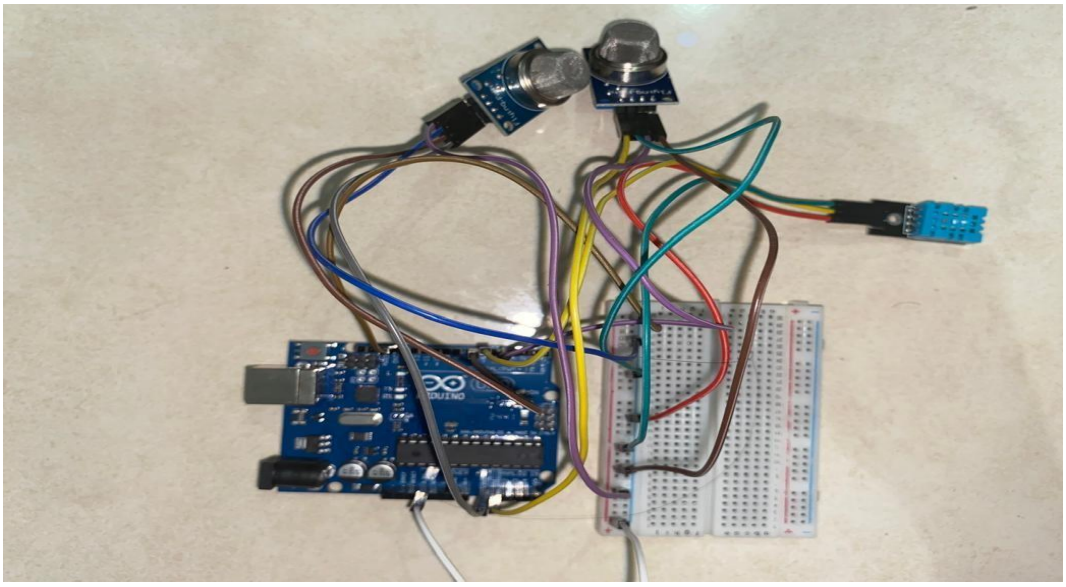


Figure 3.12 First Prototype

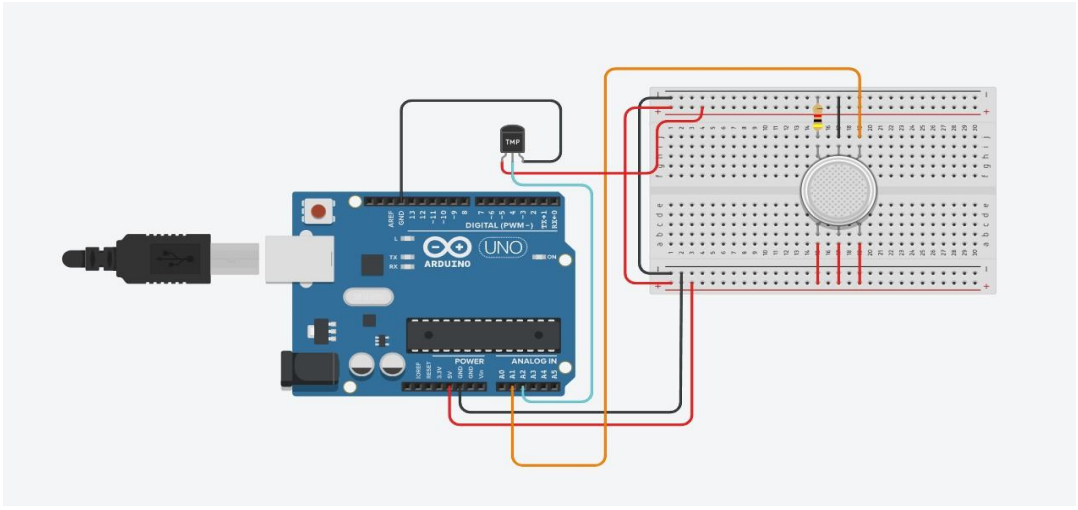


Figure 3.13 The Schematic Diagram

3.6 Test Phase

In the testing phase, the prototype developed in the previous phase will be tested using any selection of fruits such as tomato or banana to check its performance. If any errors or issues are found during this phase, the prototype will be redesigned and redeveloped to fix them. So far, the gas sensor has been tested using the TinkerCad simulation, and Figure 16 shows the results from this testing.

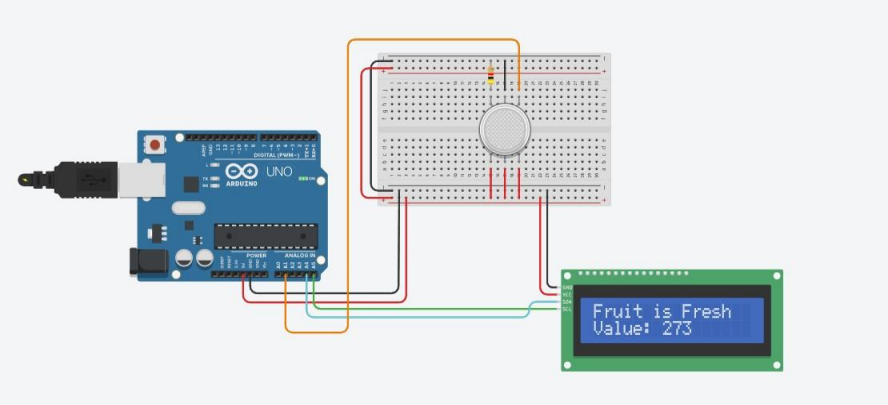


Figure 3.14 Fruit is Fresh

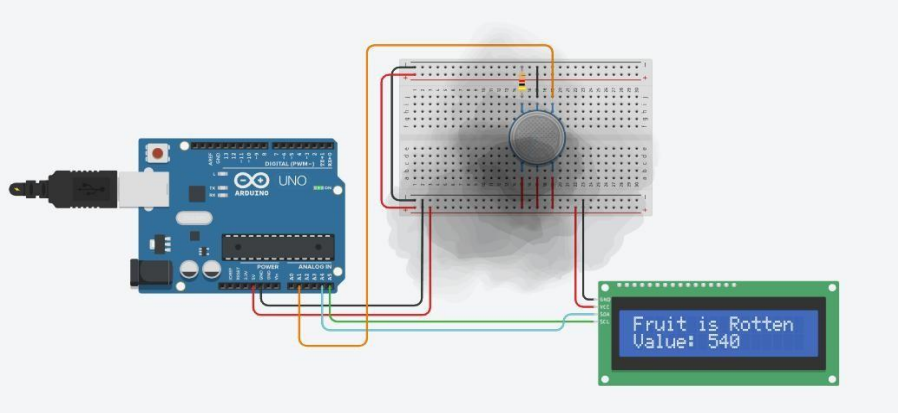


Figure 3.15 Fruit is Rotten

When the gas sensor detects a specific value, the LCD will display a message indicating whether the fruit is fresh or rotten, helping people decide if it is safe to consume.

3.7 Deploy Phase

The completed prototype is installed and tested in a real household environment to ensure it functions as expected. The system is set up with the sensors and users can monitor fruit conditions through the dashboard and receive notifications. Any final adjustments are made based on real-world performance to optimize usability and reliability for daily use.

3.8 Review Phase

In this phase, the entire project is evaluated to ensure it meets the initial objectives and requirements. Feedback is collected from users to identify any areas for improvement, and the system's performance is assessed in real-world conditions. Any necessary adjustments or enhancements are made to refine the system further. This phase ensures the project delivers a reliable, user-friendly solution that effectively reduces fruit spoilage and supports household sustainability.

3.9 Launch Phase

The final version of the system is officially released for use by households. All components, including hardware and software, are fully operational and tested. Users are provided with instructions for setup and usage, and the system is made available for broader adoption. This phase marks the transition from development to real-world application, ensuring the solution is accessible and ready to help reduce fruit spoilage effectively.

3.10 Conclusion

In conclusion, this chapter outlined the methodology for developing the IoT-based rotten fruit detection system using the Agile approach. The iterative nature of Agile ensures continuous improvements and adaptability throughout the project development process. Each phase, from requirement gathering to testing, is structured to allow flexibility, enabling enhancements based on feedback and testing results. The integration of MQ4 gas sensors, DHT22, and ESP8266 ensures real-time monitoring of fruit spoilage, while the user interface facilitates easy access to data. By following the Agile methodology, this project ensures an efficient, user-centered, and scalable solution for minimizing household food waste.

CHAPTER 4: RESULT AND DISCUSSION

4.1 Introduction

This following chapter describes result and discussion of Rotten Fruit Detection using IoT if it fulfils the project objectives. Initial testing for the system has been done prior before the full system testing. All the system's performance has been recorded in the section.

4.2 Project Testing

4.2.1 Connectivity Testing

The initial testing was conducted using an Arduino UNO connected to a DHT11 sensor and an MQ4 gas sensor. The system functioned correctly, marking the beginning of the data collection process.

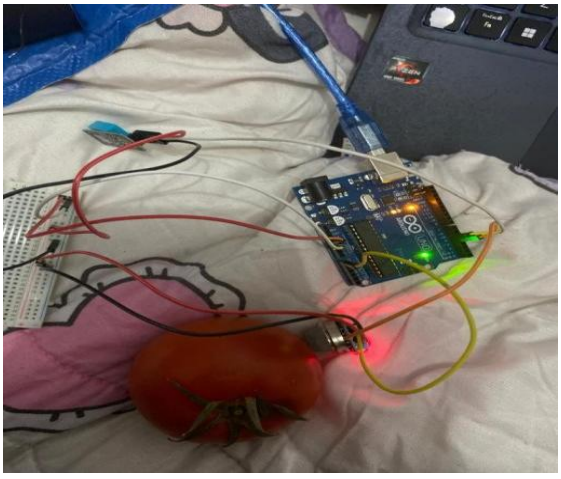


Figure 4.1 First Prototype Testing

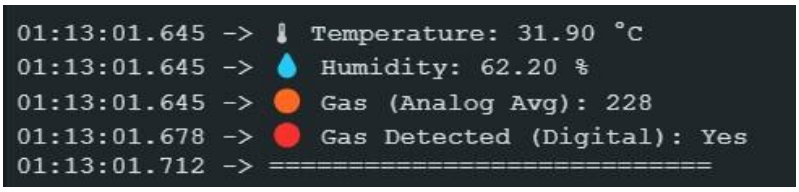


Figure 4.2 Serial Print IDE

4.2.2 Blynk Dashboard

The initial dashboard testing was conducted using Blynk, where notifications were triggered when gas readings exceeded the predefined threshold. However, after further research, it was discovered that Blynk does not support direct integration with RapidMiner, limiting its ability to perform realtime predictions using the trained model. As a result, the decision was made to switch to Node-RED, which offers better flexibility and integration capabilities for implementing the predictive model.

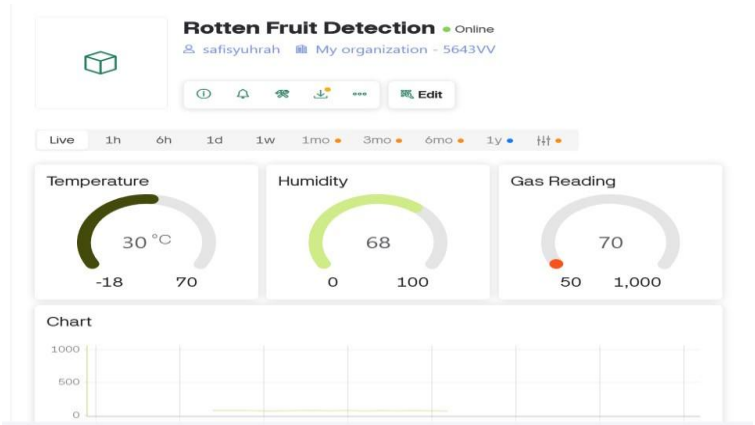


Figure 4.3 Blynk Dashboard

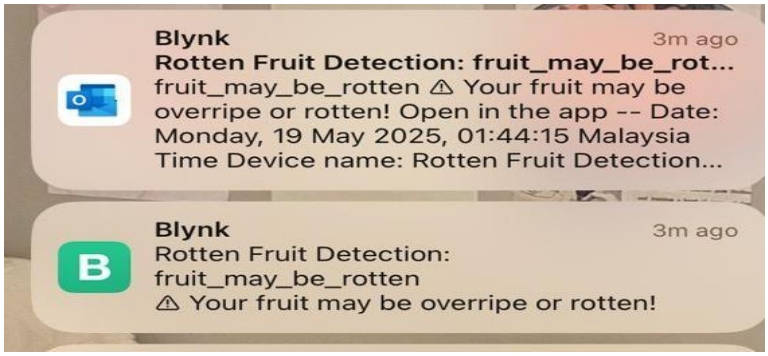


Figure 4.4 Blynk Notification

4.3 Collection Data

The data was collected from seven types of fruit: strawberry, blueberry, cucumber, mango, grapes, tomato, and banana. Daily observations were made by recording gas readings to monitor changes in the fruits over time, from fresh to rotten. These readings were consistently collected each day and saved in an Excel file for further analysis. The collected dataset was then used to develop a prediction model capable of identifying the fruit's ripeness level based on the gas readings.

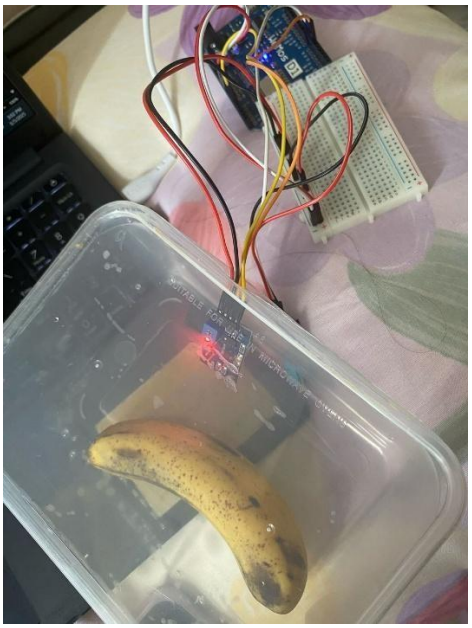


Figure 4.5 Banana During Gas Data Collection

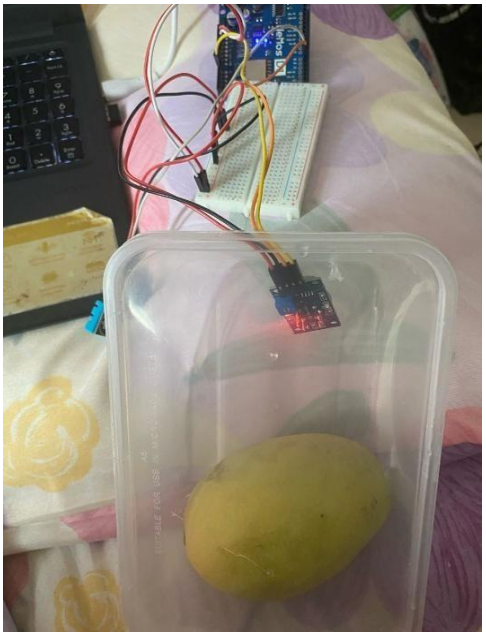


Figure 4.6 Mango During Gas Data Collection

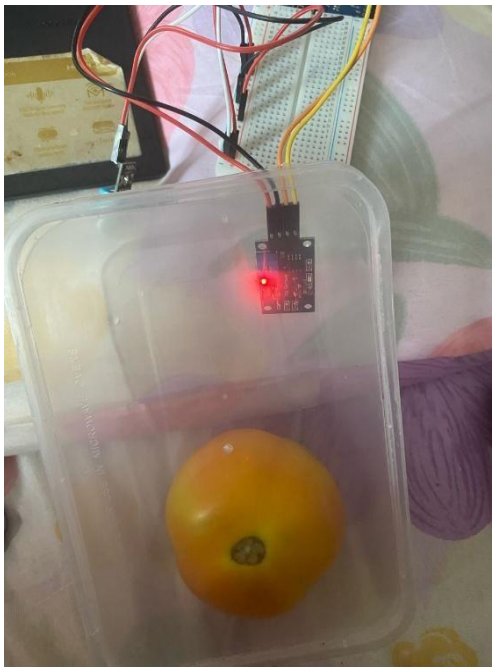


Figure 4.7 Tomato During Gas Data Collection

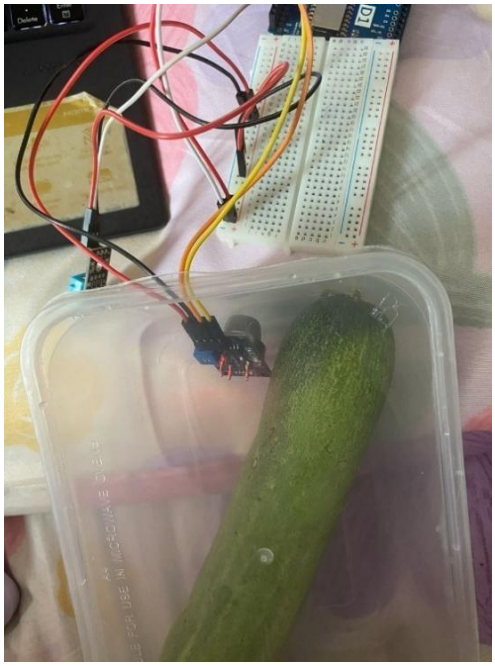


Figure 4.8 Cucumber During Gas Data Collection

Figure 4.9 Strawberry During Gas Data Collection

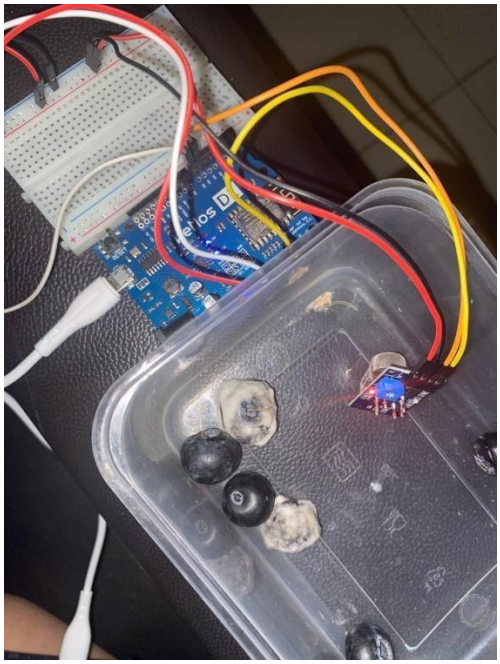


Figure 4.10 Blueberry During Gas Data Collection

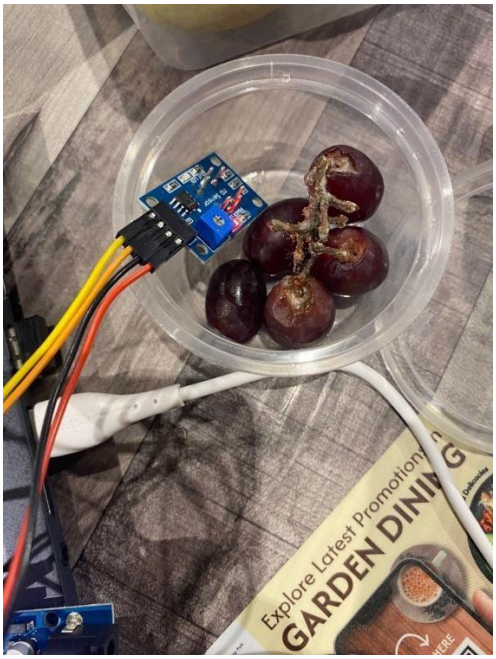
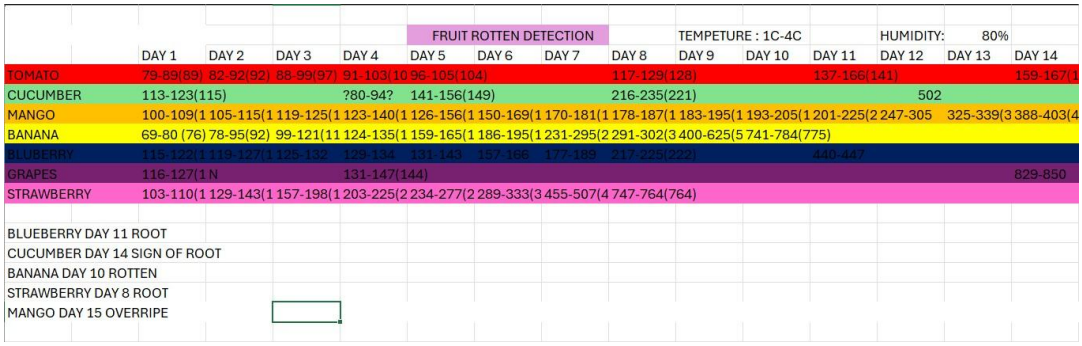


Figure 4.11 Grapes During Gas Data Collection



					FRUIT ROTTEN DETECTION				TEMPETURE : 1C-4C			HUMIDITY:	80%	
	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	DAY 8	DAY 9	DAY 10	DAY 11	DAY 12	DAY 13	DAY 14
TOMATO	79-89(89)	82-92(92)	88-99(97)	91-103(10	96-105(104)			117-129(128)			137-166(141)			159-167(1
CUCUMBER	113-123(115)			?80-94?	141-156(149)			216-235(221)				502		
MANGO	100-109(1	105-115(1	119-125(1	123-140(1	126-156(1	150-169(1	170-181(1	178-187(1	183-195(1	193-205(1	201-225(2	247-305	325-339(3	388-403(4
BANANA	69-80 (76)	78-95(92)	99-121(11	124-135(1	159-165(1	186-195(1	231-295(2	291-302(3	400-625(5	741-784(775)				
BLUBERRY	115-122(1	119-127(1	125-132	129-134	131-143	157-166	177-189	217-225(222)			440-447			
GRAPES	116-127(1	N		131-147(144)										829-850
STRAWBERRY	103-110(1	129-143(1	157-198(1	203-225(2	234-277(2	289-333(3	455-507(4	747-764(764)						

BLUEBERRY DAY 11 ROOT
CUCUMBER DAY 14 SIGN OF ROOT
BANANA DAY 10 ROTTEN
STRAWBERRY DAY 8 ROOT
MANGO DAY 15 OVERRIPE

Figure 4.12 First Excel Data Collection

4.4 RapidMiner: Prediction Model

All the collected data was transferred to an Excel CSV file, consisting of four attributes: Fruit, Day, Gas Reading, and Status. This dataset was then uploaded into RapidMiner to develop a prediction model for determining the ripeness status of the fruits.

Fruits	Gas Reading	Day	Status
Strawberry	106.5	1	Fresh
Strawberry	136	2	Fresh
Strawberry	177.5	3	Fresh
Strawberry	214	4	Fresh
Strawberry	255.5	5	Overripe
Strawberry	311	6	Overripe
Strawberry	481	7	Overripe
Strawberry	755.5	8	Root
Blueberry	118.5	1	Fresh
Blueberry	123	2	Fresh
Blueberry	128.5	3	Fresh
Blueberry	131.5	4	Fresh
Blueberry	137	5	Fresh
Blueberry	161.5	6	Fresh
Blueberry	183	7	Fresh
Blueberry	221	8	Overripe
Blueberry	255	9	Overripe
Blueberry	329	10	Overripe
Blueberry	443.5	11	Root
Banana	74.5	1	Fresh
Banana	86.5	2	Fresh
Banana	110	3	Fresh

Figure 4.13 Latest Excel Data Collection

Several RapidMiner operators were used to develop the prediction model. The process began with the Read CSV operator to import the dataset, followed by the Set Role operator to assign the correct roles to each attribute. The Decision Tree algorithm was selected to build the predictive model due to its simplicity and interpretability. The Cross Validation operator was applied to evaluate the model's performance. Additionally, the Execute Python operator was used to run custom Python scripts, further enhancing the model's capabilities. This workflow enabled the successful development and refinement of the predictive model for the project.

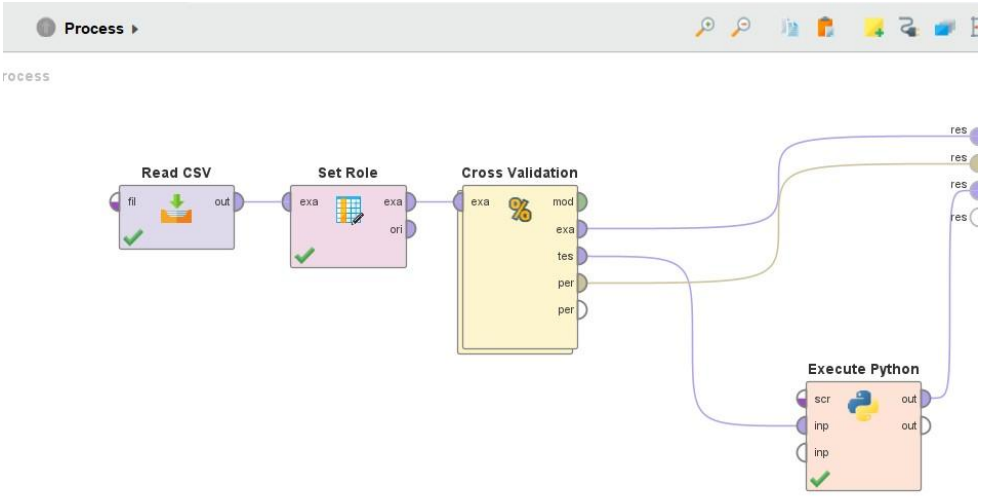


Figure 4.14 Process in RapidMiner

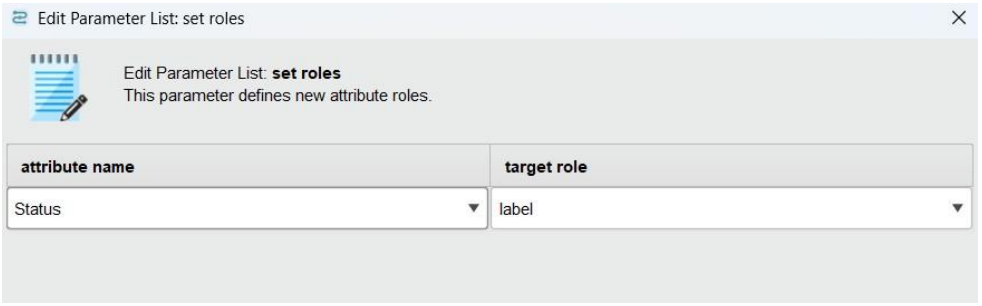


Figure 4.15 Attributes selected in Set Roles

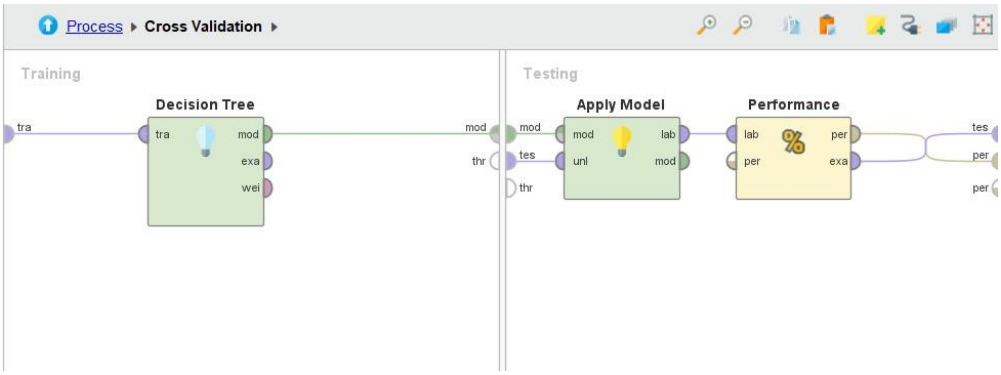


Figure 4.16 Inside Cross Validation

The model achieved an overall accuracy of 91.09% ± 8.79%. It performed best in identifying the Fresh class, with a precision of 90.88% and a recall of 92.54%. The model also showed good performance for the Overripe class, achieving 83.87% precision and 82.90% recall. However, it struggled to accurately classify the Rot class, with a precision of 71.43% and a recall of 71.43%, likely due to class imbalance in the dataset. To improve performance, future work could focus on handling class imbalance using oversampling methods, adding more relevant features, or fine-tuning the model's parameters to achieve better accuracy across all ripeness levels.

accuracy: 91.09% +/- 8.75% (micro average: 91.18%)

	true Fresh	true Overripe	true Root	class precision
pred. Fresh	62	1	1	96.88%
pred. Overripe	4	26	1	83.87%
pred. Root	1	1	5	71.43%
class recall	92.54%	92.86%	71.43%	

Figure 4.17 Prediction Result

The file model.pkl is a pickled Python object generated using the Execute Python operator within the RapidMiner workflow. This .pkl file contains the trained machine learning model used for fruit ripeness prediction. Saving the model in this format allows it to be reloaded and utilized in external systems. This model was integrated into the Rotten Fruit Detection system using IoT and Node-RED, enabling real-time predictions and improving the overall accuracy and efficiency of the system.

Figure 4.18 .pkl File

4.5 MQTT Connection

The connection between the IoT device and Node-RED was established using the MQTT communication protocol. For this project, the Mosquitto broker was selected, operating on port 1883. Real-time data from the system is published to specific MQTT topics, which can be monitored using MQTT Explorer. The topic names were defined and initialized within the device's source code to ensure accurate communication and data flow between the IoT hardware and the Node-RED dashboard.

```
// ========== MQTT Config ==========
const char* mqtt_server = "test.mosquitto.org";
const char* mqtt_topic_temp = "fruit_data/temperature";
const char* mqtt_topic_humidity = "fruit_data/humidity";
const char* mqtt_topic_gas = "fruit_data/gas";
```

Figure 4.19 MQTT Config in IDE

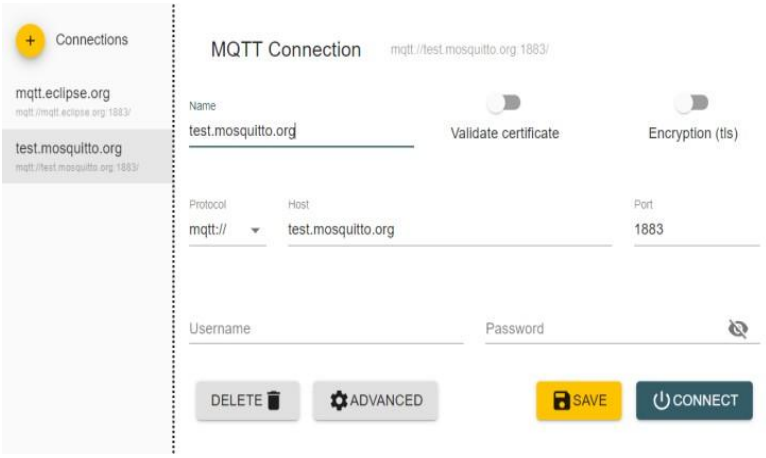


Figure 4.20 MQTT Explorer

4.6 Node-RED Dashboard and Telegram

The Node-RED flow shown above illustrates the data collection and processing system for the Rotten Fruit Detection using IoT project. Three types of real-time sensor data gas, temperature, and humidity are published to specific MQTT topics from the IoT device. These topics are subscribed to by Node-RED using MQTT input nodes. The data is visualized on a dashboard through labelled nodes such as Gas Reading, Temperature (°C), and Humidity (%), providing users with a clear view of the fruit environment in real time.

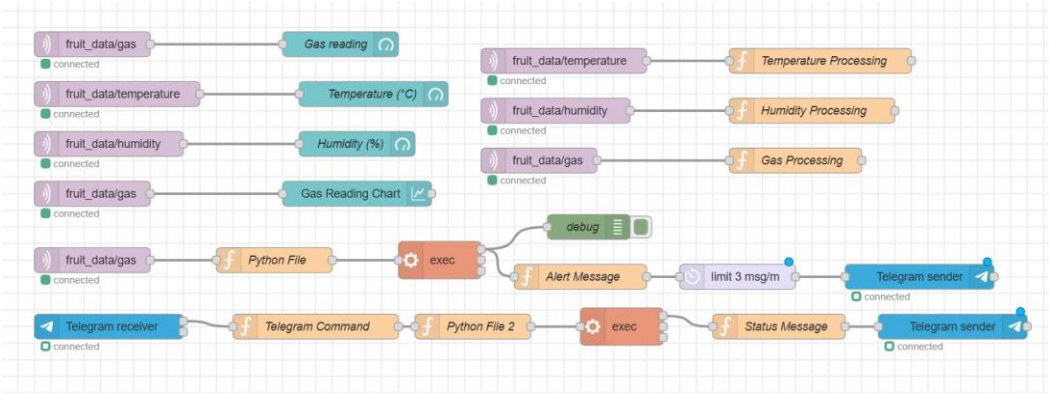


Figure 4.21 Node-RED Flow

Once the data is received, it passes through dedicated Function nodes—Temperature Processing, Humidity Processing, and Gas Processing—which likely prepare and structure the data for analysis. The processed gas data is forwarded to a Function node labelled Python File, which constructs a command to execute a Python script. This script loads the trained RapidMiner model saved as model.pkl and makes predictions about the fruit's ripeness. The prediction result is returned via the exec node, and a message is generated in the Alert Message node to inform users of the fruit's status.

To enhance user interaction, the system is integrated with Telegram for sending notifications. If the predicted result indicates an undesirable condition, an alert is sent via the Telegram sender node. Additionally, a Telegram bot allows users to send commands, which are handled through the Telegram receiver and Telegram Command nodes. These commands trigger a second Python script via Python File 2 and exec, sending back a Status Message to keep the user informed. This integration ensures that users are notified promptly and can check fruit status remotely, making the system highly interactive and effective for real-time fruit monitoring.

4.7 Result

The fruit ripeness monitoring system was designed to work in real time using IoT integration and a machine learning model. The system begins by collecting environmental data specifically gas concentration, temperature, and humidity from the fruits using a Wemos D1 ESP8266 connected to an MQ4 gas sensor and a DHT22 sensor.

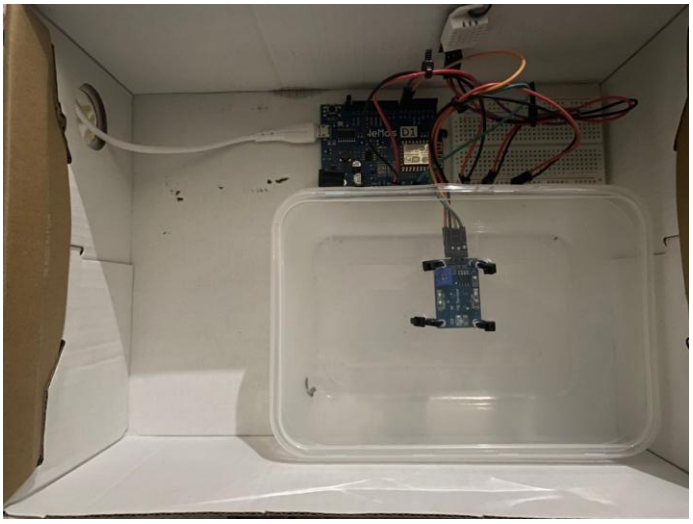


Figure 4.22 Prototype of The Project

Once the data is collected, it is transmitted wirelessly to an MQTT broker. The broker acts as a messaging server where sensor data is published under specific topics. The data transmission and topic activity are monitored in real time using MQTT Explorer, confirming that the system is actively sending values.

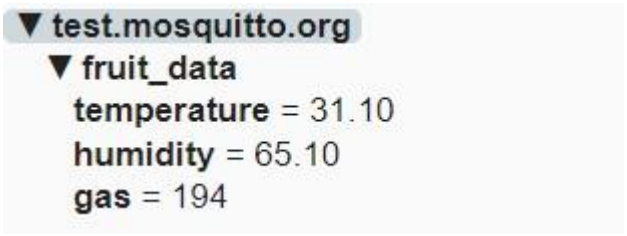


Figure 4.23 MQTT Dashboard

Node-RED is configured to subscribe to the relevant MQTT topics. The system is configured to subscribe to these MQTT topics. When the data is received, it is displayed on the Node-RED dashboard, showing live readings of gas levels, humidity, and temperature. Simultaneously, a machine learning model is triggered to analyse the data. As soon as the data is received, a predefined prediction model is executed using a Python script. This model classifies the current fruit condition into one of three categories: Fresh, Overripe, or Rotten.

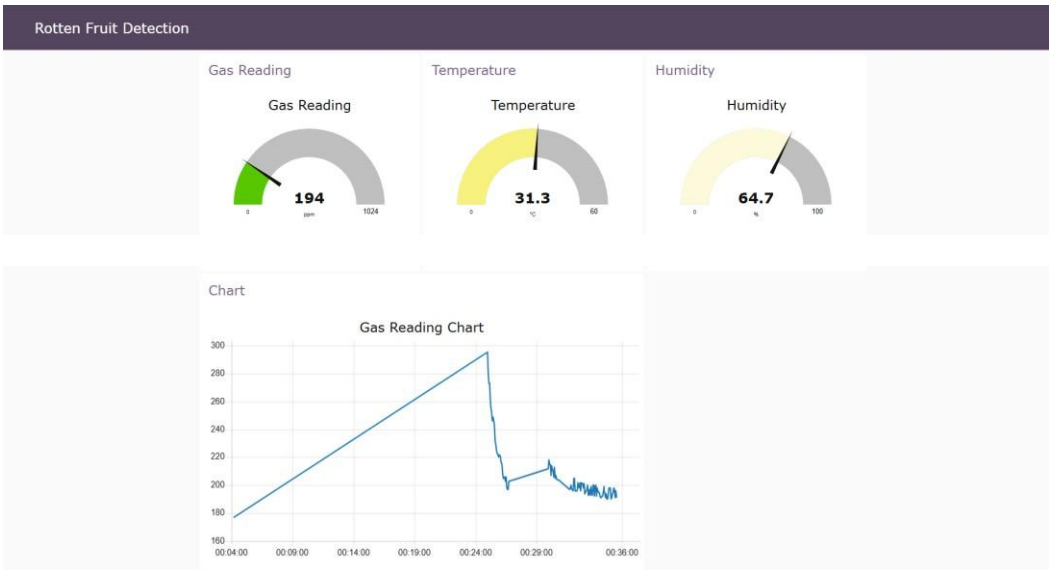


Figure 4.24 Node-RED Dashboard

To enhance usability, the system is integrated with Telegram for real-time alerts and remote monitoring. If the fruit condition is predicted to be Overripe or Rotten, NodeRED sends an automatic notification to the user via a Telegram bot. Additionally, users can interact with the system by sending specific commands to receive the latest fruit condition, making the system easily accessible via mobile devices.

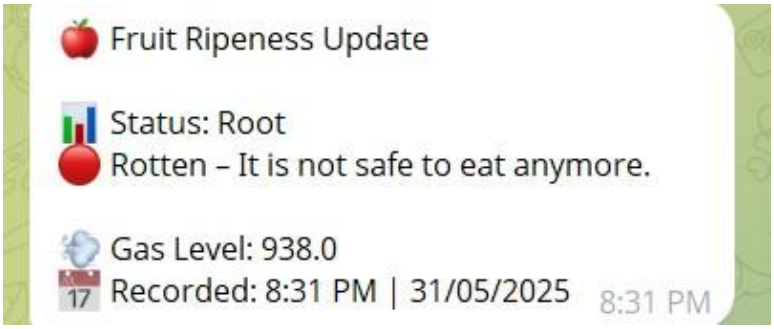


Figure 4.25 Telegram Alert (Rotten)

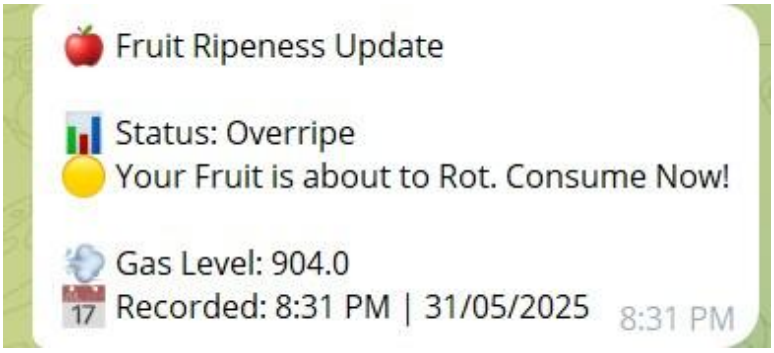


Figure 4.26 Telegram Alert (Overripe)

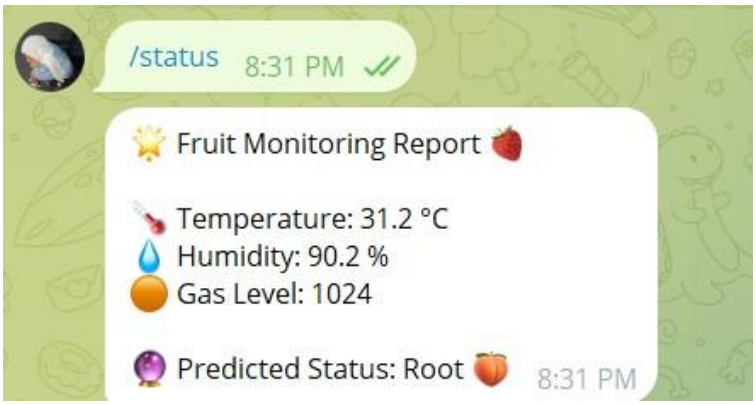


Figure 4.27 Telegram Command

The system runs continuously and updates the prediction based on the latest sensor data. This allows for early detection of spoilage and timely user alerts. Overall, the system worked effectively in real-time prediction, data visualization, and user notification, demonstrating its potential for applications in food monitoring, storage, and safety

4.8 Discussion

The fruit ripeness detection system successfully integrated IoT hardware, real-time communication, and a machine learning model to classify fruit conditions with high accuracy. The seamless connection between the sensors, MQTT broker, Node-RED, and Telegram enabled the system to operate reliably and provide timely alerts to users. The ability to remotely monitor fruit ripeness via Telegram commands adds convenience and practicality, especially for users managing perishable items.

The prediction model showed strong performance, especially for Fresh and Overripe classifications. However, the system faced challenges in accurately identifying the Rotten class. This is likely due to class imbalance in the training data, as rotten fruits were more limited compared to fresh and overripe ones. This imbalance affected the model’s ability to generalize well to all categories, particularly when deployed in the real-time system through Node-RED.

Despite this limitation, the integration of the prediction model into the live Node-RED environment resulted in improved accuracy compared to the initial RapidMiner testing. This improvement could be due to better data handling and real-time processing. Still, further enhancement is possible by addressing class imbalance through oversampling, collecting more data for the Rotten class, or by tuning the model with additional features like humidity and temperature trends.

Overall, the system demonstrated the potential for real-world applications in agriculture, food storage, and logistics. Future work can focus on improving model performance, adding more fruit types, expanding the Telegram bot's functions, and potentially integrating cloud storage or a mobile app for broader usability.

4.9 Conclusion

The system successfully monitored fruit ripeness in real time using gas, temperature, and humidity sensors. Data was sent through MQTT, processed in Node-RED, and displayed on a dashboard. The integrated prediction model classified fruit conditions accurately, and Telegram alerts allowed users to monitor remotely. Overall, the system proved effective and practical for early spoilage detection.

CHAPTER 5: CONCLUSION

5.1 Introduction

This chapter concludes the project report on the Rotten Fruit Detection using IoT. It summarizes the key findings, evaluates the achievement of project objectives, and highlights the overall significance and potential impact of the system. The integration of IoT, machine learning, and real-time notification in this project offers a smart solution to monitor fruit conditions efficiently, helping reduce food waste and improve storage management.

5.2 Objectives Achievement

The project aimed to develop a smart system that could monitor and predict fruit ripeness levels using IoT sensors and machine learning. The following outcomes reflect the successful achievement of the core objectives:

- **Real-Time Data Collection Using IoT Sensors:**
  The system was successfully developed using Wemos D1 ESP8266 with an MQ4 gas sensor and a DHT22 temperature and humidity sensor. These sensors collected environmental data crucial for ripeness classification.

- **Reliable Data Transmission through MQTT:**
  The data collected by the sensors was transmitted using the MQTT protocol via the Mosquitto broker. This ensured real-time, lightweight, and reliable communication between the IoT device and the processing unit.

- **Visualization in Node-RED Dashboard**:
  Live dashboard was created in Node-RED to display gas, temperature, and humidity values. This allowed users to monitor fruit conditions visually in real time.

- **Accurate Fruit Ripeness Prediction Model:**
  Prediction model was built using RapidMiner and integrated into NodeRED. The model achieved an accuracy of 91.09% ± 8.75%, showing strong performance in identifying Fresh and Overripe fruit conditions.

- **Telegram Notification Integration**:
  The system integrated Telegram to send alerts to users when fruit reached Overripe or Rotten conditions. Telegram commands also allowed users to request the latest fruit status, improving interactivity and monitoring convenience.

5.3 Summary

The Rotten Fruit Detection using IoT developed in this project demonstrated the feasibility and benefits of integrating real-time sensing, machine learning, and user notification. The key achievements include:

- **Accurate prediction model** with 91.09% accuracy.
- **Real-time monitoring** of gas, temperature, and humidity via Node-RED.
- **Effective user alerts** through Telegram bot.
- **Seamless data transmission** using MQTT and Mosquitto broker.
- **Fully automated system** from data collection to user notification.

This system provides a practical solution for fruit storage and quality monitoring, especially in household and commercial settings. By enabling early detection of spoilage, it helps reduce waste, improve decision-making, and support better inventory management.

5.4 Future Recommendations

The success of this project opens several avenues for future research and development:

1. **Expansion to Other Types of Produce**
   The concept and technologies developed in this project can be extended to monitor and predict ripeness for other fruits and vegetables with varying characteristics, such as bananas, mangoes, or even leafy greens, by training the system with broader datasets.

2. **Integration with Smart Storage Systems**
   Future enhancements could involve integrating this system with smart storage units like refrigerators or storage shelves, enabling automatic adjustments in temperature or humidity to preserve fruit freshness.

3. **Improvement of Machine Learning Model**
   Although the Decision Tree model provided promising results, future work could explore more advanced algorithms such as Random Forest, SVM, or deep learning to increase prediction accuracy, especially for the "Rotten" class.

4. **Enhanced User Interface and Feedback**
   The Telegram bot interaction could be expanded to support voice commands, visual dashboard summaries, or integration with mobile applications to improve usability and user engagement.

5.5 System Limitations

While the developed system has shown high accuracy and functionality in predicting the ripeness of certain fruits, it also has some limitations. One of the main weaknesses observed is the system's inability to accurately detect the ripeness status of blueberries.

The gas reading for rotten blueberries is typically around 400+ ppm, whereas the prediction model in this system tends to classify a reading of 700+ ppm or higher as indicating a rotten condition. This mismatch results in the system failing to detect rotten blueberries accurately, as it does not associate the lower gas value with spoilage for that fruit type.

This limitation highlights the need for fruit-specific calibration in future versions of the system. Expanding the training dataset to include a wider variety of fruits and their corresponding gas emission patterns during ripening and spoilage would improve the model's versatility and reliability across different produce types.

5.6 Conclusion

In conclusion, this project successfully demonstrated the feasibility and benefits of an IoT-based system for real-time fruit ripeness monitoring and prediction. By combining environmental sensors, machine learning, and instant user notifications through Telegram, the system offers a practical and scalable solution for reducing food waste and enhancing food quality monitoring. The project serves as a foundation for future developments in smart agriculture, food safety, and intelligent home systems.

REFERENCES

[1] Persis, V. (n.d.). *Developing an IoT and ML-driven Platform for Fruit Ripeness Evaluation and Spoilage Detection: A Case Study on Bananas*. https://ssrn.com/abstract=4838544

[2] Saha, A., Ali, L., Rahman, R., Hossain, T., Mohammad Bin Shamim, S. J., Supto, M. A. A., Shidujaman, M., Uddin, M. R., & Hasan, M. (2023). IoT Based Fruit Quality Inspection and Lifespan Detection System. *2023 7th International Conference On Computing, Communication, Control And Automation, ICCUBEA 2023*. https://doi.org/10.1109/ICCUBEA58933.2023.10392254

[3] Manjulamna, P., Sai Sangeetha, G., Thaseen, S., Meghana, B., Madhuri, K. N. V., & Babitha, K. (2024). IOT-BASED FOOD SPOILAGE DETECTION SYSTEM WITH UV STERILIZATION. In *International Journal of Research in Engineering, IT and Social Sciences* (Vol. 14). http://indusedu.org

[4] Gaikwad, D. A., Pophale, A. D., Palande, S. D., Baviskar, K. S., & Pirjade, N. J. (n.d.). FRUIT FRESHNESS DETECTION AND MONITORING USING IOT. www.irjmets.com

[5] KarthickeyanP, B., Devi, K., & Professor, A. (2022). IOT Based Food Freshness Detection Using Deep Learning Techniques.

[6] Zakaria, A., Rahman, A., & Rahim, A. (2014). AN OVERVIEW OF FRUIT SUPPLY CHAIN IN MALAYSIA (Vol. 37).

[7] Kumar, V., & Chambail, S. (2024). Kumar & Chambail 61. International Journal of Electrical, Electronics P.G College Dharamshala, 13(2), 61–63. www.researchtrend.net

APPENDIX A : SOURCE CODE

A.1 ARDUINO IDE SENSOR CODE

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

// ========== WiFi Config ==========
const char* ssid = "Infisya Family";
const char* password = "R4542172510L";

// ========== MQTT Config ========== const char*
mqtt_server = "test.mosquitto.org";  const char*
mqtt_topic_temp = "fruit_data/temperature"; const char*
mqtt_topic_humidity = "fruit_data/humidity"; const char*
mqtt_topic_gas = "fruit_data/gas";

WiFiClient espClient;
PubSubClient client(espClient);

// ========== Sensor Config ==========
#define DHTPIN D5
#define DHTTYPE DHT22
#define MQ4_ANALOG A0
#define MQ4_DIGITAL D4
```

```
DHT dht(DHTPIN, DHTTYPE);

// ========== Function to Get Averaged Gas Reading ========== int
getAverageGasValue(int samples = 10) {
 int total = 0;   for (int i = 0; i <
samples; i++) {
   total += analogRead(MQ4_ANALOG);
   delay(20);
  }
  return total / samples;
}
// ========== Setup ==========
void setup() {
Serial.begin(9600);   delay(10);

  dht.begin();
  pinMode(MQ4_DIGITAL, INPUT);

  Serial.println();
  Serial.println("  Connecting to WiFi...");
WiFi.begin(ssid, password);   while
(WiFi.status() != WL_CONNECTED) {
delay(500);
    Serial.print(".");
  }
  Serial.println("\n  WiFi connected");
```

```
  client.setServer(mqtt_server, 1883);

  Serial.println("  Warming up MQ-4 sensor..."); delay(30000);
  Serial.println("  MQ-4 ready.");

 reconnect();
}
// ========== WiFi Reconnect ==========
void checkWiFiConnection() {  if
(WiFi.status() != WL_CONNECTED) {
   Serial.println("  WiFi disconnected. Reconnecting...");
WiFi.begin(ssid, password);    while (WiFi.status() !=
WL_CONNECTED) {      delay(500);
     Serial.print(".");
   }
   Serial.println("\n  WiFi reconnected.");
  }
}
// ========== MQTT Reconnect ==========
void  reconnect()  {     while
(!client.connected()) {
   Serial.print("  Connecting to MQTT...");
   String clientId = "ESP8266Client-" + String(random(1000, 9999));
if (client.connect(clientId.c_str())) {
     Serial.println("  Connected to MQTT broker");
   } else {
```

```
      Serial.print("  Failed, rc=");
Serial.print(client.state());     Serial.println(" Retrying in 5
seconds...");      delay(5000);
   }
 }
}
// ========== Main Loop ==========
void loop() {
checkWiFiConnection();   if
(!client.connected()) {
reconnect();
  }
  client.loop();

  sendSensorData();
  delay(3000);  // Send data every 3 seconds
}
// ========== Publish Sensor Data ==========
void sendSensorData() {   float temperature =
dht.readTemperature();   float humidity =
dht.readHumidity();   int gasAnalogValue =
getAverageGasValue();   int gasDigitalValue =
digitalRead(MQ4_DIGITAL);

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("  Failed to read from DHT22!");     return;
  }
```

```
 Serial.println("===== Sensor Readings =====");
 Serial.print(" 🌡 Temperature: "); Serial.print(temperature); Serial.println(" °C");
  Serial.print("  Humidity: "); Serial.print(humidity); Serial.println(" %");
  Serial.print("  Gas (Analog Avg): "); Serial.println(gasAnalogValue);
  Serial.print("  Gas Detected (Digital): ");
  Serial.println(gasDigitalValue == HIGH ? "Yes" : "No");
  Serial.println("============================");

  // Publish with retain flag = true
  client.publish(mqtt_topic_temp, String(temperature).c_str(), true);
client.publish(mqtt_topic_humidity, String(humidity).c_str(), true);
client.publish(mqtt_topic_gas, String(gasAnalogValue).c_str(), true);
}
```

A.2 RAPIDMINNER EXECUTE PYTHON

```
 def rm_main(rm_input):     import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import joblib     import os

try:
      df = pd.DataFrame(rm_input)
      print("Input data verified. Columns:", df.columns.tolist())

      model = DecisionTreeClassifier()
      model.fit(df[['Day','Gas Reading']], df['Status'])

      output_dir = r"Downloads"
        os.makedirs(output_dir, exist_ok=True)
              output_path = os.path.join(output_dir, "model.pkl")

        print(f"Attempting to save to: {output_path}")
        print(f"Path exists: {os.path.exists(output_dir)}")
        print(f"Write test: {os.access(output_dir,
        os.W_OK)}")

        joblib.dump(model, output_path)
        print("  Model saved successfully!")

      return df

   except Exception as e:
      error_msg      =      f"            Failed:      {str(e)}"
print(error_msg)
      return pd.DataFrame({"Error": [error_msg]})
```

A.3 NODE-RED FUNCTION CODE

PYTHON FILE Function

```
let gas = msg.payload; msg.gas

= gas;

msg.execCommand = `python -u
"C:\\Users\\ASUS\\OneDrive\\Documents\\AltairRapidMiner\\AI
Studio\\Repositories\\Final  Year  Project\\processor\\Downloads\\predict.py"  ${gas}`;

return msg;
```

ALERT MESSAGES Function

```
let output = msg.payload.toString().trim(); let

lines = output.split("\n");


// Extract and sanitize values let status

= (lines[0] || "unknown").trim(); let gas =

(lines[1] || "unknown").trim();


if (status === "Fresh") {     return null; // Stop the

flow, no message sent

}

// Friendly status message let condition = "  Unknown condition"; if (status ===

"Overripe") condition = "  Your Fruit is about to Rot. Consume Now!";

else if (status === "Root" || status === "Rotten") condition = "  Rotten – It is not safe to
eat anymore.";
```

```
// Date and time formatting let

now = new Date();

let hour12 = now.getHours() % 12 || 12;

let minutes = now.getMinutes().toString().padStart(2, '0');

let ampm = now.getHours() >= 12 ? "PM" : "AM"; let time

= `${hour12}:${minutes} ${ampm}`;

let date = `${now.getDate().toString().padStart(2, '0')}/${(now.getMonth() + 1).toString().padStart(2, '0')}/${now.getFullYear()}`;


// Telegram message format

msg.payload = { chatId:

"5755480500", type:

"message",

 content: ` Fruit Ripeness Update\n\n Status: ${status}\n${condition}\n\n Gas Level: ${gas}\n Recorded: ${time} | ${date}`

};


return msg;
```

TELEGRAM COMMAND Function

```
if (msg.payload.content === "/status") {

return msg;

}

return null;
```

PYTHON FILE 2 Function

```
let gas = flow.get("gas"); let

temp = flow.get("temp"); let

humid = flow.get("humid");

msg.payload = "";

msg.execCommand = `python -u
"C:\\Users\\ASUS\\OneDrive\\Documents\\AltairRapidMiner\\AI
Studio\\Repositories\\Final Year Project\\processor\\Downloads\\predict2.py" ${gas}`;

return msg;
```

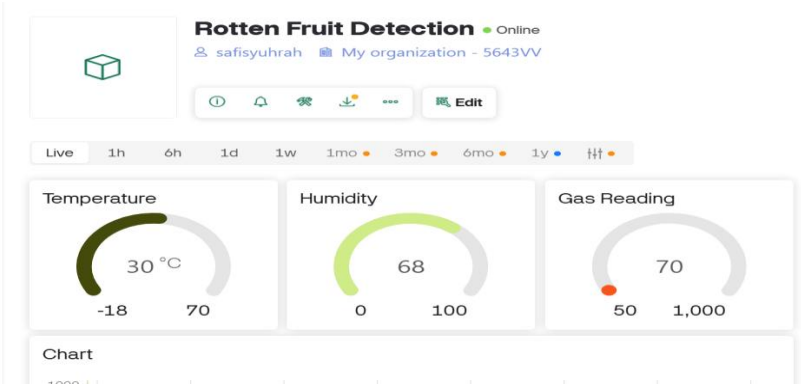
STATUS MESSAGE Function

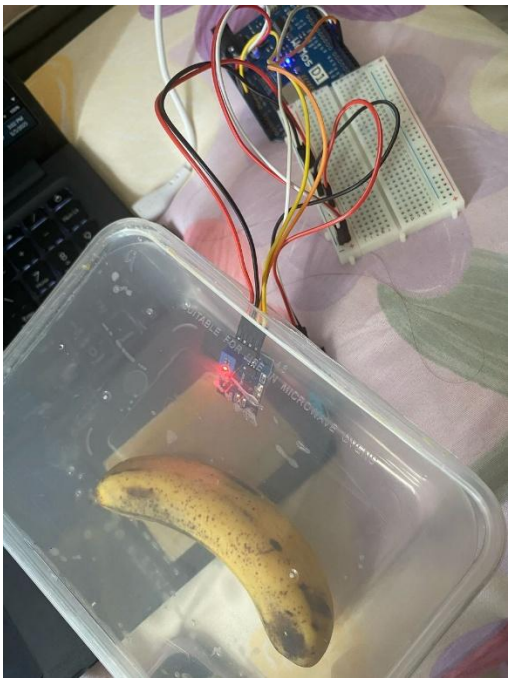
```
let prediction = msg.payload.trim().split("\n")[0];  // Only get first line

let temp = flow.get("temp"); let humid = flow.get("humid"); let gas =

flow.get("gas");

msg.payload = {

chatId: "5755480500",

type: "message",

content:

    `  Fruit Monitoring Report  \n\n` +

    `  Temperature: ${temp} °C\n` +

    `  Humidity: ${humid} %\n` +

    `  Gas Level: ${gas}\n\n` +

    `  Predicted Status: ${prediction}  `

};

return msg;
```

APPENDIX B : DATA COLLECTION

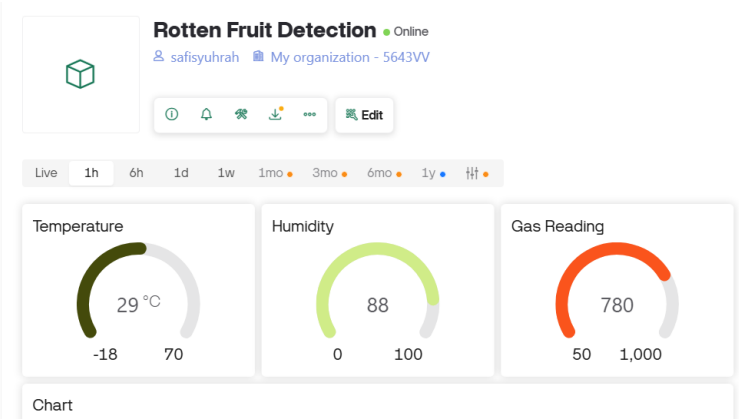
B.1 BANANA

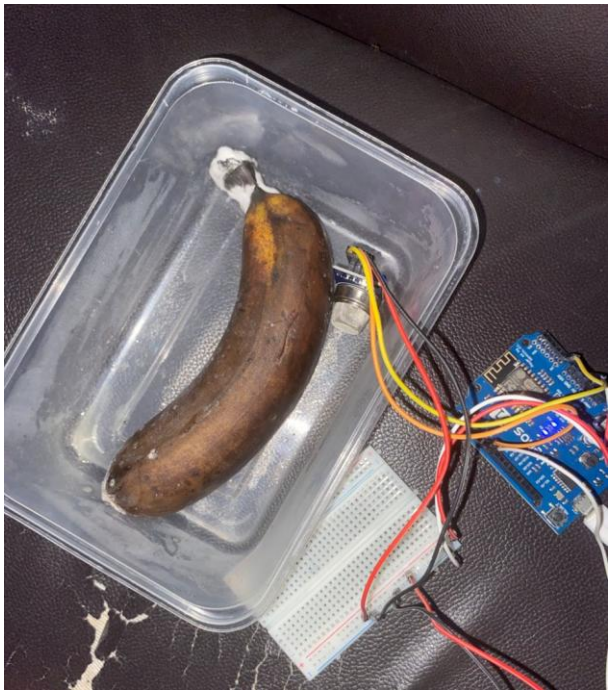
DAY 1: FRESH





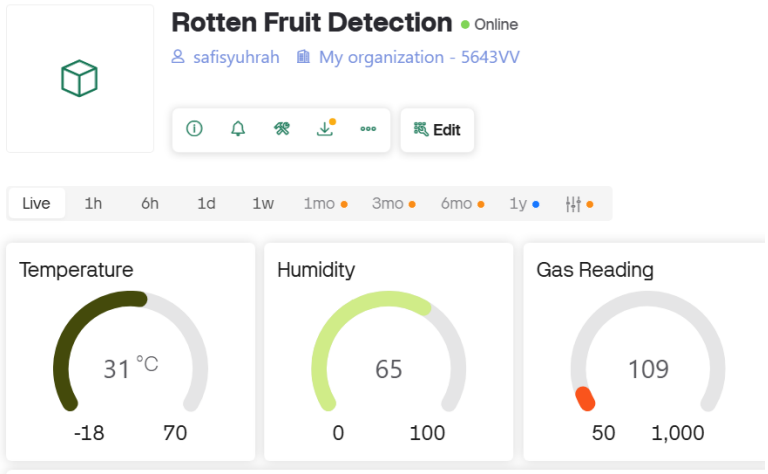
DAY 10: ROTTEN

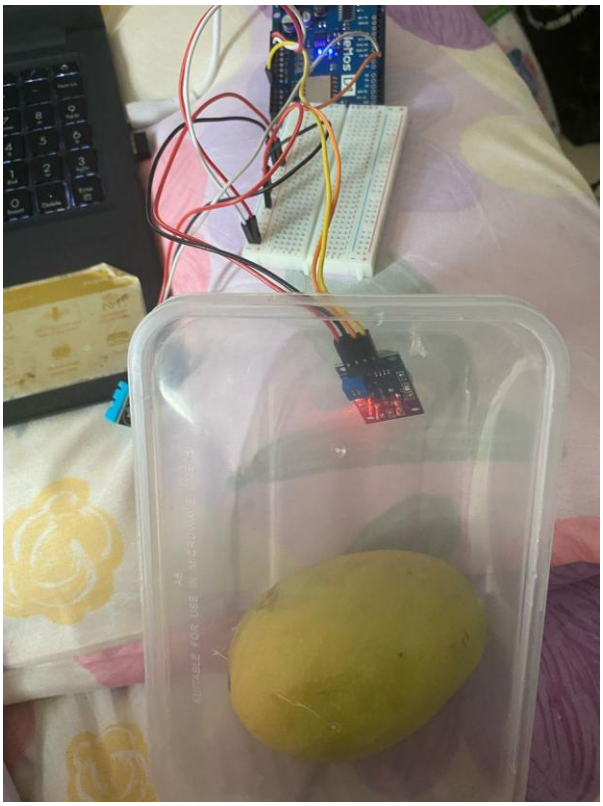




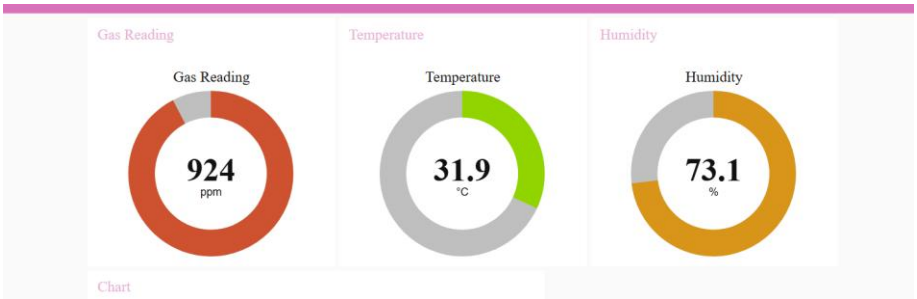
B.2 MANGO

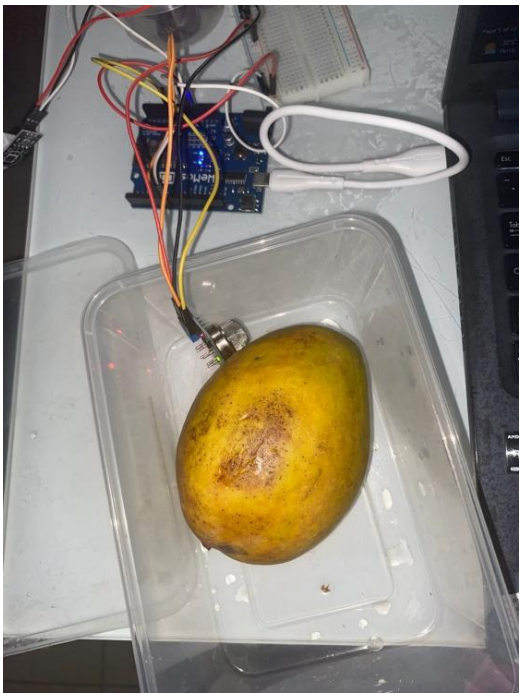
DAY 1: FRESH





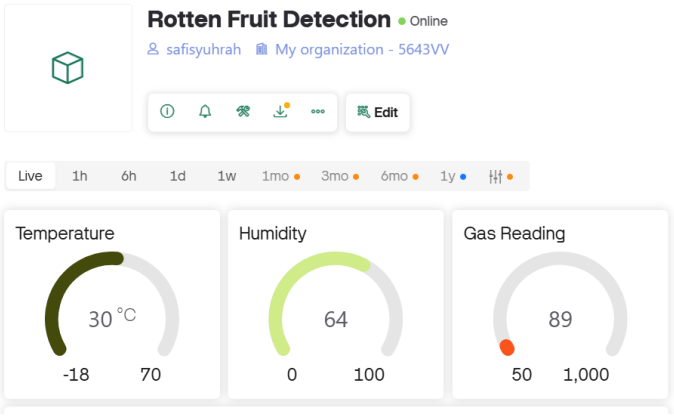
DAY 21: ROTTEN

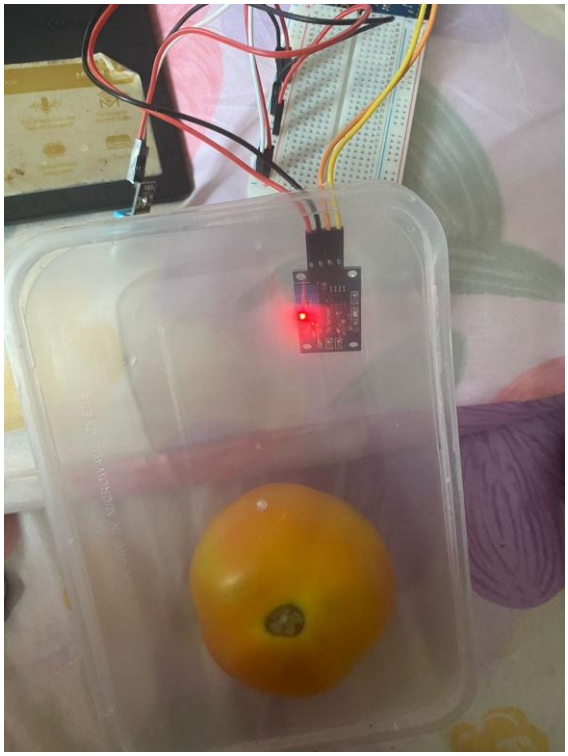




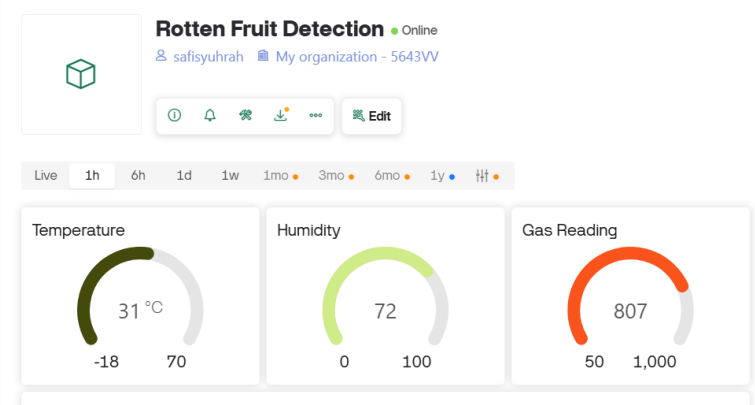
B.3 TOMATO

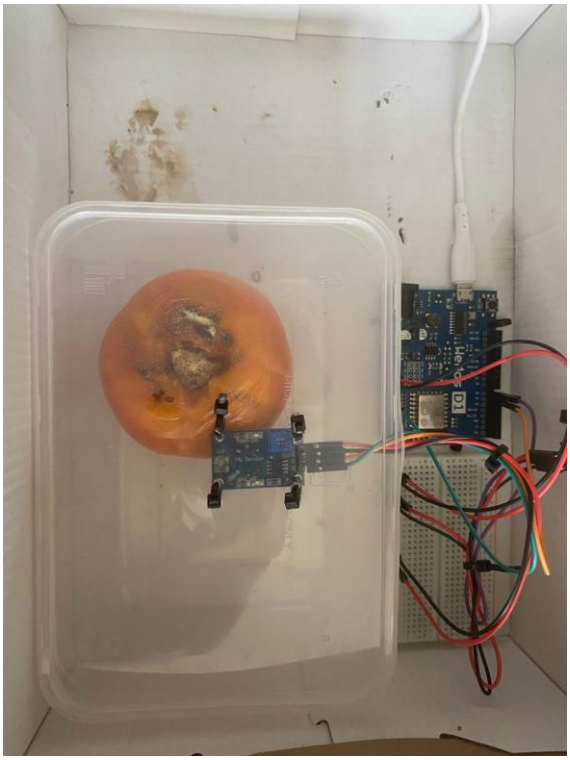
DAY 1: FRESH





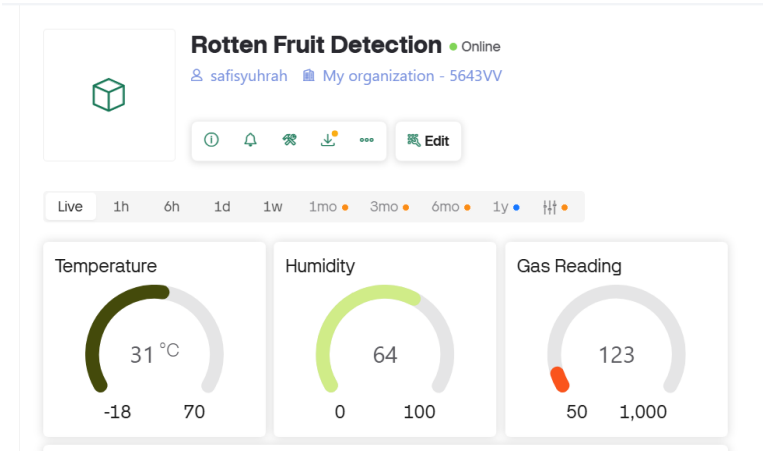
DAY 25: TOMATO

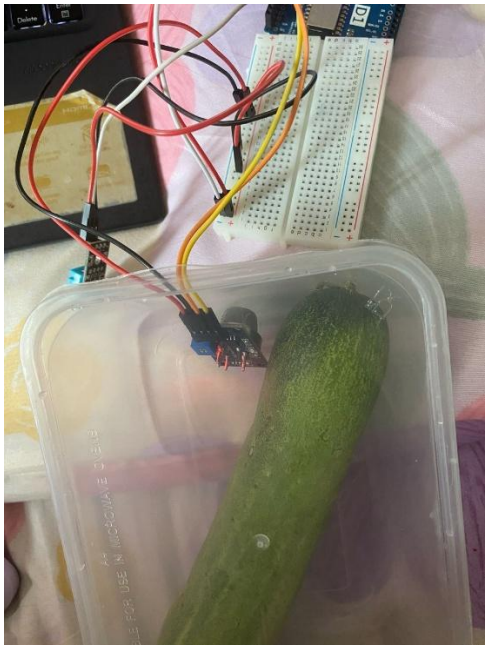




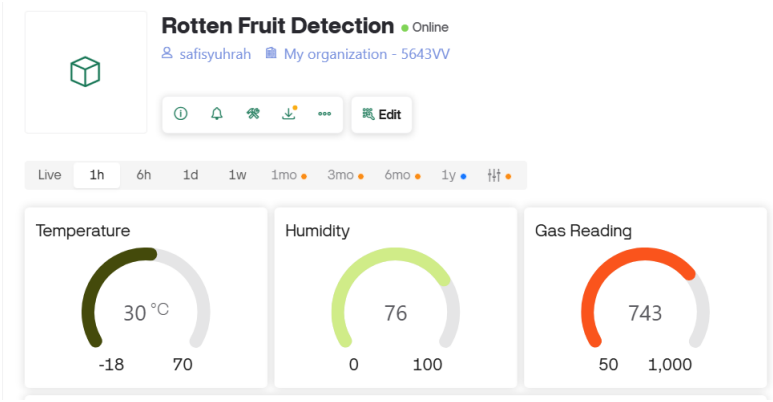
B.4 CUCUMBER

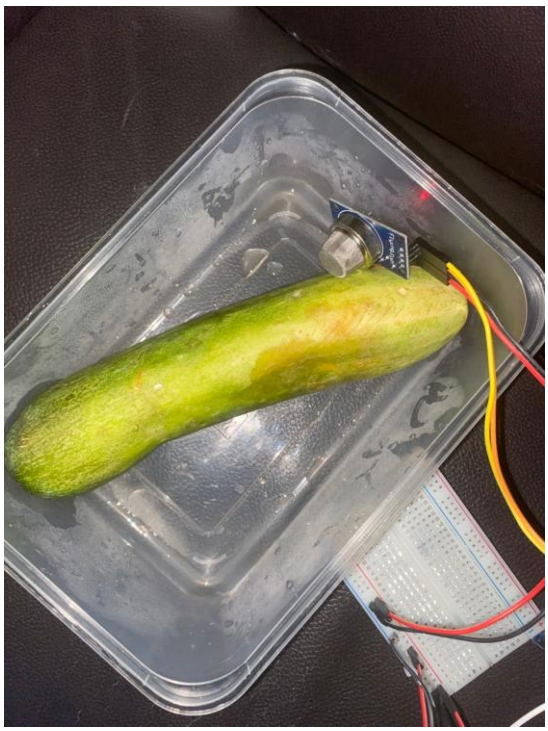
DAY 1: FRESH





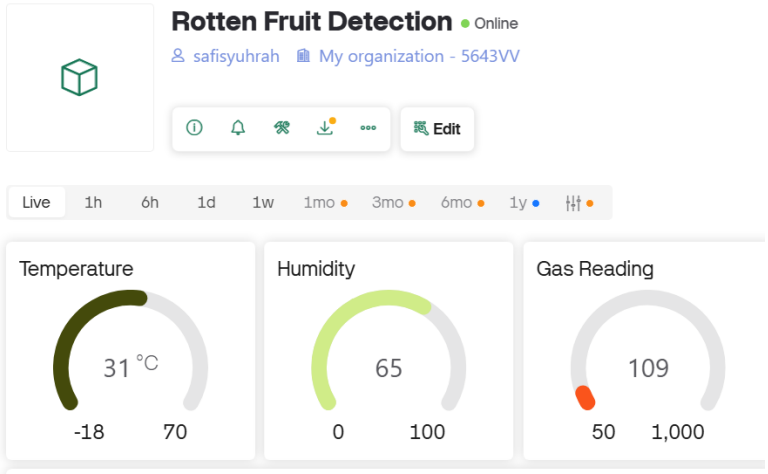
DAY 13: CUCUMBER

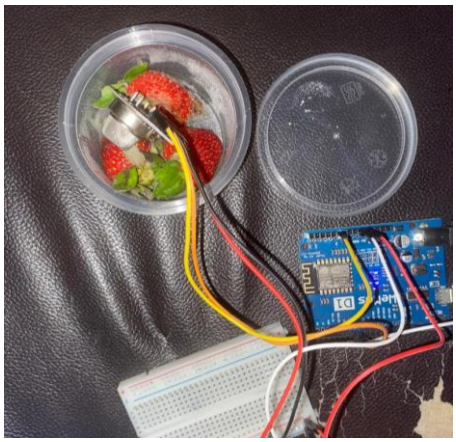




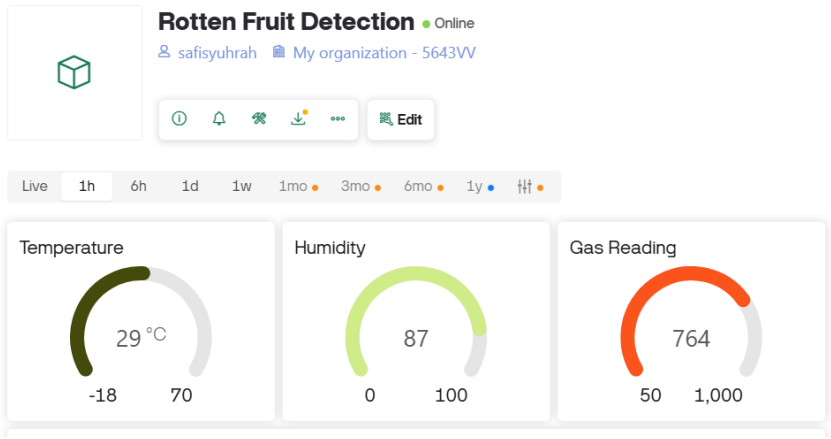
B.5 STRAWBERRY

DAY 1: FRESH



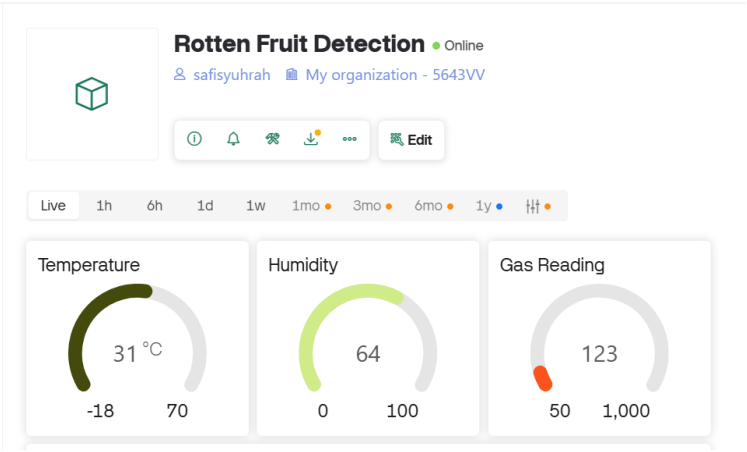


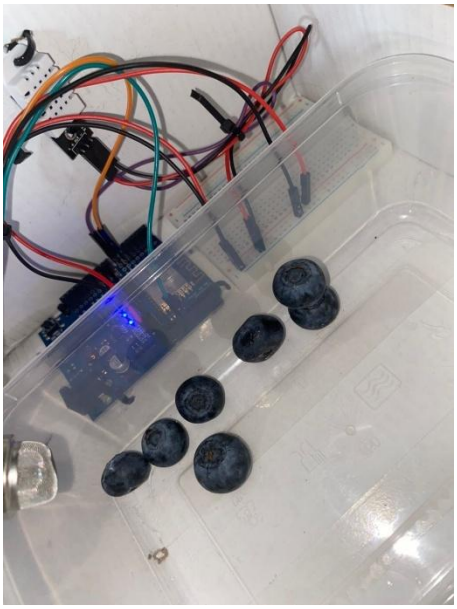
DAY 8: ROTTEN


Rotten Fruit Detection
Online
safisyuhrah
My organization - 5643VV
Edit
Live
1h
6h
1d
1w
1mo
3mo
6mo
1y
Temperature
29 °C
-18
70
Humidity
87
0
100
Gas Reading
764
50
1,000

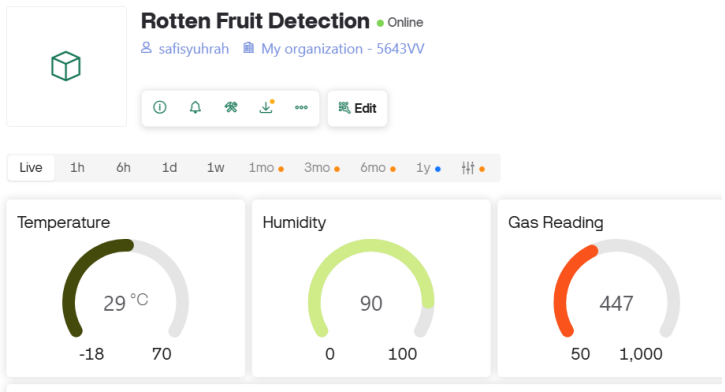
B.6 BLUEBERRY

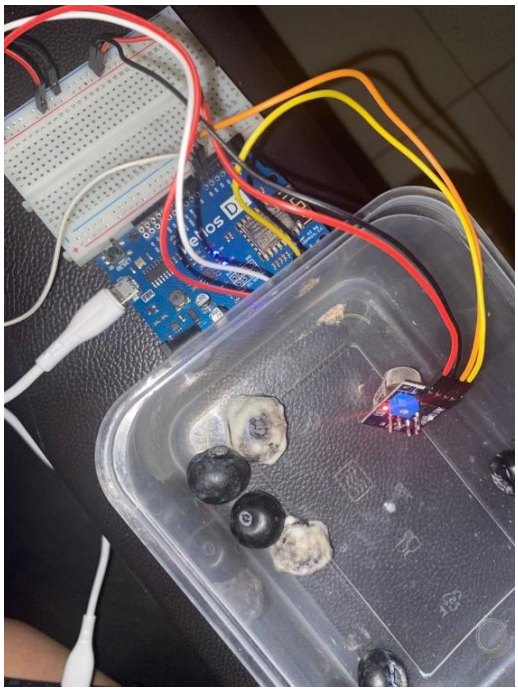
DAY 1: FRESH





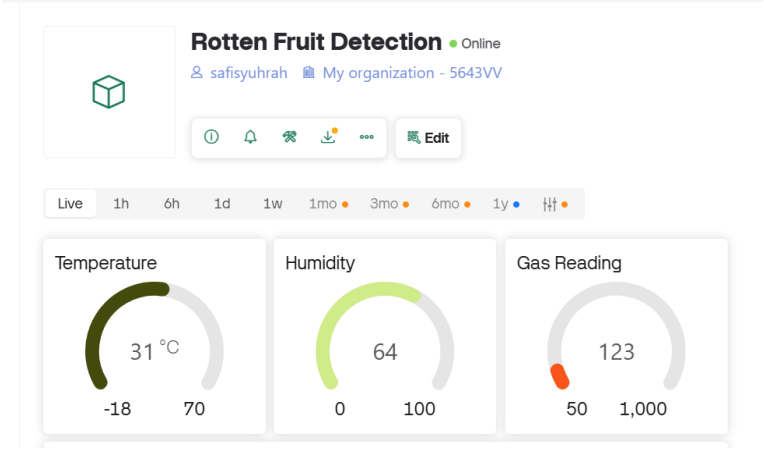
DAY 11: ROTTEN



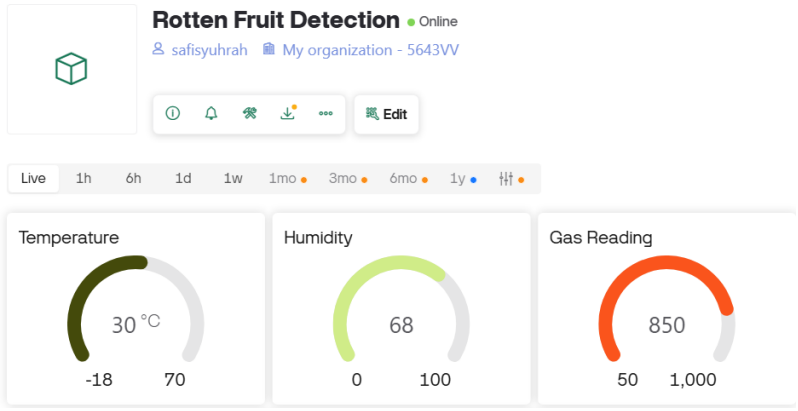


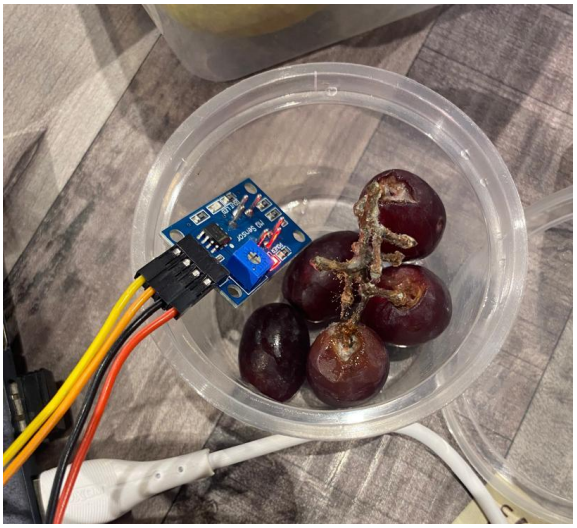
B.7 GRAPES

DAY 1: FRESH



DAY 14: ROTTEN





B.8 COLLECTED DATA

Fruits	Gas Reading	Day	Status
Strawberry	106.5	1	Fresh
Strawberry	136	2	Fresh
Strawberry	177.5	3	Fresh
Strawberry	214	4	Fresh
Strawberry	255.5	5	Overripe
Strawberry	311	6	Overripe
Strawberry	481	7	Overripe
Strawberry	755.5	8	Root
Blueberry	118.5	1	Fresh
Blueberry	123	2	Fresh
Blueberry	128.5	3	Fresh
Blueberry	131.5	4	Fresh
Blueberry	137	5	Fresh
Blueberry	161.5	6	Fresh
Blueberry	183	7	Fresh
Blueberry	221	8	Fresh
Blueberry	255	9	Overripe
Blueberry	329	10	Overripe
Blueberry	443.5	11	Root
Banana	74.5	1	Fresh
Banana	86.5	2	Fresh
Banana	110	3	Fresh
Banana	129.5	4	Fresh
Banana	162	5	Fresh
Banana	190.5	6	Fresh
Banana	263	7	Overripe
Banana	296.5	8	Overripe
Banana	512.5	9	Overripe
Banana	762.5	10	Root
Cucumber	87	1	Fresh
Cucumber	108	2	Fresh
Cucumber	118	3	Fresh
Cucumber	131	4	Fresh
Cucumber	148.5	5	Fresh
Cucumber	162.5	6	Fresh
Cucumber	189	7	Fresh
Cucumber	225.5	8	Fresh
Cucumber	264	9	Overripe
Cucumber	313.5	10	Overripe
Cucumber	445.5	11	Overripe
Cucumber	502	12	Overripe
Cucumber	767	13	Root

Grapes	129	1	Fresh
Grapes	139	2	Fresh
Grapes	144.5	3	Fresh
Grapes	162.5	4	Fresh
Grapes	172	5	Fresh
Grapes	213	6	Fresh
Grapes	229	7	Fresh
Grapes	243.5	8	Fresh
Grapes	311.5	9	Fresh
Grapes	352	10	Overripe
Grapes	413	11	Overripe
Grapes	575	12	Overripe
Grapes	669	13	Overripe
Grapes	841.5	14	Root
Mango	102	1	Fresh
Mango	104.5	2	Fresh
Mango	110	3	Fresh
Mango	122	4	Fresh
Mango	131.5	5	Fresh
Mango	141	6	Fresh
Mango	159.5	7	Fresh
Mango	175.5	8	Fresh
Mango	179.5	9	Fresh
Mango	193	10	Fresh
Mango	199	11	Fresh
Mango	213	12	Fresh
Mango	276	13	Fresh
Mango	332	14	Overripe
Mango	370.5	15	Overripe
Mango	437	16	Overripe
Mango	494	17	Overripe
Mango	567	18	Overripe
Mango	622	19	Overripe
Mango	827	20	Overripe
Mango	1024	21	Root
Tomato	1024	1	Fresh
Tomato	80	2	Fresh
Tomato	97	3	Fresh
Tomato	100.5	4	Fresh
Tomato	117.5	5	Fresh
Tomato	121	6	Fresh
Tomato	128	7	Fresh
Tomato	132.5	8	Fresh
Tomato	148	9	Fresh
Tomato	151.5	10	Fresh
Tomato	154	11	Fresh

Tomato	158.5	12	Fresh
Tomato	163	13	Fresh
Tomato	176	14	Fresh
Tomato	191	15	Fresh
Tomato	231.5	16	Fresh
Tomato	301	17	Fresh
Tomato	321	18	Fresh
Tomato	345.5	19	Fresh
Tomato	388	20	Overripe
Tomato	409	21	Overripe
Tomato	456.5	22	Overripe
Tomato	491	23	Overripe
Tomato	508	24	Overripe
Tomato	992	25	Root

APPENDIX C : GANTT CHART

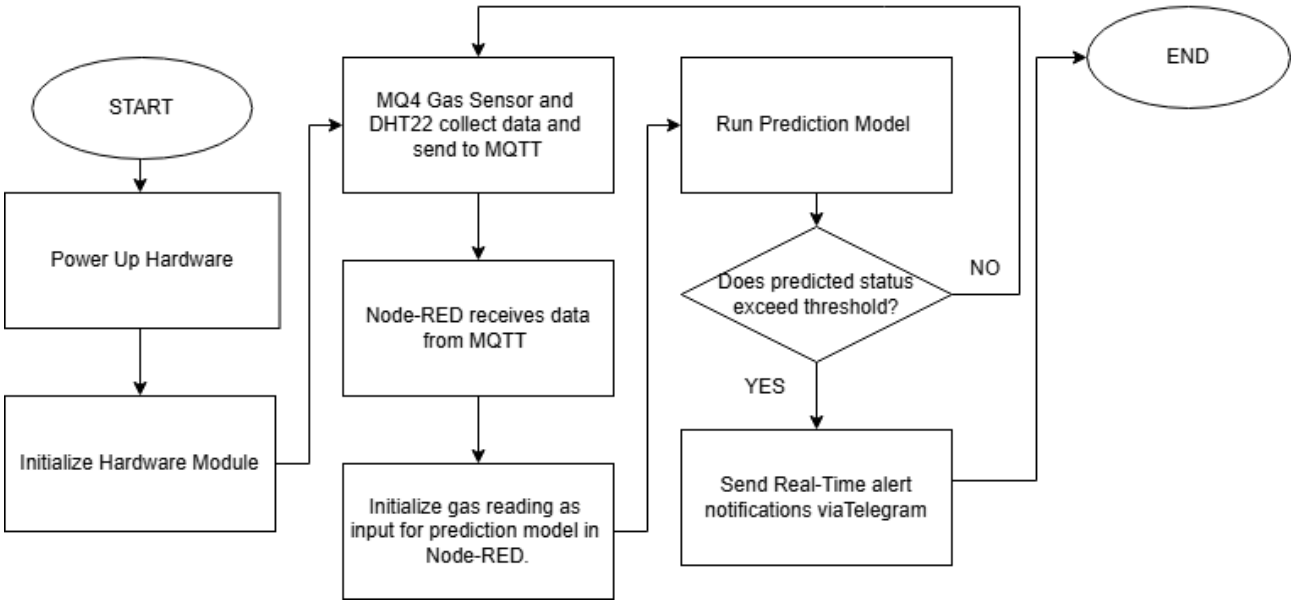
GANTT CHART

PROJECT TITLE: ROTTEN FRUIT DETECTION USING IOT — **16/10/2024** (START DATE) — **ONGOING** (END DATE)

NUR SAFI SYUHRAH BINTI ABDUL RANI

No	Task Name	Start Date	End Date	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16	WEEK 17	WEEK 18	WEEK 19	WEEK 20	WEEK 21	WEEK 22	WEEK 23	WEEK 24	WEEK 25	WEEK 26	WEEK 27	WEEK 28
1	Find Supervisor	16/10/2024	21/10/2024																											
2	Title Confirmation	21/10/2024	28/10/2024																											
3	Research on Title	25/10/2024	15/11/2025																											
4	Proposal Report	1/11/2024	6/11/2024																											
5	Report Chapter 1	13/11/2024	26/11/2024																											
6	Report Chapter 2	8/12/2024	24/12/2024																											
7	Report Chapter 3	30/12/2024	5/11/2025																											
8	Presentation FYP1	2/1/2025	15/1/2025																											
9	Submission of Report FYP1	15/1/2025	20/1/2025																											
10	Building the system	1/2/2025	1/5/2025																											
11	Troubleshooting	1/5/2025	22/5/2025																											
12	Testing Prototype	22/5/2025	1/6/2025																											
13	Report Chapter 4	31/5/2025	1/6/2025																											
14	Report Chapter 5	2/6/2025	2/6/2025																											
15	Presentation FYP2	3/6/2025	9/6/2025																											
16	Submission of Thesis	9/6/2025	13/6/2025																											
17	Submission of Report FYP2	9/6/2025	13/6/2025																											

APPENDIX D : FLOWCHART



APPENDIX E : POSTER

UNIVERSITI KUALA LUMPUR
MALAYSIAN INSTITUTE OF INFORMATION TECHNOLOGY
BACHELOR OF INFORMATION TECHNOLOGY (HONS) IN INTERNET OF THINGS
FINAL YEAR PROJECT 2
ROTTEN FRUIT DETETCTION USING IOT

NUR SAFI SYUHRAH BINTI ABDUL RANI 52224122591
ASSOC. PROF. TS. DR. HAIDAWATI BINTI MOHAMAD NASIR

INTRODUCTION

Spoiled fruits often go unnoticed until it's too late, leading to unnecessary food waste. Rotten Fruit Detection System uses IoT technology and automation to monitor fruit freshness in real time. By combining the MQ4 gas sensor and DHT22 sensor, the system detects early signs of spoilage based on gas emissions, temperature, and humidity levels. The Wemos D1 ESP8266 microcontroller collects this data and sends it to the Node-RED platform via MQTT. A built-in prediction model analyzes the readings to determine the fruit's ripeness status. If spoilage is detected, the system instantly sends a Telegram alert to notify users. This smart system helps reduce food waste, improves fruit storage management, and supports sustainable household practices.

PROBLEM STATEMENT

Many households face the common problem of forgetting fruits stored in the fridge, often leading to spoilage and unnecessary food waste. Without a reliable system to monitor their freshness, users are unaware of early signs of spoilage, resulting in fruits being discarded once they are completely rotten. This project addresses this issue by providing an IoT-based system that detects early indicators of fruit spoilage, such as gas emissions based on predictions model and promptly notifies users. By enabling timely action, this system helps reduce food waste, ensures better produce management, and promotes sustainability in daily household practices.

OBJECTIVES

- · To use MQ4 gas sensors to get gas reading for rotten fruits by setting baseline readings in a fresh, non-spoiling environment.
- · To develop an IoT-based system using sensors to continuously monitor the condition of fruits and detect early signs of spoilage, such as changes in gas emissions or other environmental factors.
- · To create an alert system that notifies users when fruits are at risk of rotting, based on the data gathered from IoT sensors.

SCOPE AND LIMITATION

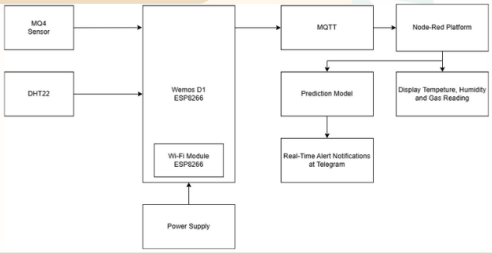
The scope of this project is to develop an IoT-based fruit spoilage detection system specifically designed for household users, including those who may not have the time or ability to regularly check their stored produce. The system will monitor environmental factors such as gas emissions, temperature, and humidity using sensors and provide real-time alerts through a user-friendly interface. By automating the detection process, the system ensures that users can easily manage their fruit storage without needing technical knowledge or constant attention, making it accessible for all, including busy individuals or those with physical or cognitive limitations. This solution aims to reduce food waste, improve convenience, and promote sustainable practices in everyday household routines.

METHODOLOGY

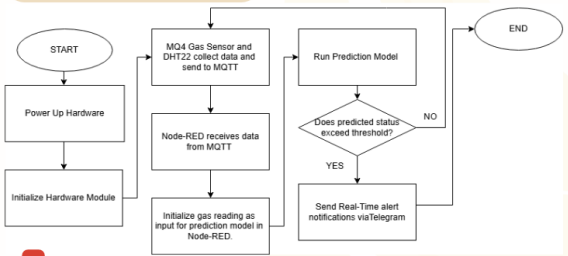
LITERATURE REVIEW

Title	Author(s) and Year	Technology Used	Methodology	Sensor, Microcontroller and Cloud
IOT Based Food Freshness Detection Using Deep Learning Techniques	Karthickeyan P. 2022	IoT, Deep Learning Algorithms Rotational Mechanism	Agile	-Raspberry Pi 4 Model B - ESP8266 - Web Camera - MQ2 Sensor - MQ135 Sensor
IoT based Fruit Quality Inspection and Lifespan Detection System	Ashim Saha, 2024	IoT, Machine Learning, Data Visulization	Waterfall	- Raspberry Pi Pico - NodeMCU ESP8266 - MQ4 Sensor - DHT11 -ThingSpeak Cloud
IoT-Based Food Spoilage Detection System with UV Sterilization	P. Manjulamna, 2024	IoT, UV Sterilization, Mobile Application	Agile	- NodeMCU ESP8266 - Gas Sensor - DHT11 - IR Sensor
Fruit Freshness Detection and Monitoring using IOT	Prof. Dipti A. 2024	IoT, Data Communication	Waterfall	- ESP8266 - Arduino UNO - MQ3 Sensor - Blynk
Developing an IoT and ML-driven Platform for Fruit Ripeness Evaluation and Spoilage Detection: A Case Study on Bananas	Rajini M, V Persis 2024	IoT, Machine Learning, Cloud Storage, Data Transmission	Waterfall	- ESP32 - SGP30 - SHT40 - ThingSpeak Cloud

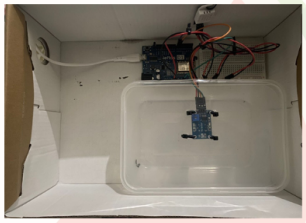
BLOCK DIAGRAM

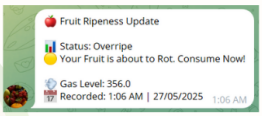


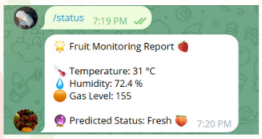
FLOWCHART

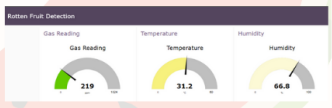


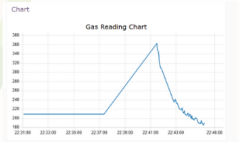
RESULT AND DISCUSSION











CONCLUSION

The Rotten Fruit Detection System offers a smart solution for monitoring fruit condition using real-time sensor data and automated alerts. By providing early warnings of spoilage, the system supports better food management and encourages more sustainable habits at home.

REFERENCES

- [KarthickeyanP, B., Devi, K., & Professor, A. (2022). IOT Based Food Freshness Detection Using Deep Learning Techniques.
- Zakaria, A., Rahman, A., & Rahim, A. (2014). AN OVERVIEW OF FRUIT SUPPLY CHAIN IN MALAYSIA (Vol. 37).