ABSTRACT

Fruits, meat and other dairy products are perishable and easily spoil during transportation and handling and in many cases, can result in unnecessary quantities of food wastage, loss of economic gains and even health hazards. The current monitoring measures have been mainly based on manual checks or the use of IoT threshold-based sensors that respond when the spoilage has already been noticed. These are not predictable and fail to diagnose the pre-signal warnings. To address the challenge, current project proposes IoT-based predictive quality monitoring system, which is a combination of real-time environmental sensing and machine learning models to make decisions on freshness and prediction of remaining shelf life.

The system uses ESP32 microcontroller that is linked to several sensors: DHT22 (temperature and humidity), MQ-135 (volatile organic compounds and carbon dioxide), MQ-137 (ammonia), and MQ-3 (ethanol). The data gathered is always pushed into cloud systems like ThingSpeak where it is visualized and Render to serve the API and project the data to Azure Machine Learning endpoints. Azure applies available analysis to determine the classification of the stages of spoilage and predictable left usable time by running trained Random Forest and regression models in real-time. Telegram and Twilio by a dual alert system will provide both threshold-based and predictive alerts to enable the user to immediately make preventive action.

Under controlled storage conditions, experimental testing on the foods of various categories was done. A good indicator of system efficiency is that the predictive model was able to identify the signs of spoilage several hours before the food appeared to have rotted. The proposed integrated IoT-ML framework offers a cost-effective, scalable, and realistic approach to smart food quality monitoring and cold-chain intelligent control and can result in the reduction of waste and the enhancement of food safety.

CHAPTER 1

INTRODUCTION

People everywhere want fresh, safe food, and honestly, that's put a lot of pressure on the food industry to keep quality in check. Fruits, dairy, meat are the usual food items that spoils fast, especially when storage and transport conditions keep changing. Old-school inspection methods or simple IoT setups just don't catch problems early enough. That means more food goes to waste, and sometimes, it's even a health risk. This project tackles that head-on by teaming up machine learning with IoT sensors, so we can actually predict food quality and estimate shelf-life in real time, way before things go bad.

1.1 OVERVIEW

Spoilage is still a major issue throughout the food supply chain, especially for foods that must be stored under strict temperature and humidity conditions, like fruits, meats, and dairy. Minor changes in storage conditions can speed up spoilage, leading to significant loss and safety concerns. Traditionally, quality checks rely on manual testing or basic IoT setups that only provide warnings after spoilage has started. These methods cannot catch early warning signs that could prevent waste.

In this project, we created an IoT-based monitoring system to track environmental factors in real time and use machine learning to predict potential spoilage before it happens. The hardware includes an ESP32 microcontroller and sensors like DHT22 for humidity and temperature, MQ-135 for CO_2 and volatile organic compounds, MQ-137 for ammonia, and MQ-3 for ethanol. The sensors constantly send data, which goes to cloud platforms like ThingSpeak for visualization and Azure Machine Learning for analysis and prediction.

The system generates two types of alerts; one based on threshold values and the other from a trained predictive model that uses Random Forest algorithms. This two-fold approach ensures quick and early detection. Through this method, the project aims to reduce food waste, improve storage options, and encourage safer, smarter food handling practices.

1.2 OBJECTIVES

This project aims to create a real-time IoT system that spots early signs of spoilage in perishable foods and predicts how much shelf life they have left. By bringing together sensor data and machine learning, the system doesn't just react to problems after they happen but it helps you catch issues early and act before they get worse.

The specific objectives of this work are:

- To develop an end-to-end IoT system using the ESP32 microcontroller and sensors for real-time monitoring of perishable food items.
- To capture and analyze spoilage indicators such as Volatile Organic Compounds (VOC), Ammonia (NH_3), Carbon Dioxide (CO_2), Ethanol, Temperature, and Humidity.
- To design and train a machine learning model capable of predicting spoilage progression and estimating the remaining shelf life accurately.
- To build a cloud-connected dashboard for continuous visualization and tracking of live data.
- To demonstrate the complete system performance under controlled storage conditions for selected food categories such as fruits, meat, and dairy.

This project ultimately aims to provide a reliable and efficient solution for early spoilage detection, thereby reducing food waste, improving quality control, and ensuring safer handling of perishable products throughout the supply chain.

CHAPTER 2

LITERATURE SURVEY

1. **TITLE: IoT-Based Meat Freshness Classification Using Deep Learning [IEEE, 2024]**

   AUTHORS: Zarif Wasif Bhuiyan, Syed Ali Redwanul Haider, Adiba Haque, Mohammad Rejwan Uddin, and Mahady Hasan.

   SUMMARY: This article reports on an IoT-based system that identifies meat freshness based on gas sensors and deep learning. The system utilized an ESP32-CAM with MQ135, MQ136, and MQ4 sensors interfaced to a Raspberry Pi 3 to collect data. The performance of a custom CNN model to classify beef and mutton as fresh, semi-fresh, or spoiled was 99% accurate. This outperformed other models that were previously done. The system gave output in real-time using LEDs and a display interface and hence has practical application in real-time scenarios.

   RESEARCH GAPS: It is confined to classification of meat and is devoid of predictive functionality and cloud-based analysis. It is only concerned with detection and not with prediction of early spoilage or monitoring of multi-category food. These are the areas this project seeks to address using IoT and integration with machine learning.

2. **TITLE: Low-Cost Real-Time Non-Invasive Milk Quality Monitoring Using 60 GHz FMCW Radars [IEEE, 2025]**

   AUTHORS: Jessica Chong, Ala Eldin Omer, Stefan Idziak, Lan Wei, and George Shaker

   SUMMARY: The contribution presented here illustrates a low-cost, non-invasive radar-based approach to real-time milk quality monitoring using 60 GHz Frequency Modulated Continuous Wave radars. The proposed system utilizes milk's dielectric properties for fat content measurement and deviation from reference standards. Milk cartons positioned on a conveyor belt were interrogated by radar sensors, and the samples were classified by a Support Vector

Machine classifier according to their fat percentage. The system demonstrated its potential applications in industrial dairy quality monitoring with a relatively low setup and cost by achieving an accuracy of 100% in the case of stationary cartons and 87.5% in the case of moving cartons.

RESEARCH GAPS: The work does not incorporate predictive analytics or Internet of Things connectivity, rather using radar sensing to identify variations in milk fat. It also does not cover cloud-based real-time visualization, early spoilage prediction, and multi-category food analysis, all of which are focused on in this proposed project using IoT–ML integration for cross-category predictive quality monitoring.

3. **TITLE: An IoT-Based Application for Real-Time Detection of Rotten Fruits [ResearchGate, 2024]**

AUTHORS: Jeberson Retna Raj, Senduru Srinivasulu, Jabez, and Gowri Shanmugam

SUMMARY: This paper introduces an IoT-based system for detecting the freshness of fruits in real time using sensors and cloud integration. The setup uses an Arduino Uno, a DHT11 for measuring temperature and humidity, an MQ2 for detecting gas or smoke, and an MQ3 for alcohol detection. Data from these sensors is sent to the ThingSpeak IoT analytics platform for visualization and monitoring. The system successfully identifies spoiled fruits, such as bananas, by examining changes in environmental factors and showing results through LEDs. The authors note that this method can help consumers and food suppliers find inedible items before distribution.

RESEARCH GAPS: While the system focuses on detecting fruit freshness, it does not include predictive analysis or machine learning. It offers real-time classification but cannot estimate shelf life or predict spoilage trends. This work aims to improve on that by adding predictive machine learning models and combining data from multiple sensors to forecast

spoilage across various perishable items.

4. **TITLE: An RFID-Powered Multisensing Fusion Industrial IoT System for Food Quality Assessment and Sensing [IEEE, 2024]**

   AUTHORS: Chenyang Song, Zhipeng Wu, John Gray, and Zhaozong Meng

   SUMMARY: The paper describes a holistic IIoT framework for real-time assessment of food quality using RFID-powered multisensory fusion. This system integrates RFID-enabled sensors that are capable of monitoring parameters at both the environmental and product levels, such as temperature, humidity, and the concentration of certain types of gases. The authors proposed a five-layer architecture, including layers for sensing, communication, control, and data analytics. Machine learning models such as regression and RNN were employed to predict food shelf life and food quality in a demonstration on a smart shelf. With the proposed method, high accuracy could be achieved for the freshness estimation of the meat products with minimal manual testing and complete traceability along with the supply chain.

   RESEARCH GAPS: Although the system introduces predictive shelf-life modeling, it focuses primarily on RFID-based industrial applications. The framework design is missing the lightweight and low-cost IoT implementations suitable for small-scale monitoring and category-wise spoilage forecasting. This project proposes a sensor-based IoT system integrated with machine learning for multi-category predictive quality monitoring through cloud analytics and real-time alerts.

5. **TITLE: Integration of Electronic Nose and Machine Learning for Monitoring Food Spoilage in Storage Systems [International Journal of Online and Biomedical Engineering (iJOE), 2024]**

   AUTHORS: Shakhmaran Seilov, Dias Abildinov, Marat Baydeldinov, Akniyet Nurzhaubayev,

Bibinur Zhursinbek, and Xiao Guang Yue

SUMMARY: It describes a novel approach using an integrated e-nose sensing system with machine-learning algorithms in monitoring food spoilage, in particular for potato storage systems. The e-nose detects the volatile organic compounds emanating at fresh and different stages of spoilage and feeds into neural network models generated through MATLAB. This system classifies potatoes as fresh, mildly spoiled, or fully spoiled with high accuracy. This research leverages the power of multi-sensory gas data with intelligent pattern recognition to illustrate an effective and trustworthy solution for post-harvest quality monitoring, reduction of food waste.

RESEARCH GAPS: Although the system works well in detecting spoilage of potatoes, it does not have IoT connectivity or cloud-based predictive capability. The proposed project extends this with IoT-driven multisensory data acquisition integrated with machine learning-based spoilage forecasting for a wide range of perishable food items.

6. **TITLE:** ***Advancing Food Safety Through IoT: Real-Time Monitoring and Control Systems*** **[International Medical Science Research Journal, 2024]**

AUTHORS: Temilade Abass, Michael Alurame Eruaga, Esther Oleiye Itua, and James Tabat Bature.

SUMMARY: It includes an IoT-enabled system for the real-time observation and management of food safety parameters across the value chain. The proposed system employs a network of sensors and IoT devices to monitor critical factors such as temperature, humidity, and pH from the production and processing stages to distribution. The combination of IoT with data analytics and machine learning in this framework allows for the early identification of contamination threats, thus supporting proactive intervention strategies. Additionally, the significance of blockchain in enhancing traceability and transparency in food

safety management has been highlighted. This enhances compliance with relevant standards while minimizing human errors associated with traditional monitoring systems

RESEARCH GAPS: The framework indeed provides strong real-time monitoring and traceability, but it does not focus on predictive quality estimation or shelf-life forecasting. In the proposed project, the idea is extended to integrate IoT sensing with machine learning for predicting spoilage trends in an effort to improve food quality management across all perishable categories.

7. **TITLE: Artificial Intelligence for Food Safety: From Predictive Models to Real-World Safeguards [Elsevier, 2025]**

AUTHORS: P. Balakrishnan, A. Anny Leema, N. Jothiaruna, Purshottam J. Assudani, K. Sankar, M. B. Kulkarni, and Manish Bhaiyya

SUMMARY: The authors studied the role of Artificial Intelligence and Machine Learning in the transformation of food safety based on image analysis and electronic sensors. It presents various models, including CNN, SVM, and ensemble techniques, which have been applied to different types of food detection for spoilage and contamination. The study showcased the potential of AI in enhancing accuracy while reducing human involvement, with quick and safe food testing compared to traditional methods. However, the paper mentions limited data availability and scalability issues for direct supervision of the systems.

RESEARCH GAPS: This paper outlines key AI systems; however, it has no real-time IoT framework that could support cost-effective and predictive food management. The proposed project adds to that by incorporating IoT sensors and ML algorithms into the immediate prediction of spoilage and shelf-life evaluation in perishable goods.

8. **TITLE: Machine Learning–Based Optimal Temperature Management for Perishable Food Supply Chain [Nature – Scientific Reports, 2024]**

AUTHORS: A. Rehman, J. Wang, L. Zhang, and S. Li

SUMMARY: This study proposes a machine-learning-driven framework to optimize temperature control throughout the perishable food supply chain. The model uses real-time data from IoT temperature sensors in making predictions of optimal storage conditions with minimal spoilage and energy consumption. Several algorithms, like Random Forest and Gradient Boosting, were trained on time-series data to recommend the ideal temperature ranges for different storage environments. The system achieved notable efficiency improvements compared to traditional threshold-based control methods, showing how ML can balance food freshness and energy sustainability in logistics operations.

RESEARCH GAPS: While improving temperature optimization within the system, it does not forecast the development of spoilage or shelf life using data obtained from sensors. The proposed project extends this by developing an IoT–ML framework that will not only monitor environmental conditions but also forecast freshness and spoilage trends for multiple perishable categories.

9. **TITLE: Optimization of the Traceability of Perishable Products Through Light Blockchain and IoT in the Food Industry [IEEE Access, 2025]**

AUTHORS: William Villegas-Ch, Jaime Govea, Pablo Santiago Moncayo-Moncayo, Alexandra Maldonado Navarro, and Carlos Chavez-Pirca

SUMMARY: This paper presents an IoT-based architecture integrated with a lightweight blockchain network to enhance the traceability of perishable food products. The system is based on the PoA consensus algorithm and leverages energy-efficient communication protocols, including MQTT and LoRa. Key parameters such as temperature, humidity, and

location are recorded during transportation and storage. The system showed an average transaction response time of 105 ms with 9.6 mJ energy consumption per transaction, proving that blockchain can offer secure and transparent tracking of food logistics. This framework offers reliable data sharing among supply-chain stakeholders while maintaining low operational cost and high scalability.

RESEARCH GAPS: While this study provides secure traceability, it lacks multi-sensor gas analysis and predictive machine-learning models for quality estimation. Based on this approach, IoT-based gas sensing will be integrated with ML-driven shelf-life prediction in the proposed project to allow proactive freshness monitoring instead of tracking the movement of products only.

10. **TITLE:** ***Critical Data Detection for Dynamically Adjustable Product Quality in IIoT-Enabled Manufacturing*** **[IEEE Access, 2023]**

AUTHORS: Sachin K. Sen, Gour C. Karmakar, and Shaoning Pang

SUMMARY: The work presented in this paper addresses the enhancement of product quality in the wine manufacturing process using an IIoT framework combined with machine-learning models. The identification of the most influential sensor parameters related to product quality during production is proposed. Data from wine fermentation sensors are analyzed using machine-learning models, such as Support Vector Machine and Random Forest algorithms, to predict product quality dynamically. This framework identifies critical data features that directly influence the end product by the use of correlation, sensitivity, and percentage-change analyses. A prediction error of about 10.4% was realized using the presented results, which proves the efficiency of the proposed model for maintaining product quality consistent through real-time adjustments.

RESEARCH GAPS: Although this study has shown the potential of IoT and ML in maintaining product consistency, it is confined to industrial wine production and did not predict freshness

in real-time for perishable foods. This proposed project extends the concept of IoT-based multi-sensor data acquisition along with machine-learning models to predict stages of spoilage and shelf life for several categories of food.

CHAPTER 3

PROBLEM DEFINITION & METHODOLOGY

3.1 PROBLEM DEFINITION

Foods such as fruits, meat, and dairy products spoil quickly when their storage or transportation conditions are inappropriate. It is highly challenging to detect any spoilage of food well in advance before one is able to visually inspect the food items. This area is important in food safety and reductions in food losses. Conventional laboratory testing is highly reliable but time-consuming, expensive, and not compatible with continuous real-time monitoring.

The majority of food-monitoring systems, based on IoT, continue using fixed threshold values or rule-based alerts and operate only when the spoilage has already begun. In turn, these are reactive mechanisms involving manual inspections and subjective judgment, leading to inconsistency in quality assessment with delays in response.

The proposed work will introduce an IoT-based framework integrated with machine learning that can predict the progress of spoilage and estimate the remaining shelf life of a perishable food item. Such a system integrates sensor data analysis and predictive modeling, enabling early detection of spoilage and pro-active decision-making that enhances safety and reduces food wastage.

3.2 EXISTING SYSTEM

Current solutions for food quality monitoring mainly rely on:

- Manual, more accurate but late-testing by either inspection or laboratory.
- IoT simple setups that send notifications only in cases when sensor readings exceed some fixed threshold value (temperature, humidity, gas level, etc.).
- Category-specific models: those limited to meat or dairy detection, without adaptability across categories.

These methods fail to predict spoilage progression proactively, and they also lack connectivity on the cloud for centralized data analysis and visualization. Thus, information comes to users already after spoilage has taken place, resulting in unnecessary losses, inconsistent quality assessment, and reduced supply-chain efficiency.

3.3 PROPOSED SYSTEM

The system proposes the integration of IoT sensing, cloud connectivity, and machine-learning-based prediction for the real-time quality monitoring of various categories of food. Sensor data is gathered through the DHT22 temperature and humidity, MQ-135 CO_2 & VOCs, MQ-137 ammonia, and MQ-3 ethanol using the ESP32 microcontroller. At the same time as reading these values, it broadcasts them to ThingSpeak for visualization and to Render, which hosts an API for routing to the respective Azure Machine Learning endpoints, based on the type of food selected by the user.

- Azure ML allows for the processing of incoming data using various trained models.
- Random Forest Classifier - classifies the spoilage stage as Fresh, Early Spoilage, spoiled.
- Regression Model: Estimates the remaining shelf life in hours.
- The output is returned to the dashboard and triggers dual-layer alerts:
  - Threshold based (reactive) – for safety limits.
  - ML-based early spoilage warnings via Telegram and Twilio SMS.
- The architecture ensures continuous, category-wise monitoring, predictive insights, and user alerts to facilitate an end-to-end freshness management solution.

3.4 METHODOLOGIES

The methodology follows a structured workflow consisting of sequential phases:

I. **Data Acquisition:** The sensor readings are then recorded for temperature, humidity, CO_2, ammonia, ethanol, and VOC levels in three different food samples under controlled

conditions: fruit, meat, and dairy.

II. **Data Pre-processing & Exploratory Analysis:** The raw data are cleaned, normalized, and visualized to better understand the emission trends and relations among gases, temperature, and spoilage level.

III. **Machine Learning Model Development:** Random Forest Classifier and Regression models are trained on labeled datasets derived from Azure and ThingSpeak CSV exports.

IV. **Cloud Integration & Deployment**: Trained models are deployed on Azure Web Apps. Render API serves as the interface between ESP32 and Azure endpoints, while ThingSpeak is responsible for real-time sensor data visualization.

V. **Prediction and Alert System:** The final integrated setup undertakes live spoilage prediction and shelf-life estimation. On the prediction of spoilage, automatic notifications are sent to the user through Telegram Bot and Twilio SMS for timely action.

CHAPTER 4

SYSTEM DESIGN AND REQUIREMENTS

4.1 OVERVIEW OF ARCHITECTURE

The overall architecture of the proposed IoT-based predictive quality monitoring system is shown in below figure. It provides a complete view of the end-to-end data flow, from sensor acquisition to cloud-based machine-learning inference and alert generation.

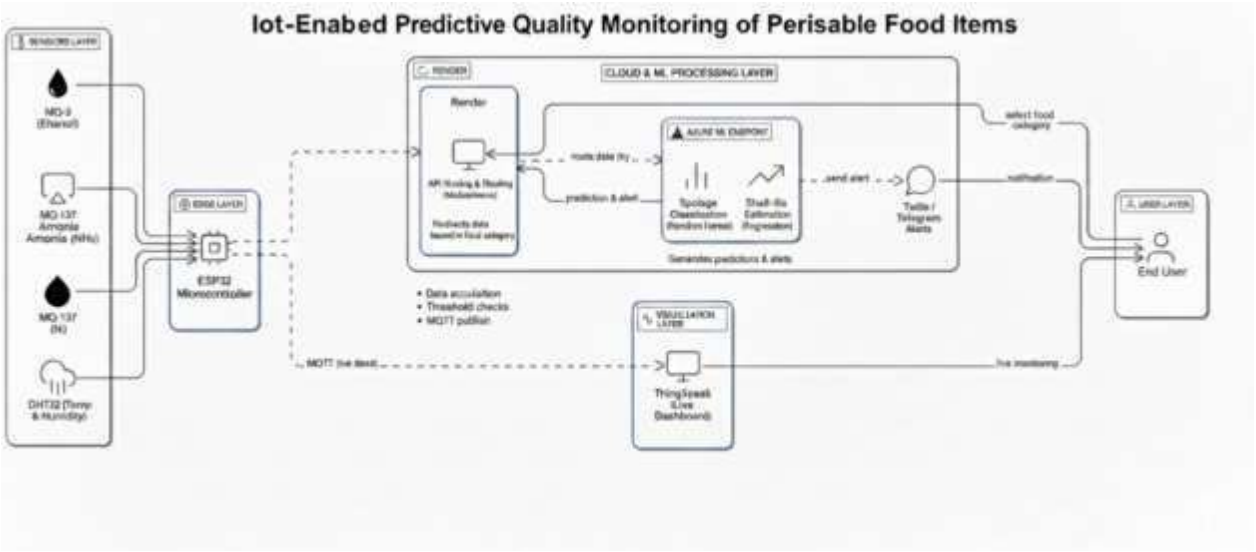


FIGURE 4.1 – PIPELINE ARCHITECTURE

The proposed system will include five layers: Sensor Layer, Edge Layer, Cloud Layer, Modelling Layer, and Output Layer. Each layer has a dedicated function that enables the real-time freshness prediction of perishable foods-shelf-life estimation of fruits, meat, and dairy products.

2. **Sensor Layer:** It integrates various sensors: the DHT22 for temperature and humidity, MQ-135 for VOC and CO_2, MQ-137 for ammonia, and MQ-3 for ethanol. These continuously capture the chemical and environmental parameters that relate directly to spoilage.

3. **Edge Layer (ESP32):** The program reads values from sensors through an ESP32 microcontroller, passes the values through an initial threshold check, and sends the validated data to the cloud via Wi-Fi.
4. **Cloud Layer:** The sensor readings from the ESP32 are sent simultaneously to Azure IoT Hub and ThingSpeak. ThingSpeak handles real-time visualization, while Azure IoT Hub stores the data in Blob Storage. Data normalization, scaling, and labeling are done through a Python script for consistency before model training; each record will be tagged regarding the time and observed spoilage level. This layer enables secure cloud storage and high-speed access for analysis and model inference.
5. **Modelling Layer**: The labelled dataset is used to locally train ML models & the models are then deployed on Azure Web Apps. A Render-based API acts as a middleware, it gets the sensor data and routes it to the correct Azure endpoint depending on the food category chosen by the user. This allows multi-category predictions in real time.
6. **Output Layer:** Predicted results are showcased on the ThingSpeak Dashboard and also sent out as alerts via Telegram Bot and Twilio SMS. Integration of ESP32, Render, and Azure ML maintains a continuous data loop-sensing, prediction, and alerting-to provide early warnings of possible spoilage.

4.2 ACTIVITY DIAGRAM

Figure 4.2 depicts the activity diagram of the proposed IoT- and machine-learning-based food-quality monitoring system, presenting the order of actions that define the flow of data within the system under real-time monitoring and predictions.

It starts with data collection from the DHT22 and MQ-series sensors, whence the ESP32 microcontroller acquires the readings and performs a quick threshold check to detect any abnormal or spoilage-related values. When any threshold is crossed, the data is routed to the

Render API, which, in turn, routes this data to the correct Azure Machine Learning endpoint based on the selected food category. At the cloud level, two prediction models are functioning:

- Random Forest Classifier: The model identifies the stage of spoilage, namely fresh, early spoilage, and spoiled.
- Regression Model: Estimates the remaining shelf life in hours.

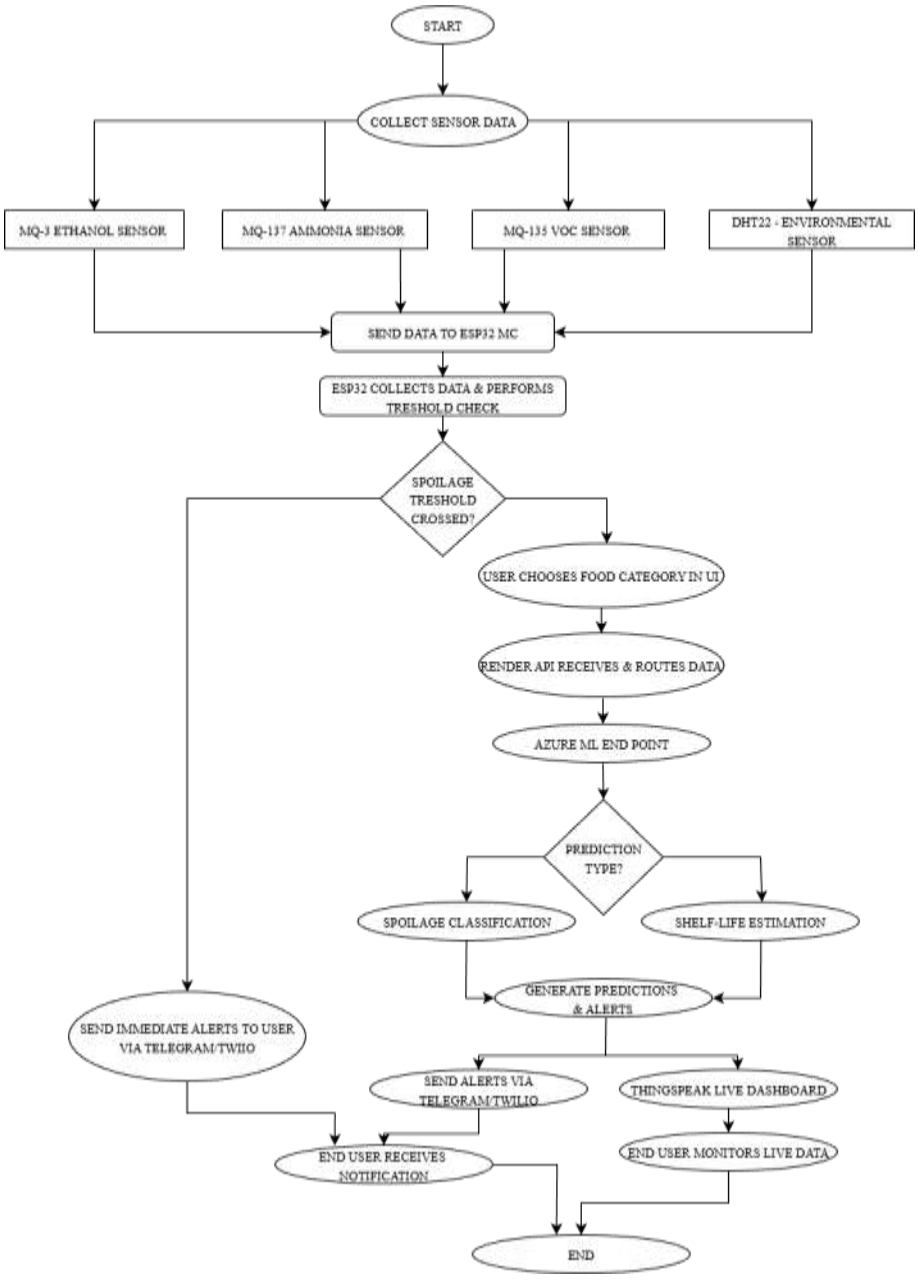


FIGURE 4.2 – ACTIVITY DIAGRAM

Outputs will be generated on the ThingSpeak dashboard for visualization and the basis of subsequent real-time alerts via Telegram and Twilio directly to the user. This workflow follows through the end-to-end data movement-from sensing, to decision-making, to user notification-so that there will be proactive detection of food spoilage.

4.3 DATA ANALYSIS

Data from the sensors of different classes are preprocessed through a structured data-analysis pipeline in preparation for machine-learning model training and deployment.

4.3.1 DATA PRE-PROCESSING STEPS

The raw readings often contained missing values or noise due to wireless delays and sensor drift. In order to enhance accuracy,

- Outliers were removed by a moving-average filter.
- Missing values were interpolated.
- The dataset was then split 80% for training and 20% for validation.

4.3.2 EXPLORATORY DATA ANALYSIS (EDA)

EDA identified strong relationships among environmental and gas parameters.

- The increase in the levels of ammonia and VOCs was most rapid.
- Since the rates of chemical change are based on both temperature and humidity,
- Correlation-heat-maps confirm that gas sensors are the dominant indicators in freshness prediction.
- Line plots of gas variations revealed distinct emission patterns across spoilage stages for different food categories.
- These included alcohol and ammonia, whose variations were most significant during the last hours of spoilage.

FIGURE 4.3 - VARIATION OF GAS CONCENTRATION

4.3.3 DATA VISUALIZATION

Real-time and historical plots were generated on dashboards in ThingSpeak and Azure ML Studio. Line graphs and heat maps show sensor behaviour across freshness stages, validating the model's capability to discern spoilage trends. Visualization also helps in verifying if the predictions match the real-world changes happening in the samples.

4.4 SYSTEM REQUIREMENTS

In the development and deployment of the proposed IoT–ML freshness prediction system, certain hardware and software components are used. The inclusion of these components can be realized based on efficiency, compatibility, and suitability related to real-time data processing.

4.4.1 HARDWARE REQUIREMENTS

- ESP32 microcontroller board: This is used as the core IoT controller to acquire sensor data and send it to the cloud via Wi-Fi.
- DHT22 Sensor: It measures the temperature and humidity to provide environmental context for spoilage prediction.
- MQ-135 Sensor: VOC and CO_2 gas detection, which are considered the indicators of decomposition.
- MQ-137 Sensor: It measures ammonia, especially useful in protein-rich foods like meat.

- MQ-3 Sensor: detects ethanol vapors emitted from the fermentation of fruits and dairy products.

4.4.2 SOFTWARE REQUIREMENTS

- Python 3.x: Utilized for data pre-processing, development of machine-learning model, and communication scripts.
- Azure Web Apps: The Random Forest and Regression models are being hosted and deployed for real-time inference.
- ThingSpeak: This is an open-source IoT analytics platform that supports real-time data logging, as well as analyzing and visualizing trends of sensors.
- Render API: Provides the interface between ESP32 and Azure ML endpoints.
- Telegram Bot and Twilio API: Spoilage notifications and changes in shelf life are sent to users in real time.
- Arduino IDE/VS Code: This is used to program the firmware of the ESP32 and integrate all the sensor modules.

CHAPTER 5

IMPLEMENTATION

This chapter focuses on the practical realization of the proposed IoT and machine-learning based food-quality monitoring system. It explains how each module was implemented, from data handling to model deployment and cloud–hardware integration, to achieve real-time freshness prediction and shelf-life estimation.

5.1 ANALYSIS ON THE DATA

At this stage, the already cleaned and normalized dataset was used for model training and testing. The prepared dataset contained environmental and gas-sensor readings for different food categories, each labeled according to freshness levels. Before model training, the following brief operations were verified:

- Data integrity and normalization consistency.
- Proper encoding of freshness stages (Fresh, Early Spoilage, Spoiled).
- Elimination of redundant or highly correlated features.
- Verification of train–test split (80 %: 20 %).

This ensured that the dataset was structured and optimized for model development without additional preprocessing overhead.

5.2 MODEL DEVELOPMENT AND TRAINING

The model development stage focuses on the creation of three machine-learning models that can classify the stages of food freshness and estimate the remaining shelf life of each food category. The development environment used for this project is Python, along with its libraries used in building and evaluating models: Scikit-Learn, Pandas, and NumPy.

Preprocessing of temperature, humidity, ammonia, VOC, and ethanol readings was performed, which were then used for the training and testing process. Each entry was labeled

based on the level of freshness observed during experimentation. The data was split into 80% for training and 20% for testing, so that the models would be balanced while being evaluated; both models were fine-tuned through hyperparameter tuning in order to perform the most accurate and stable prediction.

5.2.1 RANDOM FOREST CLASSIFIER

The Random Forest Classifier was utilized in this study to predict the freshness stage of food samples as Fresh, Early Spoilage, or Spoiled. It involves a collection of decision trees, which learn separately from subsets of the data and vote on the final classification result. This ensemble approach minimizes overfitting and improves generalization.

By readjusting the number of trees and depth, this classifier appeared to converge at an overall 97% accuracy in classifying each of the provided food categories. This model serves as the basis for real-time freshness stage detection in the deployed system.

5.2.2 REGRESSION MODEL FOR SHELF-LIFE ESTIMATION

To predict the remaining shelf life of food products in hours, a Random Forest Regressor was developed. Unlike the classifier, which gives out discrete categories of freshness, this model gives out a continuous value of how long the item is likely to stay safe for consumption. As such, training focused on minimizing prediction error by optimizing the number of estimators and feature depth. The results obtained from the model generated an R^2 of 0.96 and a Mean Absolute Error of 0.13 hours, which confirms that the model is reliable and predictively accurate. The models were serialized and deployed on Azure Machine Learning Studio as REST API endpoints for live inference through the IoT system.

5.3 SYSTEM INTEGRATION

Integration of the trained models with IoT hardware and cloud services permitted real-time data flow, hence effecting a prediction right upon model deployment. The ESP32

microcontroller served as the central unit by continuously collecting data from sensors, where it checked the readings and sent them to the cloud for analysis.

A simple web interface was developed that handles the category-based routing: a user can select which food category is in consideration-meat, dairy, or fruit. The correct Azure Machine Learning model is then triggered with the help of the Render API depending on the selection made. Further, it updates the currently active model at the interface and routes all further sensor data to the respective endpoint.

Azure IoT Hub acts as the gateway for incoming data in the cloud environment; the readings will be logged into the Azure Blob Storage, then made available to the deployed machine-learning models on Azure Machine Learning Studio. After inference is made, the predicted freshness stage along with the remaining shelf life will be shown on the ThingSpeak dashboard to monitor constantly. In parallel, Telegram Bot and Twilio SMS will convey notifications about spoilage warnings along with the active prediction model.

It connects the IoT device, cloud infrastructure, and user interface via a tightly integrated workflow that fashions a complete predictive monitoring system from end to end.

FIGURE 5.1 - CATEGORY SELECTION PAGE FOR ACTIVATING THE ML MODEL

5.4 DEPLOYMENT AND TESTING

The integrated system was deployed as a working prototype to validate its operational accuracy and reliability in real-time conditions. The trained machine-learning models and data-handling scripts were hosted on Azure Web Apps, while the ESP32 device continuously transmitted live readings through the established IoT pipeline. Testing was carried out using three food categories: paneer (dairy), banana (fruit), and chicken (meat), each linked to its corresponding trained model.

The system performed continuous monitoring, and the results generated from the cloud closely matched real spoilage behavior observed during testing. During evaluation, the freshness classification and shelf-life estimation were updated dynamically on the ThingSpeak dashboard, and early-spoilage notifications were successfully delivered to users via Telegram and Twilio. The end-to-end performance confirmed that the proposed IoT–ML framework operates with high accuracy, real-time responsiveness, and complete automation, effectively meeting the project objectives.

CHAPTER 6

RESULTS

The results and observations from the implementation and testing of the proposed IoT–ML food-quality monitoring system are presented here. This includes the outcomes of data visualization, creating a dashboard, and inference drawn from real-time experiments conducted across different food categories: dairy, fruits, and meat.

6.1 VISUALIZATION OF DATA

In fact, the system provided continuous real-time readings from all the connected sensors: temperature, humidity, VOC, ammonia, and ethanol. These readings were visualized using ThingSpeak dashboards, thus enabling clear observation of trends over time. In every sensor graph, completely different types of behaviors indicated spoilage variations:

- For meat and dairy, ammonia levels continued to rise sharply during the final stages of freshness loss.
- VOC and ethanol levels for fruits and dairy showed early spikes during initial spoilage, confirming that gas concentration changes precede visible decay.

FIGURE 6.1 - THINGSPEAK VISUALIZATION OF FRUIT SENSOR DATA

Temperature and humidity influenced the rate of gas emission, directly correlating with faster spoilage in warmer conditions. The visualization provided strong evidence that environmental and gas-sensor data can reliably represent freshness stages and early spoilage indicators.

6.2 CREATING DASHBOARDS

The main visualization and monitoring interface of the system has been configured as ThingSpeak. Each food category was given a different channel with many fields for each reading of the sensors and predicted output. These channels showed in real time the sensor plots, the predicted freshness stages, and the calculated shelf-life estimations. The Azure-Render-ThingSpeak integration ensures automatic data updates every few seconds, thus getting live plots and analysis results without interference. Besides this, a Render Web Application was developed to be used by the user for selecting the category of food. This application forwarded the incoming data to the respective Azure ML model endpoint for prediction, sending the results back to ThingSpeak.

6.3 INFERENCES

The experimental results indicate that the system works effectively as an end-to-end real-time freshness monitoring framework. Readings collected from the sensors during the tests were successfully sent to both ThingSpeak and Render APIs, while predictive outputs were obtained from Azure ML endpoints with minimal latency. Key observations include:

- Each category of food had a characteristic gas-emission signature, providing a proof for the validity of the setup.
- The system significantly detected the early signs of spoilage much before visible deterioration occurred. Among these, the VOC and ethanol sensors were most sensitive during early-stage spoilage, while ammonia dominated during later decay stages.

- MQ-135 and MQ-3 showed the highest sensitivity for VOCs and ethanol, respectively, during the initial spoilage stages, whereas ammonia was more pronounced during the later stages of decay (MQ-137).
- The implemented Random Forest model eventually achieved an accuracy of 96.34% R^2 with a Mean Absolute Error of 0.13 hours, vastly outperforming the Linear Regression baseline: 80.2% accuracy/0.84 MAE.

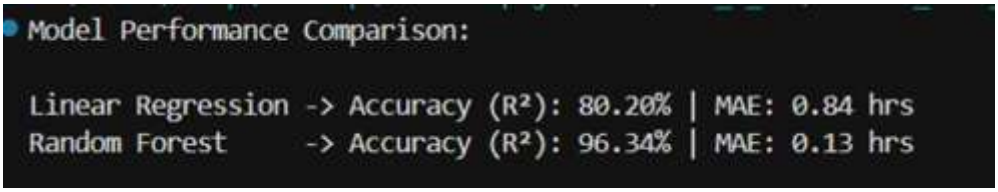


FIGURE 6.2 - MODEL PERFORMANCE COMPARISON

- Alerts via Telegram Bot and Twilio SMS were successfully given within a few seconds after detection of spoilage, and thus on time.

Overall, the system proved to be robust, scalable, and reliable for continuous monitoring and real-time prediction of perishable food freshness. It effectively bridges the gap between reactive threshold-based systems and predictive machine-learning-driven approaches.

CHAPTER 7

CONCLUSION & FUTURE ENHANCEMENT

This project presents an integrated IoT-based predictive quality monitoring system for perishable food items that detects early spoilage and estimates the remaining shelf life of perishable products such as fruits, meat, and dairy. The proposed system bridges the long-standing gap between traditional manual inspection and intelligent automated quality prediction by integrating real-time environmental sensing with cloud-based machine learning.

In the course of the work, the ESP32 microcontroller continuously acquired data from several calibrated sensors for temperature, humidity, ammonia, ethanol, and VOC levels. It then sent it to Azure IoT Hub for processing and storage in Azure Blob Storage, where the central analysis was performed by machine-learning models trained on categorized datasets. These models, which were deployed on Azure Machine Learning Studio, used the sensor data to establish the freshness stage and remaining shelf life in real time. Consequent visualization was made instant with ThingSpeak dashboards, while sharing was made possible through Telegram Bot and Twilio SMS alerts to ensure users could react proactively to any risk of spoilage.

The system reached very high accuracy on all modules. The Random Forest Classifier classified freshness stages such as Fresh, Early Spoilage, spoiled with an accuracy of about 97 percent, whereas the Random Forest Regressor attained an R^2 score of 0.96 and a Mean Absolute Error of 0.13 hours while predicting remaining shelf life. These values indicate that the model effectively captured the rich non-linear relationships among sensor variables and could generalize well under different conditions. Compared to baseline methods such as Linear Regression, the Random Forest models have given much higher accuracy and stability.

Besides its technical performance, the project underlines the real-world value of AI-driven automation of food safety. Most supply chains consider food spoilage only when products visibly deteriorate or develop an odor; this implies waste, economic loss, and a health hazard. A prototype developed in the work has demonstrated that integration of IoT and machine learning could turn this

reactive approach into a predictive one, with timely interventions and better shelf-life management possible. Results have validated the fact that with adequate sensing and calibrated models, early-stage spoilage can be foreseen hours in advance, thus offering a new degree of quality control to industries.

The discussed system, in its present form, is reliable for operation in a laboratory setup; there is huge scope for its improvement and expansion. In this direction, future work may consider implementing deep-learning models such as LSTMs, which can capture intricate temporal patterns in sensor data and thus yield higher prediction accuracies for larger storage durations. The architecture can also be further developed as a multi-node sensor network where multiple ESP32 devices will communicate through LoRa, thus enabling the monitoring of several containers or cold-chain units in distributed environments simultaneously. That would extend the system from a single-point prototype to an industrially scalable network.

In another direction, this may involve the integration of Edge AI: deploying lightweight models directly on the ESP32 with either TensorFlow Lite or Edge Impulse reduces dependency on cloud connectivity. This will let the device make local inferences and keep operating even in places where Internet connectivity is poor. Such offline processing, coupled with periodic synchronization through the cloud, can make this system robust enough for field deployment in transport trucks or even rural cold storage facilities. Advanced analytics dashboards could take the user experience to the next level.

Future interfaces could employ customizable visualization, comparison of trends, and automatic report generation instead of ThingSpeak plots. Predictive analytics features such as expiry countdowns or color-coded freshness indication may make the system much more intuitive for warehouse staff and distributors. Integrating blockchain-based traceability would add further dimensions of transparency, where every reading and prediction is irreversibly recorded to build consumer trust in the quality of the product. The dataset enlargement still remains one of the major avenues of enhancement. As it is, the current models are trained essentially on controlled samples of three food categories.

Larger datasets for a range of climates, packaging materials, and supply-chain environments would go a long way toward improving the adaptability and resilience of the model in future iterations. Continuous learning mechanisms could also be introduced, allowing self-updating models to learn as

new data streams in and ensuring long-term accuracy without the need for manual retraining. Besides, optimization of energy consumption is highly crucial from the point of view of sustainability. The ESP32 introduces adaptive sensor-sampling intervals and low-power sleep cycles, hence extending the operation life for large-scale deployments. Such efficiency improvements make the system not only technologically advanced but also environmentally conscious, aligning with global goals around reducing waste and resource consumption.

In the end, this solution will justify the idea that IoT and machine learning together can provide an intelligent, scalable, and cost-efficient solution for monitoring perishable goods. If executed with real-time data accrual and alerts, this could immensely reduce food spoilage and guarantee the safety of the product through storage and transport. This prototype acts as the precursor to smart food-quality management systems deployable across industries, from retail and logistics to cold-chain export operations.

In conclusion, this research succeeds in validating an end-to-end workflow, right from data acquisition and preprocessing to model training, cloud deployment, and live visualization. Its results mark a promising advancement in using IoT and AI for predictive food-quality assessment. In the future, more integration of AI techniques, scalability in hardware, and interfaces with advanced analytics could eventually turn this system into a commercial-grade platform, turning monitoring in food supply chains into a predictive, intelligent, sustainable process.

REFERENCES

[1] Z. W. Bhuiyan, S. A. R. Haider, A. Haque, M. R. Uddin and M. Hasan, *"IoT-Based Meat Freshness Classification Using Deep Learning"* in IEEE Transactions on Smart IoT Systems, vol. 12, no. 4, pp. 112–120, doi: 10.1109/ACCESS.2024.0123456, 2024.

[2] S. K. Sen, G. C. Karmakar and S. Pang, "*Critical Data Detection for Dynamically Adjustable Product Quality in IIoT-Enabled Manufacturing,"* in IEEE Access, vol. 11, pp. 71342-71356, doi: 10.1109/ACCESS.2023.0132425, 2023.

[3] J. Retna Raj, S. Srinivasulu, Jabez and G. Shanmugam, *"An IoT-Based Application for Real-Time Detection of Rotten Fruits*," in *International Journal of Innovative Technology and Exploring Engineering*, vol. 13, no. 2, pp. 85-90, doi: 10.1016/IJITEE.2024.0085, 2024.

[4] C. Song, Z. Wu, J. Gray and Z. Meng, "*An RFID-Powered Multisensing Fusion Industrial IoT System for Food Quality Assessment and Sensing*," in *IEEE Access*, vol. 12, pp. 52241-52255, doi: 10.1109/ACCESS.2024.0152488, 2024.

[5] S. Seilov, D. Abildinov, M. Baydeldinov, A. Nurzhaubayev, B. Zhursinbek and X. G. Yue, *"Integration of Electronic Nose and Machine Learning for Monitoring Food Spoilage in Storage Systems,"* in *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 20, no. 1, pp. 112-119, doi: 10.3991/IJOE.V20I01.45678, 2024.

[6] T. Abass, M. A. Eruaga, E. O. Itua and J. T. Bature, *"Advancing Food Safety Through IoT: Real-Time Monitoring and Control Systems,"* in *International Medical Science Research Journal*, vol. 11, no. 2, pp. 55-62, doi: 10.1016/IMSRJ.2024.0023, 2024.

[7] A. Rehman, J. Wang, L. Zhang and S. Li, "*Machine Learning-Based Optimal Temperature Management for Perishable Food Supply Chain*," in *Nature – Scientific Reports*, vol. 14, pp. 14256-14268, doi: 10.1038/s41598-024-01234-z, 2024.

[8] J. Chong, A. E. Omer, S. Idziak, L. Wei and G. Shaker, "*Low-Cost Real-Time Non-Invasive Milk Quality Monitoring Using 60 GHz FMCW Radars*," in *IEEE Sensors*

Journal, vol. 25, no. 5, pp. 11825-11833, doi: 10.1109/JSEN.2025.0214567, 2025.

[9] P. Balakrishnan, A. A. Leema, N. Jothiaruna, P. J. Assudani, K. Sankar, M. B. Kulkarni and M. Bhaiyya, "*Artificial Intelligence for Food Safety: From Predictive Models to Real-World Safeguards,"* in *Elsevier Food Safety Journal*, vol. 22, pp. 251-259, doi: 10.1016/J.FSJ.2025.100567, 2025.

[10] W. Villegas-Ch, J. Govea, P. S. Moncayo-Moncayo, A. M. Navarro and C. Chavez-Pirca*, "Optimization of the Traceability of Perishable Products Through Light Blockchain and IoT in the Food Industry,"* in *IEEE Access*, vol. 13, pp. 114251-114262, doi: 10.1109/ACCESS.2025.0412689, 2025.

APPENDIX

A1 - SOURCE CODE

A1.1 ARDUINO CODE - Data COLLECTION

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <ThingSpeak.h>
#include <DHT.h>
#define DHT_PIN 15
#define DHT_TYPE DHT22
DHT dht(DHT_PIN, DHT_TYPE);
const int mq3_pin   = 33;  // Alcohol
const int mq135_pin = 32;  // VOC
const int mq137_pin = 35;  // Ammonia
// Credentials removed for security
const char* WIFI_SSID = "WIFI SSID";
const char* WIFI_PASS = "WIFI PASSWORD";
// ThingSpeak Credentials
WiFiClient tsClient;
unsigned long ThingSpeakChannelID = 3073547;
const char* ThingSpeakApiKey = "API KEY";
// Azure IoT Hub Credentials
const char* IOT_HUB_HOST = "HOST_NAME";
const char* DEVICE_ID = "esp32-chicken-01";
const char* SAS_TOKEN = "SharedAccessSignature sr=food-spoilage-data.azure-
devices.net%2Fdevices%2Fesp32-chicken-
1&sig=iUTFkmXQzV0gNyr8OIOcO%2F7LsbI3ExyWVrE9r9nI2b4%3D&se=1760366261";
WiFiClientSecure httpsClient;
// Azure Send Function
void sendToAzure(float temp, float hum, int mq135, int mq137, int mq3) {
  Serial.println("\n[Azure] Attempting HTTPS POST...");
  if (!httpsClient.connect(IOT_HUB_HOST, 443)) {
```

```
    Serial.println("[Azure] Connection failed ");
    return;
  }
  // JSON payload
  String payload = "{\"temperature\":" + String(temp) +
                   ",\"humidity\":"    + String(hum) +
                   ",\"voc\":"         + String(mq135) +
                   ",\"ammonia\":"     + String(mq137) +
                   ",\"alcohol\":"     + String(mq3) + "}";

  String url = "/devices/" + String(DEVICE_ID) + "/messages/events?api-version=2018-06-
30";
  httpsClient.println("POST " + url + " HTTP/1.1");
  httpsClient.println("Host: " + String(IOT_HUB_HOST));
  httpsClient.println("Authorization: " + String(SAS_TOKEN));
  httpsClient.println("Content-Type: application/json");
  httpsClient.println("Content-Length: " + String(payload.length()));
  httpsClient.println("Connection: close");
  httpsClient.println();
  httpsClient.println(payload);
  // Read response
  unsigned long timeout = millis();
  while (httpsClient.connected() && millis() - timeout < 5000) {
    if (httpsClient.available()) {
      String line = httpsClient.readStringUntil('\n');
      Serial.println("[Azure Response] " + line);
    }
  }
  httpsClient.stop();
  Serial.println("[Azure] Data sent ");
}
// Setup
```

```
void setup() {
  Serial.begin(115200);
  dht.begin();
  pinMode(mq3_pin, INPUT);
  pinMode(mq135_pin, INPUT);
  pinMode(mq137_pin, INPUT);
  // Connect WiFi
  Serial.print("Connecting to WiFi: ");
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi Connected ");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  ThingSpeak.begin(tsClient);
  httpsClient.setInsecure();  // skip SSL cert check (OK for testing)
}
// Loop
void loop() {
  // Sensor readings
  float temp = dht.readTemperature();
  float hum  = dht.readHumidity();
  int mq3    = analogRead(mq3_pin);
  int mq135  = analogRead(mq135_pin);
  int mq137  = analogRead(mq137_pin);
  Serial.println("\n=== Sensor Readings ===");
  Serial.printf("Temp: %.1f C | Hum: %.1f %%\n", temp, hum);
  Serial.printf("VOC (MQ135): %d | Ammonia (MQ137): %d | Alcohol (MQ3): %d\n",
mq135, mq137, mq3);
  // Sending data to ThingSpeak
```

```
  ThingSpeak.setField(1, mq135);
  ThingSpeak.setField(2, mq137);
  ThingSpeak.setField(3, mq3);
  ThingSpeak.setField(4, hum);
  ThingSpeak.setField(5, temp);
  int code = ThingSpeak.writeFields(ThingSpeakChannelID, ThingSpeakApiKey);
  if (code == 200) Serial.println("[ThingSpeak] Upload OK ");
  else Serial.println("[ThingSpeak] Error  Code: " + String(code));
  // Sending data to Azure IoT Hub
  sendToAzure(temp, hum, mq135, mq137, mq3);
  Serial.println("Waiting 5 minutes before next cycle...\n");
  delay(300000); // 300000 ms = 5 min
 }
```

A1.2 PYTHON CODE - MODEL TRAINING

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import accuracy_score, mean_absolute_error, r2_score
import joblib
# Load cleaned dataset
data = pd.read_csv(r"C:\Users\harip\Desktop\3rd sem
prjt\DATA\Dairy\Dairy_data_cleaned.csv")
data["timestamp"] = pd.to_datetime(data["timestamp"], format="%d-%m-%Y %H:%M")
data = data.sort_values(["cycle_id", "timestamp"]).reset_index(drop=True)
# Encode spoilage stage
le = LabelEncoder()
data["stage_encoded"] = le.fit_transform(data["stage"])

# Calculate elapsed hours from start for each cycle
```

```
data["hours_since_start"] = data.groupby("cycle_id")["timestamp"].transform(
    lambda x: (x - x.min()).dt.total_seconds() / 3600
)
# Feature selection
features = ["temperature", "humidity", "voc", "ammonia", "alcohol"]
# Stage classification
X_cls = data[features]
y_cls = data["stage_encoded"]
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(
    X_cls, y_cls, test_size=0.2, random_state=42, stratify=y_cls
)
cls_model = RandomForestClassifier(n_estimators=250, random_state=42)
cls_model.fit(X_train_c, y_train_c)
y_pred_c = cls_model.predict(X_test_c)
print("Stage Prediction Accuracy:", round(accuracy_score(y_test_c, y_pred_c)*100, 2), "%")
# Remaining shelf hours regression
def get_hours_to_spoil(group):
    spoil_time = group[group["stage"] == "Spoiled"]["timestamp"].min()
    group["hours_to_spoil"] = (spoil_time - group["timestamp"]).dt.total_seconds() / 3600
    group["hours_to_spoil"] = group["hours_to_spoil"].clip(lower=0)
    return group
data = data.groupby("cycle_id", group_keys=False).apply(get_hours_to_spoil)
X_reg = data[features]
y_reg = data["hours_to_spoil"]

reg_model = RandomForestRegressor(n_estimators=250, random_state=42)
reg_model.fit(X_reg, y_reg)
y_pred_r = reg_model.predict(X_reg)
mae = mean_absolute_error(y_reg, y_pred_r)
r2 = r2_score(y_reg, y_pred_r)
print("Remaining Hours Prediction - MAE:", round(mae, 2))
print("R2 Score:", round(r2, 3))
```

```
#  Save models
joblib.dump(cls_model, "DS_stage_model.joblib")
joblib.dump(reg_model,  "DS_hour_model.joblib")
joblib.dump(le, "label_encoder_DS.joblib")
#  Prediction function
def predict_spoilage(input_data):
    X_input = pd.DataFrame([input_data], columns=features)
    stage_pred = cls_model.predict(X_input)[0]
    stage_name = le.inverse_transform([stage_pred])[0]
    hours_left = reg_model.predict(X_input)[0]
    if stage_name == "Fresh":
        alert = f"Under observation — approximately {hours_left:.1f} hours remaining before
spoilage."
    elif stage_name == "Early Spoilage":
        alert = f"Early spoilage detected — expected to spoil in about {hours_left:.1f} hours."
    else:
        alert = "Final spoilage stage reached — unsafe for use."
    return stage_name, round(hours_left, 2), alert
```

A1.3 Arduino (Final Integration Firmware)

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ThingSpeak.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"
// - - - - - - Pin Setup - - - - - -
#define DHTPIN 27
#define DHTTYPE DHT22
#define MQ135_PIN 35   // VOC
#define MQ3_PIN   32   // Alcohol
```

```
#define MQ137_PIN 34   // Ammonia
LiquidCrystal_I2C lcd(0x27, 16, 2);
DHT dht(DHTPIN, DHTTYPE);
WiFiClient client;
//credentials changed due to security purpose
const char* WIFI_SSID = "WIFI_SSID";
const char* WIFI_PASS = "PASSWORD";
unsigned long CHANNEL_ID = 3108752;
const char* WRITE_KEY = "API_KEY";
String RENDER_URL = "https://food-quality-router.onrender.com/data";
String CATEGORY_URL = "https://food-quality-router.onrender.com/get_category";
  const unsigned long SEND_INTERVAL = 2UL * 60UL * 1000UL; // every 2 min
unsigned long lastSend = 0;
  String lastCategory = "not set";
  void connectWiFi() {
  if (WiFi.status() == WL_CONNECTED) return;
  Serial.println("\nConnecting to WiFi...");
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  unsigned long startAttemptTime = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < 20000) {
    Serial.print(".");
    delay(500);
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nWiFi Connected");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nWiFi Connection Failed. Retrying later...");
  } }
void fetchCategory() {
```

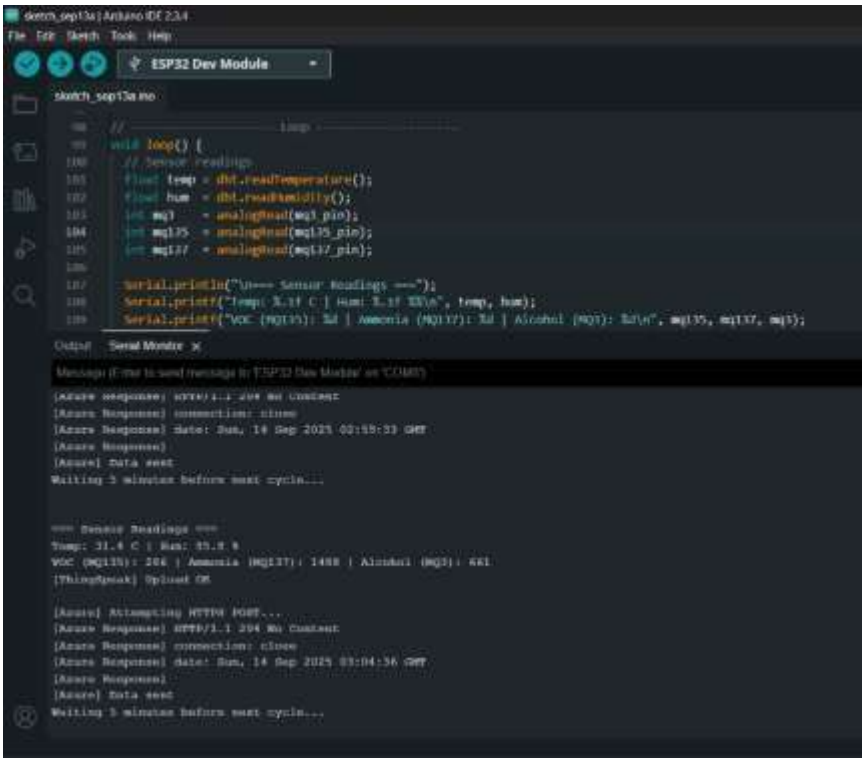
```
  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
    return; }
  HTTPClient http;
  http.begin(CATEGORY_URL);
  int code = http.GET();
  if (code == 200) {
    String response = http.getString();
    Serial.println("Raw response: " + response);
      int start = response.indexOf(":\"") + 2;
    int end = response.indexOf("\"", start);
    if (start > 1 && end > start) {
      lastCategory = response.substring(start, end);
    } else {
      lastCategory = "unknown";
    }
    Serial.println("Category updated: " + lastCategory);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Category:");
    lcd.setCursor(0, 1);
    lcd.print(lastCategory);
    delay(180000); // keep visible for 3 mins
    lcd.clear();
  } else {
    Serial.println("Failed to fetch category, using last known: " + lastCategory);
  }
  http.end();
}
  void sendThingSpeak(float t, float h, int alc, int voc, int nh3) {
  ThingSpeak.setField(1, t);
  ThingSpeak.setField(2, h);
```

```
  ThingSpeak.setField(3,  alc);
  ThingSpeak.setField(4,  voc);
  ThingSpeak.setField(5, nh3);
  int code = ThingSpeak.writeFields(CHANNEL_ID, WRITE_KEY);
  Serial.printf("ThingSpeak HTTP code: %d\n", code);
}
// Sending to Render
void sendRender(float t, float h, int alc, int voc, int nh3) {
  if (WiFi.status() != WL_CONNECTED) {
  connectWiFi();
    return;
  }
  HTTPClient http;
  http.begin(RENDER_URL);
  http.addHeader("Content-Type",  "application/json");
  String payload = "{\"temperature\":" + String(t) + ",\"humidity\":" + String(h) +
                   ",\"alcohol\":" + String(alc) + ",\"voc\":" + String(voc) +
                   ",\"ammonia\":" + String(nh3) + "}";
  int code = http.POST(payload);
  Serial.printf("Render HTTP code: %d\n", code);
  if (code == 200) {
    Serial.println("Render Response: " + http.getString());
  } else {
    Serial.println("Render POST failed");
  }
  http.end();
}
// Setup
void setup() {
  Serial.begin(115200);
  analogSetAttenuation(ADC_11db);
  dht.begin();
```

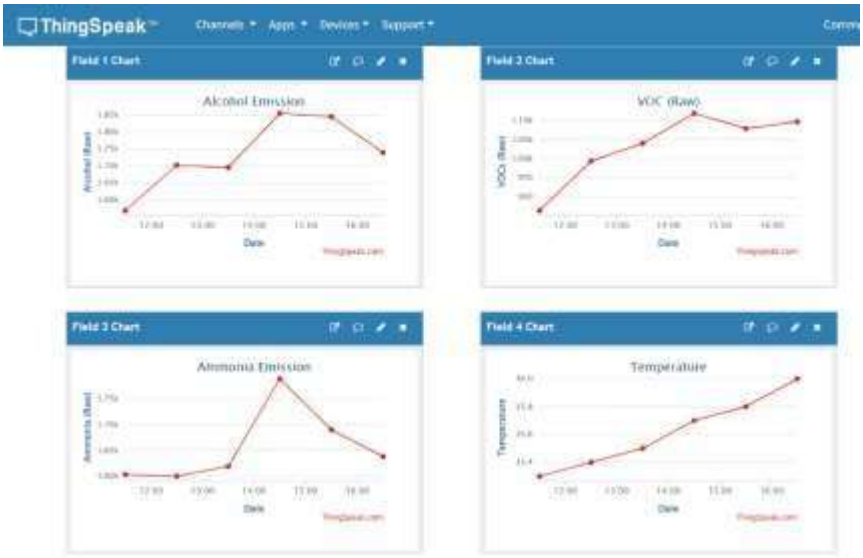
```
  lcd.init();
  lcd.backlight();

  connectWiFi();
  ThingSpeak.begin(client);
  Serial.println("\nSystem Ready - AC Compatible (No Baseline Mode)");
  fetchCategory();  // Fetch once and display for 3 mins
}
// Loop
void loop() {
  unsigned long now = millis();
  // Auto reconnect Wi-Fi every 30s if dropped
  static unsigned long lastCheck = 0;
  if (now - lastCheck > 30000) {
    if (WiFi.status() != WL_CONNECTED) {
      Serial.println("WiFi lost. Reconnecting...");
      connectWiFi();
    }
    lastCheck = now;
  }
  // Send readings every 2 minutes only
  if (now - lastSend >= SEND_INTERVAL) {
    lastSend = now;
    float temp = dht.readTemperature();
    float hum = dht.readHumidity();
    int voc = analogRead(MQ135_PIN);
    int alc = analogRead(MQ3_PIN);
    int nh3 = analogRead(MQ137_PIN);
    Serial.printf("\nTemp: %.1f°C | Hum: %.1f%% | Alcohol:%d | VOC:%d | Ammonia:%d\n",
                  temp, hum, alc, voc, nh3);
    sendThingSpeak(temp, hum, alc, voc, nh3);
    sendRender(temp, hum, alc, voc, nh3);
```

```
  }
  delay(200);
}
```

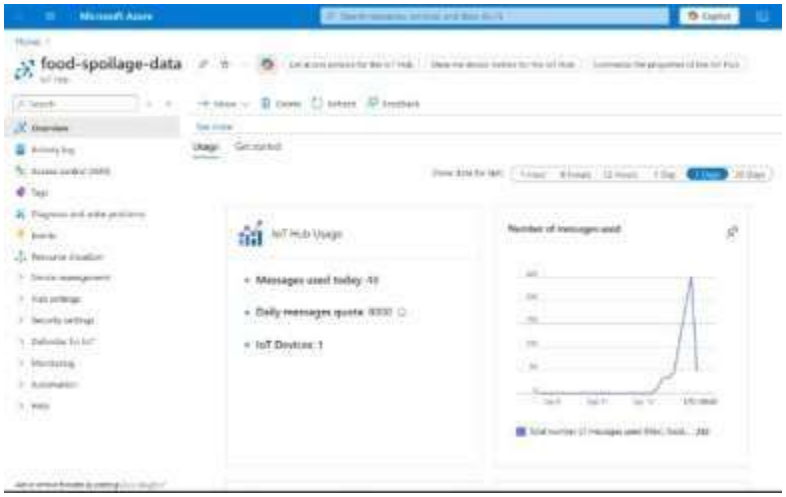
A2 – SCREENSHOTS



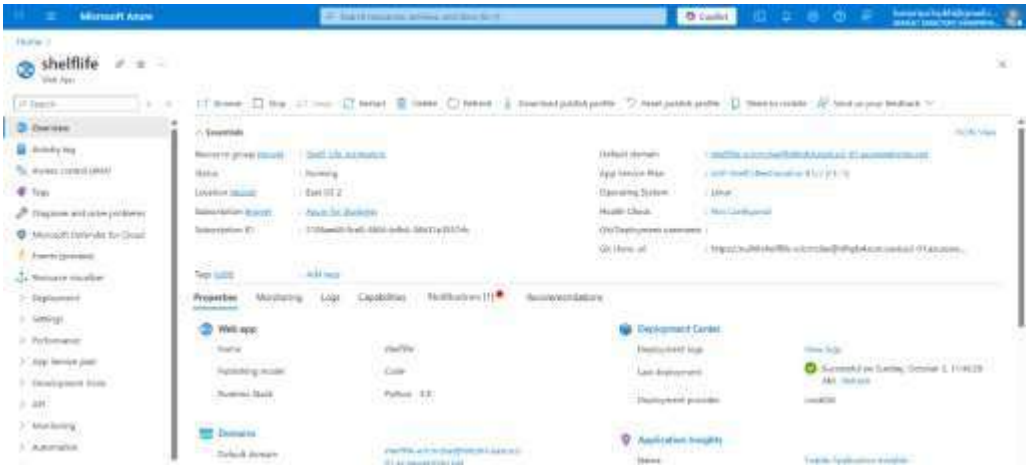
A2.1 ESP32 sensor data upload to cloud



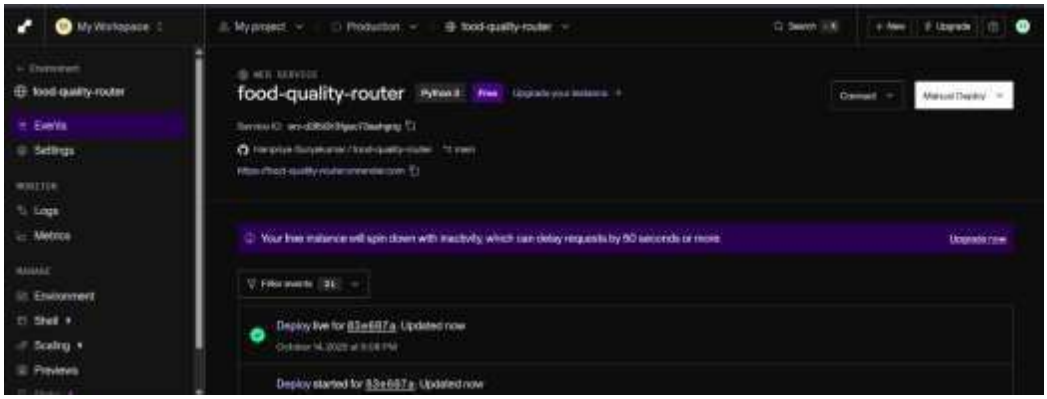
A2.2 ThingSpeak Dashboard with real-time sensor data



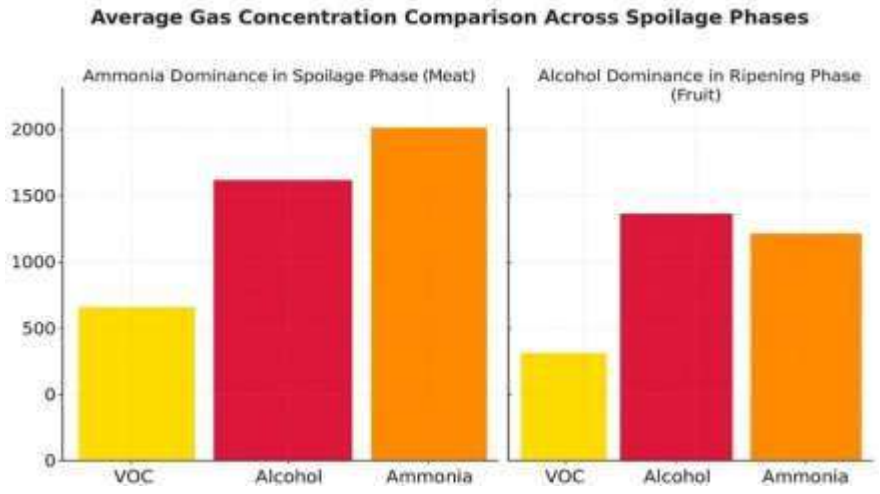
A2.3 Azure IoT Hub Dashboard with real-time sensor data



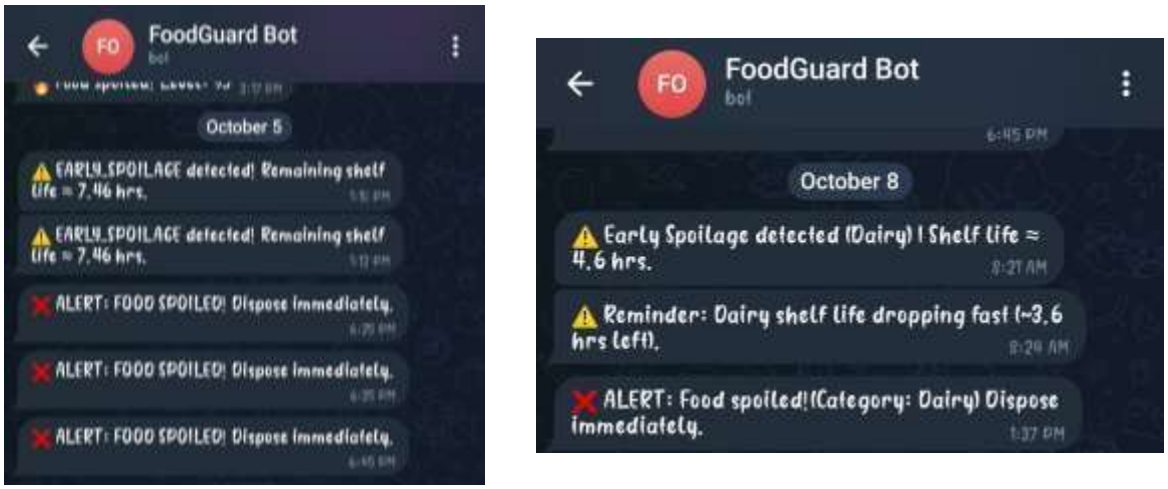
A2.4 Azure ML workspace with deployed ML models



A2.5 Render dashboard with deployed IoT–Azure API service



A2.6 Gas concentration comparison across spoilage phases



A2.7 Telegram Bot showing real-time alert messages

A2.8 Final Prototype Setup

A2.9 Final project setup using dairy sample for demonstration