ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ, ಬೆಳಗಾವಿ

Visvesvaraya Technological University

"Jnana Sangama" Belagavi – 590018

Project Report

On

"Smart Grocery Packaging for Expiry Detection"

Submitted in partial fulfilment for the award of degree of

Bachelor of Engineering

By

Bindiya Sail	**1AT21EC022**
Chhaya Naidu	**1AT21EC026**
Mohammed Abbas	**1AT21EC084**
Shashank SN	**1AT21EC144**

Under the Guidance of

Prof. Kavitha S

Assistant Professor

Department of Electronics & Communication Engineering,

Atria Institute of Technology - Bengaluru

Department of Electronics & Communication Engineering

Atria Institute of Technology

Bengaluru-560024

2024-2025

ATRIA INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
BENGALURU

Department of Electronics & Communication Engineering

CERTIFICATE

Certified that the project work entitled **"Smart Grocery Packaging for Expiry Detection"** carried out by

Ms. Bindiya Sail	**1AT21EC022**
Ms. Chhaya Naidu	**1AT21EC026**
Mr. Mohammed Abbas	**1AT21EC084**
Mr. Shashank SN	**1AT21EC144**

a bonafide students of VIII Semester Electronics & Communication Engineering, Atria Institute of Technology, **Bengaluru** in partial fulfillment for the award of Bachelor of Engineering in Visvesvaraya Technological University, Belagavi during the year **2024-25**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Prof. Kavitha S **Assistant Prof.** **Dept. of E&CE** **ATRIA IT, Bengaluru**	**Dr. Jagadeesh H S** **Prof. & Head, Dept. of E&CE,** **ATRIA IT, Bengaluru**	**Dr. Rajesha S** **Principal,** **ATRIA IT, Bengaluru**

Name of the Examiners

1.
2.

DECLARATION

We Ms. **Bindiya Sail - 1AT21EC022,** Ms. **Chhaya Naidu -1AT21EC026,** Mr. **Mohammed Abbas - 1AT21EC084** and Mr. **Shashank SN - 1AT21EC144** students of final year Bachelor of Engineering , Department of Electronics and Communication Engineering, Atria Institute of Technology Bangalore, would hereby declare the project entitled **"Smart Grocery Packaging for Expiry Detection"** has been carried out by us at Atria Institute of Technology Bengaluru and submitted in partial fulfillment of the course requirements for the award of degree of Bachelor of Engineering in Electronics and Communication Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2024 – 2025.

Place: Bengaluru
Date:

Signature of Students

Bindiya Sail (1AT21EC022)

Chhaya Naidu(1AT21EC026)

Mohammed Abbas(1AT21EC084)

Shashank SN(1AT21EC144)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without mention of the people who made it possible, whose constant guidance and encourages crowned our effort of success. We take this opportunity to express deepest gratitude and appreciation to all those who helped us directly or indirectly towards the successful completion of final year project work.

We express our gratitude and respect to **Dr. Rajesha S**, Principal, Atria IT, Bangalore for granting permission to carry out the project.

We express our heartfelt gratitude to **Dr. Jagadeesh H S**, Head of Electronics and Communication Engineering Department, Atria IT, Bangalore for being supportive and encouraging in completing the project.

We express our sincere gratitude to **Dr. Ramesh Nuthakki, Dr. Prasuna VNP,** Final Project coordinators, of Electronics and Communication Department, Atria Institute of Technology, for the support extended to us.

We express our deep sense of gratitude to our Guide **Prof. Kavitha S**, Assistant Professor, Dept. of Electronics and Communication Engineering, Atria IT, Bangalore for motivating and guiding me in successfully carrying out our project.

Finally, we also thank the entire staff of the Dept. of Electronics and Communication Engineering, friends and family for their valuable suggestions and encouragement in all means.

ABSTRACT

The premature spoilage of packaged food items poses significant health risks and contributes to an increasing number of consumer court cases, despite labeled expiry dates. Smart food packaging monitoring system is designed to predict the freshness of packaged food in real-time using an Internet of Things (IoT)-based architecture integrated with machine learning. The proposed system utilizes the ESP32 AI Thinker micro-controller interfaced with an MQ135 gas sensor and a DHT22 temperature and humidity sensor to acquire environmental parameters such as air quality, temperature, and humidity. These sensor readings are transmitted to an LCD for local display and to a Firebase Realtime Database for cloud storage. A K-Nearest Neighbors(KNN) machine learning model, trained on a dataset of 100 samples comprising temperature, humidity, and gas concentration (in ppm), is employed to classify the freshness status of the food as either "Fresh" or "Spoiled". Real-time sensor values are compared against this trained model, and the predicted outcome is updated in the Firebase database. Additionally, a user interface developed via MIT App Inventor fetches and displays the sensor data and freshness status on a mobile device, enabling remote food condition monitoring.The system offers a scalable, cost-effective solution to ensure food safety by detecting spoilage based on environmental conditions within packaging, thereby enhancing consumer protection and trust.

CONTENTS

List of Figures

Chapter – I

Introduction

Chapter 1

Introduction

The rapid growth of the food packaging industry, driven by urbanization and changing consumer lifestyles, has increased the need for advanced methods of monitoring food safety. Although expiry dates provide a general estimate of shelf life, they fail to account for real-time environmental fluctuations such as changes in temperature, humidity, and gas composition within the packaging. These factors can significantly impact the rate of food spoilage, especially in perishable products like dairy and meat. Consequently, reliance on static labeling can lead to premature spoilage, health risks, increased waste, and loss of consumer trust.

Smart packaging technologies have emerged as a promising solution to overcome these limitations by integrating sensing, communication, and data processing capabilities into conventional packaging systems. Among various approaches, the combination of Internet of Things (IoT) with machine learning (ML) has shown strong potential for enabling real-time monitoring and intelligent decision-making. IoT enables continuous acquisition of environmental data, while ML algorithms allow for pattern recognition and spoilage prediction based on historical and real-time inputs.

In this work, a smart packaging model is presented that utilizes an IoT-based sensor network to monitor critical environmental parameters, specifically temperature, humidity, and volatile gas concentration. These parameters are transmitted to a lightweight machine learning model—KNN Algorithm—trained to classify the freshness condition of the food product as either *Fresh* or *Spoiled*. The model has been developed using curated training data from controlled food spoilage scenarios and deployed on a Flask-based backend system that processes real-time sensor input and delivers classification results.

The results of the classification are stored in a cloud-based database and made accessible to users through a mobile interface. This layered approach allows for intelligent freshness monitoring that is scalable, low-cost, and adaptable for various

food products and storage environments. The integration of cloud connectivity and predictive analytics also supports remote access and timely decision-making, thereby enhancing food safety, minimizing waste, and supporting smart supply chain management.

1.1 Problem statement

The global food industry faces considerable challenges regarding the safety and shelf-life of packaged food products. Often, perishable goods deteriorate before their stated expiry dates due to suboptimal storage conditions such as excessive humidity, temperature fluctuations, or exposure to volatile gases. Traditional packaging provides no means of assessing the actual condition of the food, making printed expiry dates unreliable when environmental conditions deviate from the ideal.

This absence of intelligent, real-time monitoring can lead to undetected spoilage, foodborne illnesses, consumer dissatisfaction, and economic losses across the supply chain. To bridge this gap, an intelligent system is needed—one that continuously gathers and analyzes environmental data using embedded sensors and machine learning. Such a solution can alert users to freshness status based on real-time gas emissions and temperature-humidity patterns, ultimately improving food safety.

1.2 Scope

This project aims to design and implement a smart food packaging system that integrates Internet of Things (IoT) technology with machine learning (ML) algorithms to monitor and predict the spoilage status of perishable food items in real time. The focus is on dairy-based products, with paneer selected as the representative item for data collection and model training. The system captures critical environmental parameters—namely temperature, relative humidity, and gas concentration.

The scope of this project is limited to non-destructive environmental monitoring without the use of invasive sensors or chemical analysis. The implementation focuses on static test environments rather than dynamic retail or logistics settings, and the model is trained using a fixed dataset specific to paneer. Future scalability to other food items or dynamic supply chain environments is considered beyond the current scope but forms the basis for potential further research and development.

1.3 Objective

The key objectives of the project are:

- To develop a smart packaging prototype capable of real-time monitoring of environmental conditions such as temperature, humidity, and gas concentration.
- To collect and label a dataset of food spoilage scenarios under controlled conditions for training a machine learning model.
- To train and evaluate a KNN Algorithm for predicting the freshness state of food as *Fresh* or *Spoiled*.
- To deploy the trained model on a Firebase database and enable real-time prediction using sensor data sent from an IoT device.
- To display prediction results on a local LCD display for immediate user feedback.
- To integrate the system with a Firebase Realtime Database and a mobile application for remote data access and user notifications.
- To propose a scalable, low-cost solution that can be extended to various types of perishable food products.

Chapter – II

Literature Survey

Chapter 2

Literature Survey

[1] "Food storage monitoring system based on IoT" – 2024

The paper titled "Food Storage Monitoring System Based on IoT", authored by Noel John, Philipose Joseph, P S Anaz Muhammed, P Sidharth Sathiyandran, Dr. Godwinraj D, and Ranjitha Rajan, was published in 2024 in the IEEE Journal of International Journal of Pure and Applied Research in Engineering and Management Sciences (IJPREMS). This research introduces an IoT-based system designed to monitor and manage the storage conditions of food items, thereby ensuring their safety and quality. The system employs various sensors to track environmental factors such as temperature, humidity, and gas levels, which can influence the freshness and safety of food. By integrating these sensors with a microcontroller and a mobile application, the system allows for real-time monitoring and control, enabling users to receive alerts and make adjustments as needed. This approach aims to reduce food spoilage, minimize waste, and enhance food safety practices across various storage environments.

[2] "Monitoring Food Storage Humidity and Temperature Data Using IoT" - 2023

The paper titled "Monitoring Food Storage Humidity and Temperature Data Using IoT" by Asif Bin Karim, Md Zahid Hassan, Md Masum Akanda, and Avijit Mallik, published in 2023 in the *MOJ Food Processing & Technology* journal, presents an IoT-based solution for real-time monitoring of food storage conditions. The system employs a DHT-11 sensor to measure temperature and humidity levels, interfaced with an ESP8266 NodeMCU microcontroller to transmit data to the ThingSpeak cloud platform. This setup enables remote monitoring via a web interface, allowing users to track environmental parameters and ensure optimal storage conditions for perishable goods. The approach aims to reduce manual oversight and enhance food safety through automated data logging and analysis.

[3] "The Freshness of Food Detection using the Internet of Things and Android App" - 2022

The paper titled "The Freshness of Food Detection using the Internet of Things and Android App", authored by Prof. Smita Thakare, Payal Jadhav, Gauri Wagh, and Snehal Rajnor from Pune Vidyarthi Griha's College of Engineering, was published in 2022 in the *International Journal Of Advance Research And Innovative Ideas In Education* (IJARIIE). This research addresses the significant issue of food waste, which is a global concern with substantial environmental implications. The authors propose an IoT-based system designed to detect the freshness of food items by monitoring the volatile organic compounds (VOCs) emitted as food deteriorates. The system employs various sensors to capture these emissions, which are then analyzed using machine learning algorithms to assess food quality. An Android application is developed to display the freshness status of food items, providing users with real-time information to make informed decisions. This innovative approach aims to reduce food waste by enabling timely consumption or disposal of perishable goods.

[4] "The Design of Food Quality Supervision Platform Based on the Internet of Things"- 2021

The paper titled "The Design of Food Quality Supervision Platform Based on the Internet of Things" by Bing Jia and Yongjian Yang, presented at the 2021 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), introduces an IoT-based platform aimed at enhancing food quality supervision. The system integrates various IoT technologies to monitor and manage food quality throughout the supply chain, ensuring that safety and freshness standards are maintained. By leveraging real-time data collection and analysis, the platform enables proactive interventions and decision-making to address potential quality issues.This approach aligns with the broader trend of utilizing IoT solutions

to improve food safety and quality control, as evidenced by subsequent research and implementations in the field.

[5] "A Sensor System for Non-Destructive Monitoring of Food Ripening Processes" - 2020

The paper titled "A Sensor System for Non-Destructive Monitoring of Food Ripening Processes" by Alessandro Zompanti, Simone Grasso, Marco Santonico, and Giorgio Pennazza was presented at the 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT. This research introduces an innovative sensor system designed to monitor the ripening processes of food, particularly fruits, without causing any damage. The system utilizes advanced sensing technologies to detect and analyze the chemical and physical changes that occur as fruits ripen. By employing non-destructive methods, the system allows for continuous monitoring throughout the ripening process, providing valuable data that can be used to optimize harvesting times and improve food quality. This approach aligns with the principles of Industry 4.0, integrating IoT solutions to enhance food safety and quality control.

[6] "Overview of IoT MOX Chemical Sensors Arrays for Agri-Food Applications" - 2019

The paper titled "Overview of IoT MOX Chemical Sensors Arrays for Agri-Food Applications" by Marco Abbatangelo, Veronica Sberveglieri, Elisabetta Comini, and Giorgio Sberveglieri, presented at the 2019 IEEE International Symposium on Olfaction and Electronic Nose (ISOEN), explores the application of Metal Oxide Semiconductor (MOX) sensor arrays in the agri-food sector. The authors highlight the potential of these sensors to monitor various aspects of food quality and safety in real-time, thereby enhancing productivity and reducing food waste. The paper emphasizes four key areas where MOX sensors can be effectively utilized, By integrating these MOX sensor arrays into the food production and supply chain,

stakeholders can obtain real-time data, enabling prompt actions to address any deviations from desired food quality parameters. This approach not only ensures consumer safety but also supports sustainable practices in the agri-food.

[7] "IoT-Based Smart Food Monitoring System" – 2019

The paper titled "IoT-Based Smart Food Monitoring System" by M. Hasan et al., published in *IEEE Access* in 2019, presents an innovative solution to enhance food safety and quality control using Internet of Things (IoT) technology. The proposed system integrates various sensors to monitor critical parameters such as temperature, humidity, and gas levels in real-time, ensuring optimal storage conditions for perishable food items. By employing a microcontroller-based platform, the system collects and processes data from the sensors, providing actionable insights to prevent food spoilage and contamination. Additionally, the system features an alert mechanism that notifies users of any deviations from the desired conditions, enabling prompt corrective actions. This IoT-based approach offers a scalable and efficient solution for food monitoring, contributing to reduced food waste and improved public health outcomes.

[8] "A Cloud-Based Smart Expiry System Using QR Code"- 2018

The paper titled "A Cloud-Based Smart Expiry System Using QR Code" by Tareq Khan, presented at the 2018 IEEE International Conference on Electro/Information Technology (EIT), introduces an innovative solution to reduce household food waste by leveraging cloud computing and QR code technology. The system automates the tracking of food expiry dates, eliminating the need for manual entry by consumers.

In this system, during checkout, a customized smart-expiry card is scanned, or alternatively, a QR code is printed on the receipt. This action uploads a table containing product names and their corresponding expiry dates to the cloud. The consumer's smartphone app then downloads this table, automatically populating the device with the expiry information. Notifications are sent to the smartphone and an IoT-enabled display device attached to the refrigerator, alerting the user several days before a product expires. Additionally, the system integrates with voice assistants like Google Home, providing spoken reminders about expiring items when prompted. This approach not only simplifies the management of food expiry dates but also promotes timely consumption, thereby contributing to a reduction in food waste and enhancing food safety in households.

Chapter-III

Hardware and Software Used

Chapter 3

Hardware and Software Used

3.1 Hardware Requirements

3.1.1 Sensing module:

1. **Temperature sensor:**

Fig 3.1 DHT22 temperature sensor

The full form of **DHT22** is **Digital Humidity and Temperature sensor, model 22**. The DHT22 is a digital sensor that accurately measures both temperature and humidity. It is well-suited for applications where environmental monitoring is essential. Compared to its predecessor, the DHT11, the DHT22 offers higher precision, wider measurement ranges, and better long-term stability. These values show the temperature of a particular device. The applications of the DHT22 temperature sensor include industrial systems, consumer products, systems which are sensitive thermally, thermostatic controls, and thermometers. Temperature Range: -40°C to +80°C (±0.5°C accuracy).

2. **MQ-135 Gas Sensor**

Fig 3.2 MQ-135 Gas Sensor

MQ stands for "**Metal Oxide Semiconductor (MOS) Gas Sensor**" series. The MQ135 is a widely used air quality sensor capable of detecting a range of gaseous pollutants, including ammonia (NH_3), nitrogen oxides (NOx), alcohol, benzene, smoke, and carbon dioxide (CO_2). It's especially suitable for applications that involve monitoring air contamination and changes in air composition—key indicators of food spoilage. Detecting the release of volatile organic compounds (VOCs) and gases produced as food begins to decay. Measuring gas concentration levels inside the packaging in parts per million (ppm). Sensitive to a wide range of spoilage-related gases. Detection Range: 10 – 1000 ppm for gases like NH_3, NOx, alcohol, benzene, and CO_2.

3.1.2 Co-Ordinator Module:

1. **Microcontroller – ESP32 AI Thinker Module**

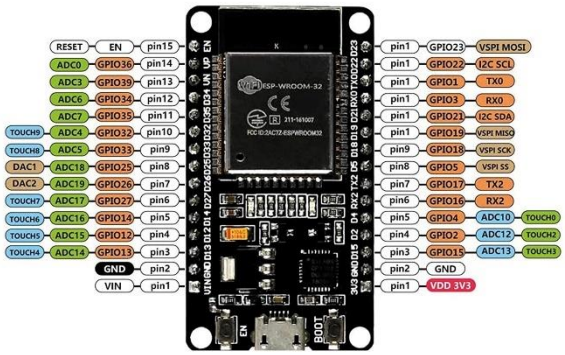

Fig 3.3 ESP32 AI Thinker Module

The ESP32 AI Thinker Module is a powerful microcontroller widely used in IoT applications due to its versatile features and low cost. It is powered by a dual-core 32-bit LX6 CPU and includes built-in 2.4 GHz Wi-Fi (802.11 b/g/n) and Bluetooth Low Energy (BLE) for seamless wireless connectivity. The module typically comes with 4MB of flash memory, making it suitable for data handling and firmware storage. It supports multiple communication protocols through its GPIO pins, including I2C, SPI, ADC, DAC, PWM, and UART, allowing integration with a

wide range of sensors and peripherals. Its low power consumption makes it ideal for battery-operated environments, and it also supports on-device machine learning using TinyML, enabling edge AI capabilities for intelligent local data processing.

2. LCD Display:

Fig 3.4 LCD 16X2 Parallel Display

The full form of **LCD** is **Liquid Crystal Display**. The 16x2 LCD Display is a simple, low-cost module that can display two lines of text with up to 16 characters per line. Based on the HD44780 controller, it is commonly used in embedded systems and Arduino-based projects due to its ease of interfacing and clear output. In this smart food packaging system, the 16x2 LCD serves as a local display unit, showing real-time sensor data such as temperature, humidity, gas concentration, and the predicted food freshness status ("Fresh" or "Spoiled"). This allows users or inspectors to quickly check the food condition without relying on the mobile app or cloud database. The display is connected to the ESP32 microcontroller via digital GPIO pins and operates using simple commands through either 4-bit or 8-bit parallel communication. Its inclusion enhances the usability of the system by offering immediate, readable feedback directly.

3. 5V Power Supply Module

Fig 3.5 5V Power Supply Module

The 5V Power Supply Module is a compact and essential component used to provide a stable 5-volt DC output required to operate various hardware components in embedded systems. In this smart food packaging project, the power supply module ensures reliable and consistent power to the ESP32 microcontroller, MQ135 gas sensor, DHT22 temperature and humidity sensor, and the 16x2 LCD display. Since most of these components operate within a 3.3V to 5V range, the module is crucial for preventing voltage fluctuations that could affect sensor readings or microcontroller performance. It can be powered by a USB source, batteries, or an adapter, and it often includes onboard voltage regulators to deliver clean, filtered output.

4. 5V-2A Power Adapter

Fig 3.6 5V-2A Power Adapter

A 5V 2A power adapter is a compact and efficient device designed to convert AC (Alternating Current) from a standard wall outlet into a stable 5V DC (Direct Current) output, providing up to 2 amperes of current. This makes it suitable for powering various low-power electronic devices such as CCTV cameras, wireless. When selecting a 5V 2A power adapter, it's essential to ensure that the output specifications match the requirements of your device to prevent potential damage/malfunction.

3.2 Software Requirements

1. Python IDLE

Fig 3.7 Python IDLE

Python IDLE (Integrated Development and Learning Environment) is the default editor that comes bundled with the standard Python distribution. It provides a simple and user-friendly interface for writing, editing, and running Python code. IDLE includes features such as a Python shell for interactive testing, a text editor with syntax highlighting, autocompletion, and debugging tools, making it suitable for beginners learning Python as well as for quick development tasks. It's lightweight, easy to install, and doesn't require additional setup, making it a convenient choice for getting started with Python programming.

2. Firebase Database

Fig 3.8 Firebase Console

Firebase, a product of Google, is a powerful platform that enables developers to build, manage, and scale applications with ease. It simplifies app development by offering a secure and efficient backend, eliminating the need for server-side programming. Firebase Database is a cloud-hosted NoSQL database provided by Google as part of the Firebase platform. It allows developers to store and sync data in real-time across all clients, making it ideal for building dynamic, collaborative applications such as chat apps, live feeds, or multiplayer games. Firebase offers two types of databases: the Realtime Database, which stores data as a large JSON tree, and Cloud Firestore, a more scalable and flexible option with structured collections and documents. Both databases support offline access, robust security rules, and seamless integration with other Firebase services.

3. MIT APP Inventor

Fig 3.9 MIT App Inventor

MIT App Inventor is a free, cloud-based platform developed by the Massachusetts Institute of Technology that allows users to create mobile applications for Android and iOS without needing advanced programming skills. It uses a visual, drag-and-drop interface where users can design the app's layout and program its functionality using block-based coding, making it especially popular for beginners, students, and educators. With App Inventor, users can access features like sensors, GPS, and web APIs, making it a powerful tool for learning app development and prototyping real-world solutions quickly and interactively.

4. KNN Algorithm for ML

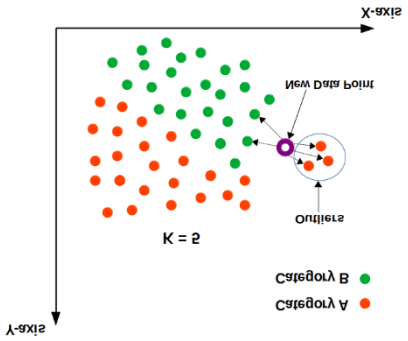

Fig 3.10 KNN Algorithm Graph

The **K-Nearest Neighbors (KNN)** algorithm is a simple and intuitive supervised machine learning method used for classification and regression tasks. It works on the principle that similar data points are close to each other. When predicting an unknown data point, KNN looks at the 'K' nearest neighbors (using a distance metric like Euclidean distance) and decides the output based on majority voting (for classification) or averaging (for regression). One of the main advantages of KNN is that it's easy to understand, requires no explicit training phase (lazy learner), and makes no assumptions about the underlying data distribution. However, it also has some disadvantages, such as being computationally expensive for large datasets.

Chapter – IV

Methodology

Chapter 4

Methodology

4.1 Block diagram

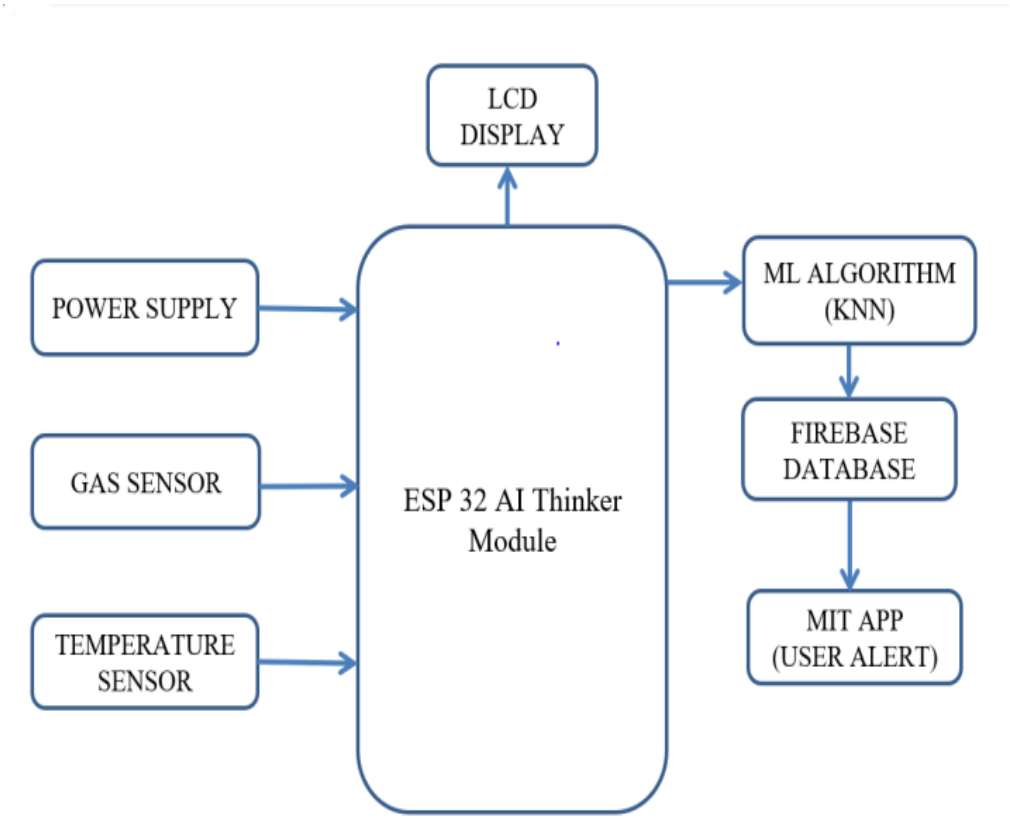

Fig 4.1 Block Diagram

The block diagram illustrates a smart food monitoring system using the ESP32 AI Thinker module. It is powered by a power supply and connected to a gas sensor and a temperature sensor to collect environmental data. This data is processed and sent to a K-Nearest Neighbors (KNN) machine learning algorithm to determine the freshness status of food. The results are uploaded to a Firebase database and simultaneously displayed on an LCD screen for local monitoring. Additionally, the data is sent to a mobile application developed using MIT App Inventor, which alerts users in real time about the food's condition.

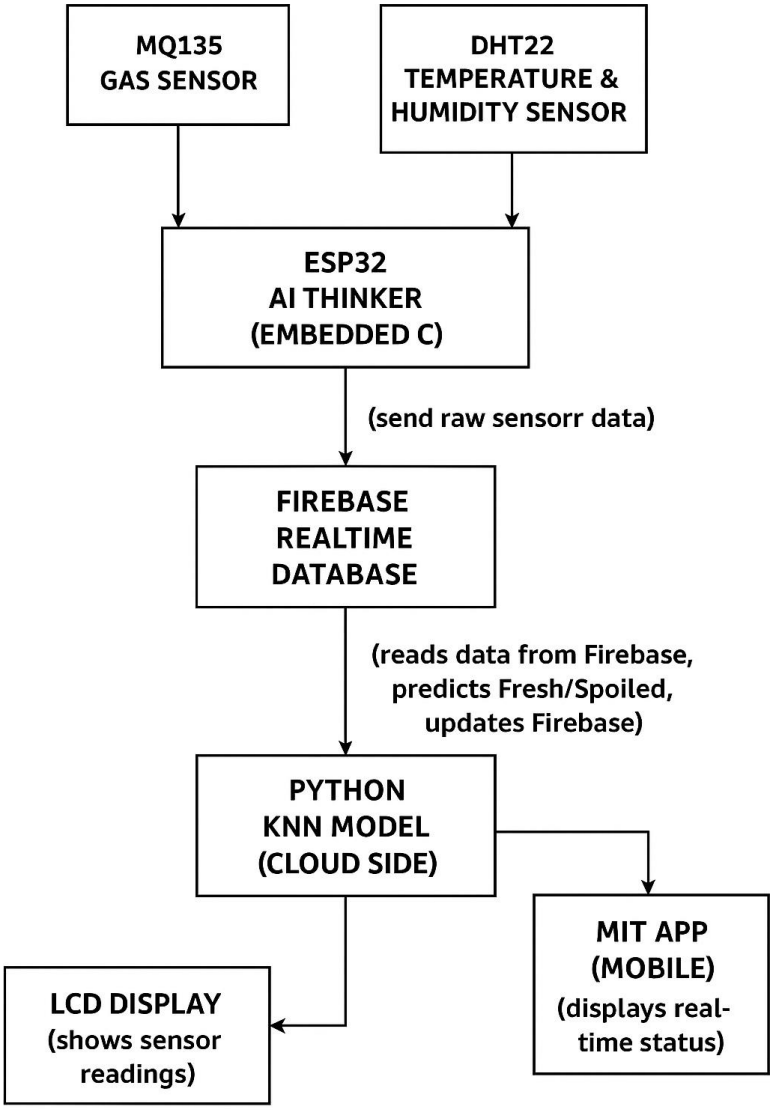

Fig 4.2 Flow Diagram

The tabular column given below gives the description of each block and the flow of the device working from the sensor connection to the deployment of the KNN Algorithm and sending the user alert briefly

Sl No.	BLOCK	DESCRIPTION
1	**Sensor Connection**	Connect MQ135 Gas Sensor and DHT22 Temperature + Humidity Sensor to the ESP32 board. MQ135 outputs analog gas level; DHT22 provides digital temperature and humidity values.
2	**ESP32 Setup**	Use Arduino IDE to program the ESP32. Initialize GPIO pins for each sensor, read sensor data, and convert raw readings into readable formats.
3	**Install Libraries**	Install required libraries in Arduino IDE: DHT.h for DHT22, WiFi.h for internet connection, Firebase_ESP_Client.h for Firebase communication.
4	**Connect to Firebase**	Use Firebase project credentials (URL & token) and WiFi details to establish cloud connectivity from ESP32. Ensure secure data transmission.
5	**Send Sensor Data**	ESP32 reads data at regular intervals and pushes formatted values (temperature, humidity, gas ppm) to Firebase Realtime Database using HTTP calls.
6	**KNN ML Cloud Script**	A Python script running on cloud/server/laptop fetches sensor values from Firebase, loads the trained KNN model using scikit-learn, and performs prediction.
7	**Update Prediction**	The script writes the result of the prediction ("Fresh" or "Spoiled") back to Firebase under a prediction_status field.
8	**Mobile App**	An MIT App Inventor mobile application reads real-time values from Firebase and displays sensor data and prediction status in the user interface.
9	**Alert System**	If prediction_status = "Spoiled", the app can trigger alerts like push notifications, sounds, or on-screen warnings to the user.
10	**Testing and Deployment**	Conduct experiments with paneer samples, validate freshness prediction accuracy, monitor Firebase response time, and ensure mobile app reliability during field deployment.

Table 1. Flow Description

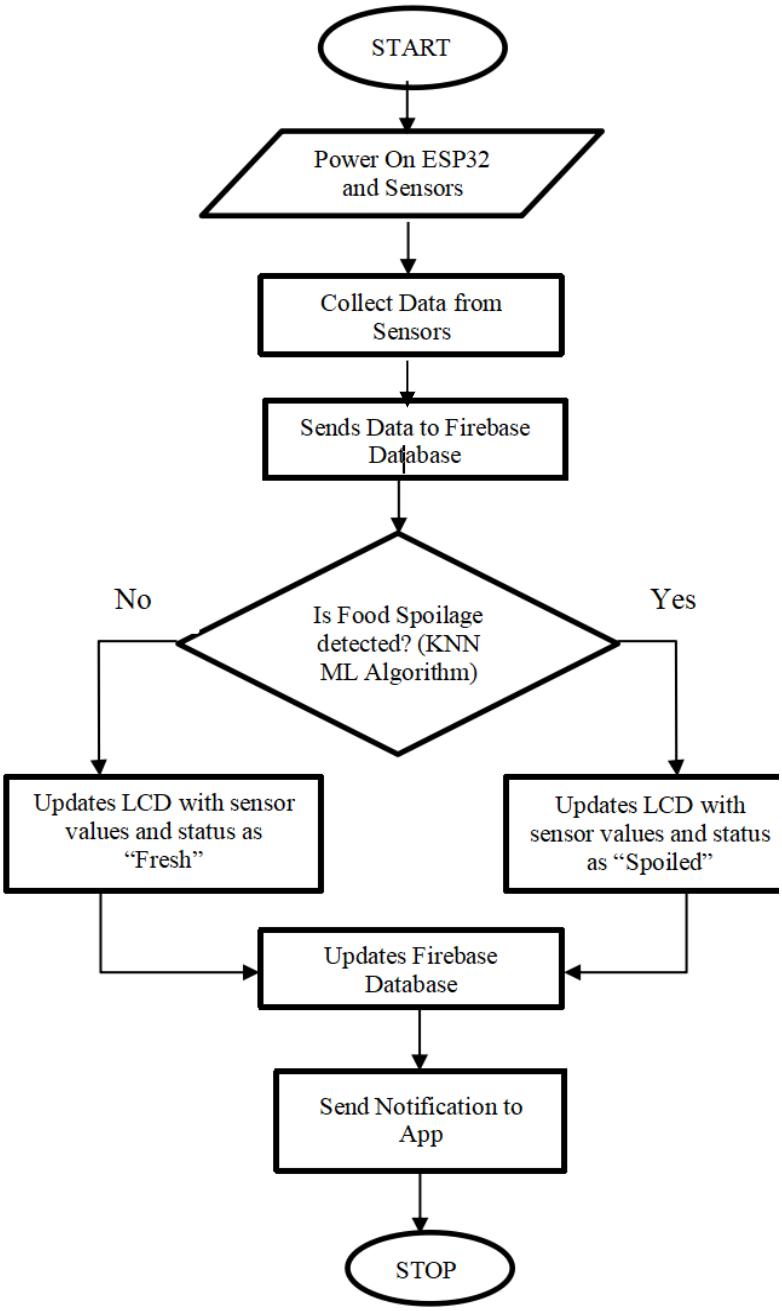

Fig 4.3 System Workflow

4.2 Implementation

4.2.1 Circuit Diagram

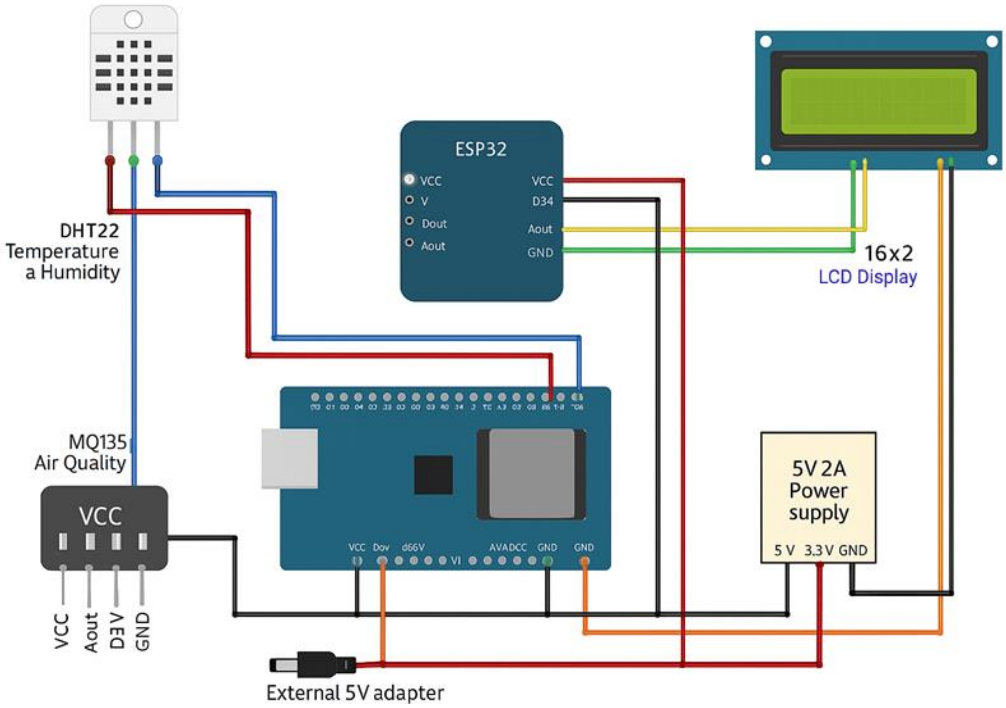

Fig 4.4 Circuit Diagram

- **Microcontroller:** ESP32 AI Thinker Module
- **Sensors:IM**
  - MQ135 Gas
  - SensorSEN0161 DHT22
  - Temperature Sensor
- **16x2 LCD Display with I2C Interface**
- **Power:**
  - DC 5V Power Module, 5V 2A Adapter
- **Other:**
  - Jumper wires for making connections

4.2.2 Hardware Setup

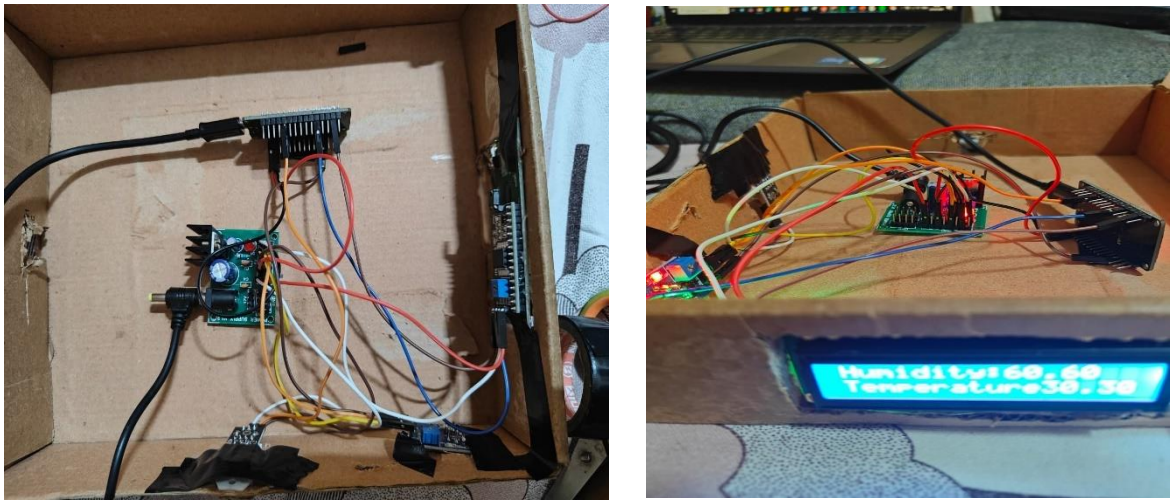

Fig 4.5 Hardware Connections

The core of the system hardware is built around the ESP32 AI Thinker module, interfacing with multiple sensors and a display for real-time monitoring. The hardware connections are as follows:

- **DHT22 Temperature and Humidity Sensor:**
  - Output pin connected to D4 of ESP32.
  - VCC and GND connected to a 5V Power Supply Module.
- **MQ135 Gas Sensor:**
  - Analog Output (A0) connected to D34 of ESP32.
  - VCC and GND connected to the same 5V Power Supply Module.
- **16x2 LCD Display with I2C Interface:**
  - SDA connected to D21 of ESP32.
  - SCL connected to D22 of ESP32.
  - VCC and GND connected to the 5V Power Supply Module.
- **Power Management**:

All sensor modules and the LCD are powered by a **5V Power Supply Module**, which is externally connected to a **5V 2A Adapter**. The ESP32 itself is powered via its **3.3V (3V3) pin** connected to the same power module's output, ensuring stable operation. All grounds (GND) are properly tied together to maintain a common reference.

4.2.3 Data Collection

➢ A custom dataset was created by monitoring paneer samples stored under various environmental conditions.

For each sample:

- Temperature (°C) [25°C -40°C]
- Humidity (%) [56%-80%]
- Gas concentration (ppm) [>400ppm, Spoiled]

were recorded using the connected sensors. The freshness of the food (Fresh or Spoiled) was manually labeled based on sensory evaluation (appearance, odor).

➢ In total, 100 labeled data samples were collected for training and validation

Temperature	Humidity (%)	MQ135 (ppm)	Result
34	60	401	Spoiled
35	61	489	Spoiled
42	62	467	Spoiled
36	61	431	Spoiled
37	70	489	Spoiled
30	59	450	Fresh
31	60	350	Fresh
34	70	433	Spoiled
34	65	487	Spoiled
45	63	430	Spoiled
27	59	341	Fresh
49	41	489	Spoiled
45	60	500	Spoiled
34	60	510	Spoiled
43	67	625	Spoiled
35	78	456	Spoiled
33	65	481	Spoiled
33	60	432	Spoiled
37	62	458	Spoiled
30	59	307	Fresh
39	60	421	Spoiled
40	65	450	Spoiled
30	60	376	Fresh
31	67	412	Spoiled
33	65	435	Spoiled
32	63	455	Spoiled
33	62	432	Spoiled
35	62	490	Spoiled
37	61	534	Spoiled
30	56	333	Fresh
30	59	543	Spoiled
33	67	498	Fresh
30	59	356	Fresh
30	60	420	Spoiled
31	61	505	Spoiled
34	68	516	Spoiled
32	59	377	Fresh

32	60	431	Spoiled
30	65	489	Spoiled
31	61	456	Spoiled
30	60	345	Fresh
38	60	433	Spoiled
33	60	481	Spoiled
31	60	423	Spoiled
30	59	456	Spoiled
30	59	412	Spoiled
29	60	345	Fresh
33	56	416	Spoiled
30	60	378	Fresh
31	61	451	Spoiled
30	62	370	Fresh
31	63	433	Spoiled
33	60	418	Spoiled
32	60	390	Fresh
32	60	321	Fresh
31	62	444	Spoiled
33	62	578	Spoiled
30	63	541	Spoiled
34	61	432	Spoiled
35	61	312	Fresh
32	67	423	Spoiled
32	60	345	Fresh
31	61	376	Fresh
30	60	562	Spoiled
30	61	543	Spoiled
30	65	578	Spoiled
39	61	419	Spoiled
38	62	426	Spoiled
30	62	389	Fresh
30	63	455	Spoiled
30	60	489	Spoiled
31	61	410	Spoiled
38	60	432	Spoiled
30	62	368	Fresh
29	61	333	Fresh
27	60	315	Fresh
26	62	388	Fresh
25	63	337	Fresh
29	61	310	Fresh
30	60	368	Fresh
30	76	411	Spoiled
33	68	430	Spoiled
31	61	422	Spoiled
32	60	444	Spoiled

Fig 4.6 100 Samples with 3 variables

4.2.4 Machine Learning Model (KNN)

The collected dataset was processed using Python and the **scikit-learn** library. A **K-Nearest Neighbors (KNN)** classification model was chosen because of its simplicity, robustness, and good performance for small datasets.

Steps followed:

- Data normalization was performed to bring all features to a similar scale.
- The dataset was split into training and testing sets (e.g., 80%-20% split).
- The KNN model was trained and tuned for optimal performance (e.g., selecting the best 'k' value).
- The trained model was serialized and made ready for real-time prediction.

The KNN model was deployed to work with cloud data, not directly on the ESP32.

```
import os
os.system('cls')
import json, requests, time
import pyrebase
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

config = {
  "apiKey": "DoVgdZGYciUs0fdPVWDc10LNIkWxb3iY7S7oNfND",
  "authDomain": "smartpackage-f3896-default-rtdb.firebaseio.com",
  "databaseURL": "https://smartpackage-f3896-default-rtdb.firebaseio.com",
  "storageBucket": "smartpackage-f3896-default"
}

firebase = pyrebase.initialize_app(config)
db = firebase.database()
print(db)

ad=pd.read_csv("./dataset.csv")
ad.shape
ad.info()
ad.describe()
ad.head()
print(ad)
y = ad['Result'].values
X = ad.drop('Result', axis=1).values
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X,y)
y_pred = knn.predict(X)

while True:

    temperature = db.child("Temperature").get()
    a=temperature.val()
    print(a)

    humidity = db.child("Humidity").get()
    b=humidity.val()
    print(b)

    mq135 = db.child("Air:").get()
    c=mq135.val()
    print(c)

    #x_new = [int(temperature),int(humidity),int(moist),int(smoke)]
    x_new = [a,b,c]
    new_pred = knn.predict([x_new])
    type(new_pred)
    print("Prediction : {}".format(new_pred))
```

```
"IDLE Shell 3.8.10"
File Edit Shell Debug Options Window Help
62.799999
444.0
Prediction : ['Spoiled']

====== RESTART: C:\Users\gayat\OneDrive\Desktop\project\smartpackaging.py ======
<pyrebase.pyrebase.Database object at 0x0000014227E329D0>
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Temperature (?)  99 non-null     int64
 1   Humidity (%)     99 non-null     int64
 2   MQ135 (ppm)      99 non-null     int64
 3   Result           99 non-null     object
dtypes: int64(3), object(1)
memory usage: 3.2+ KB
    Temperature (?)  Humidity (%)  MQ135 (ppm)   Result
0                34            60          401  Spoiled
1                35            61          489  Spoiled
2                42            62          467  Spoiled
3                36            61          431  Spoiled
4                37            70          489  Spoiled
..              ...           ...          ...      ...
94               30            60          368    Fresh
95               30            76          411  Spoiled
96               33            68          430  Spoiled
97               31            61          422  Spoiled
98               32            60          444  Spoiled

[99 rows x 4 columns]
30.5
62.799999
448.0
Prediction : ['Spoiled']
30.6
62.799999
430.0
Prediction : ['Spoiled']
```

Fig 4.7 ML Training & Prediction using KNN Algorithm

4.2.5 Cloud Integration (Firebase)

➢ The ESP32 is programmed to:

- Read real-time sensor data.
- Push the temperature, humidity, and gas readings into the **Firebase Realtime Database** at regular intervals.

➢ On the cloud side:

- A **Python script** continuously monitors the Firebase database for new entries.
- When new sensor data is detected, the script fetches it, runs it through the trained KNN model, and predicts whether the food is "Fresh" or "Spoiled."
- The prediction result is updated back into the Firebase database under a separate field (prediction_status).

  Thus, both sensor readings and freshness predictions are available on the cloud in real time.

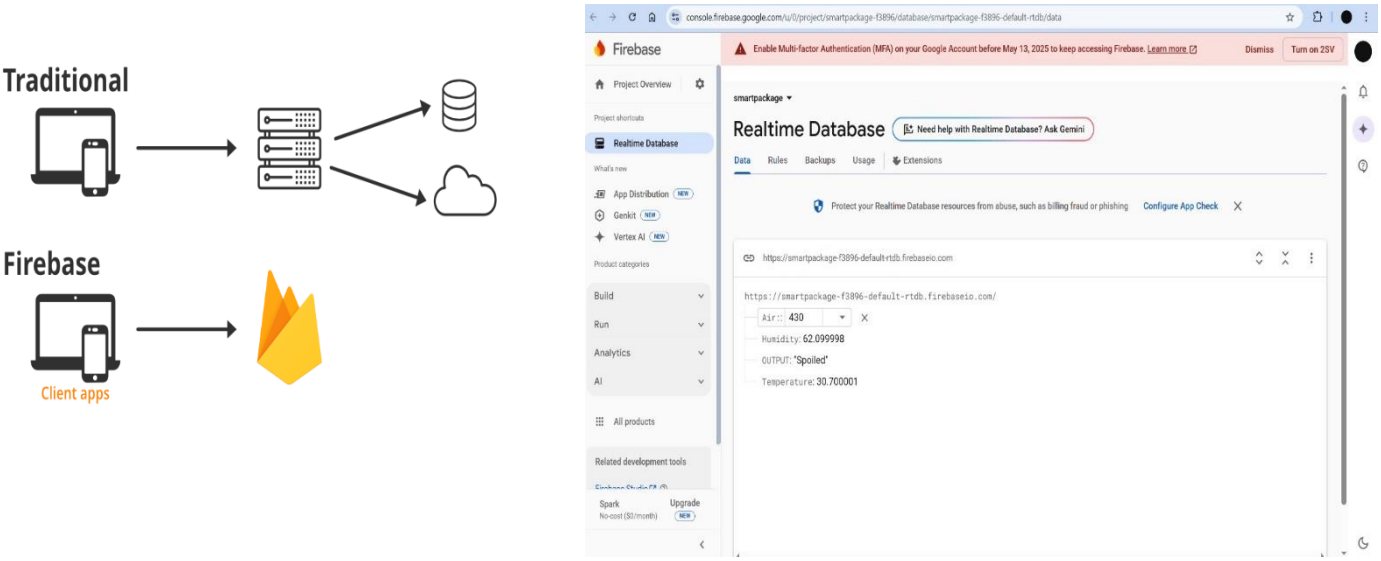

Fig 4.8 Firebase Database with Updates

4.2.6 Mobile Application (MIT App Inventor)

A mobile application was developed using MIT App Inventor. The app functionalities include:

- Fetching the latest sensor data (temperature, humidity, gas concentration) from Firebase.
- Displaying the real-time freshness status ("Fresh" or "Spoiled") based on the KNN model prediction.
- Providing a simple dashboard for users to monitor food quality remotely.
- Future expansion possibilities for real-time notifications or alerts when spoilage is detected.
- This ensures end-users have anytime, anywhere access to the food packaging status.

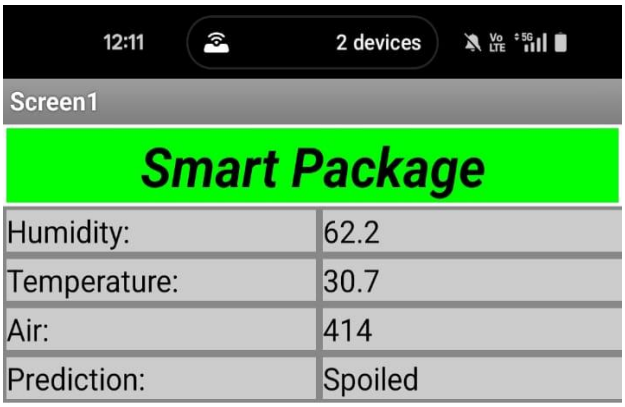

Fig 4.9 MIT App with 1 SCREEN

Chapter V

Result

Chapter 5

Result

- Initially we calibrated the sensors and made the connection as per the circuit diagram and then collected the values by placing the sensors in different types of packaged food. The different sample values of temperature, humidity and air quality is sent to the firebase database through the ESP32 AI thinker module microcontroller.

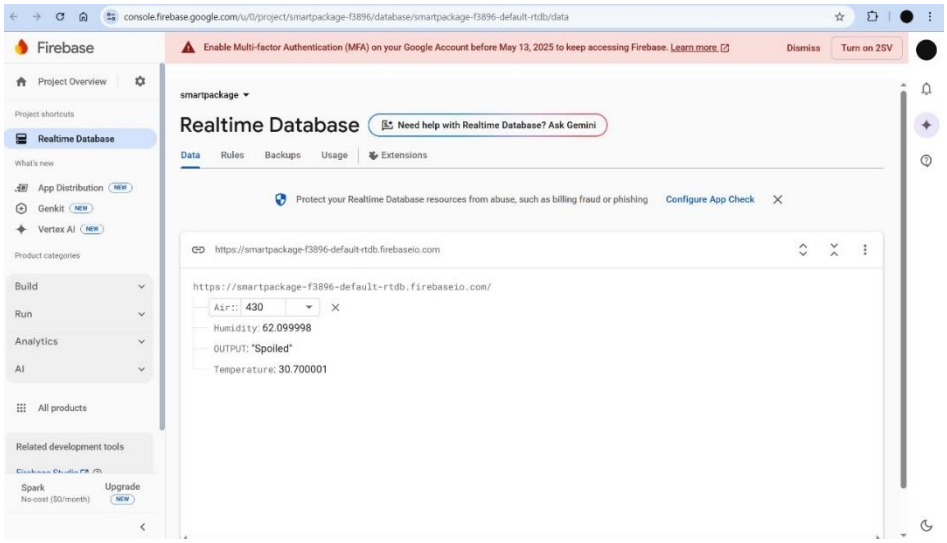

Fig 5.1 Prediction using Sample Values

- After training the ML using KNN Algorithm for different sample values, these sample values are again trained for prediction. This ML is then deployed into the firebase database. After successful prediction if the food is Fresh/Spoiled. This is applied to real-time gas and temperature sensor values. For this the ML had to be trained with a dataset, we selected 100 samples for 3 variables- temperature [25°C -40°C], humidity [56%-80%], air quality**)** [>400 ppm, spoiled] and the result- Fresh/Spoiled.

Temperature	Humidity (%)	MQ135 (ppm)	Result
34	60	401	Spoiled
35	61	489	Spoiled
42	62	467	Spoiled
36	61	431	Spoiled
37	70	489	Spoiled
30	59	450	Fresh
31	60	350	Fresh
34	70	433	Spoiled
34	65	487	Spoiled
45	63	430	Spoiled
27	59	341	Fresh
49	41	489	Spoiled
45	60	500	Spoiled
34	60	510	Spoiled
43	67	625	Spoiled
35	78	456	Spoiled
33	65	481	Spoiled
33	60	432	Spoiled
37	62	458	Spoiled
30	59	307	Fresh
39	60	421	Spoiled
40	65	450	Spoiled
30	60	376	Fresh
31	67	412	Spoiled
33	65	435	Spoiled
32	63	455	Spoiled
33	62	432	Spoiled
35	62	490	Spoiled
37	61	534	Spoiled
30	56	333	Fresh
30	59	543	Spoiled
33	67	498	Fresh
30	59	356	Fresh
30	60	420	Spoiled
31	61	505	Spoiled
34	68	516	Spoiled
32	59	377	Fresh
32	60	431	Spoiled
30	65	489	Spoiled
31	61	456	Spoiled
30	60	345	Fresh
38	60	433	Spoiled
33	60	481	Spoiled
31	60	423	Spoiled
30	59	456	Spoiled
30	59	412	Spoiled
29	60	345	Fresh
33	56	416	Spoiled
30	60	378	Fresh
31	61	451	Spoiled
30	62	370	Fresh
31	63	433	Spoiled
33	60	418	Spoiled
32	60	390	Fresh
32	60	321	Fresh
31	62	444	Spoiled
33	62	578	Spoiled
30	63	541	Spoiled
34	61	432	Spoiled
35	61	312	Fresh
32	67	423	Spoiled
32	60	345	Fresh
31	61	376	Fresh
30	60	562	Spoiled
30	61	543	Spoiled
30	65	578	Spoiled
39	61	419	Spoiled
38	62	426	Spoiled
30	62	389	Fresh
30	63	455	Spoiled
30	60	489	Spoiled
31	61	410	Spoiled
38	60	432	Spoiled
30	62	368	Fresh
29	61	333	Fresh
27	60	315	Fresh
26	62	388	Fresh
25	63	337	Fresh
29	61	310	Fresh
30	60	368	Fresh
30	76	411	Spoiled
33	68	430	Spoiled
31	61	422	Spoiled
32	60	444	Spoiled

Fig 5.2 100 Sample data

- Real-time values from different sensors are taken and sent to the firebase database and also displays it on the LCD display. For the below image there is no input given for the sensors, thus its showing the normal optimum variable values.

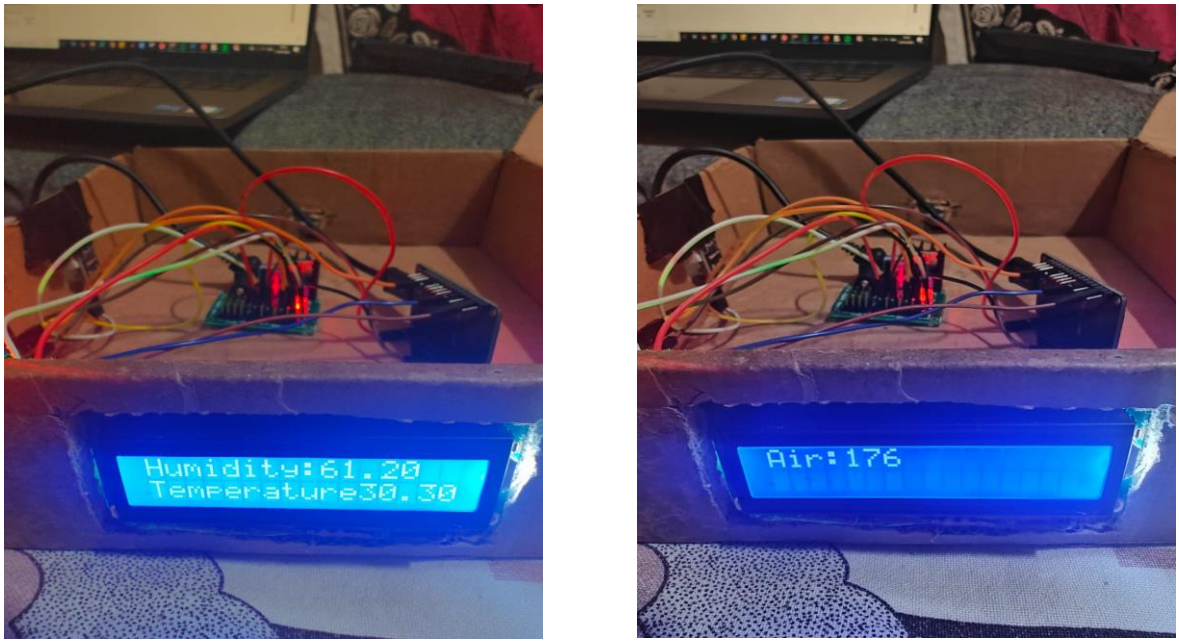

Fig 5.3 LCD Display without any input packaged food

- Now Considering Paneer as the input spoiled packaged food, it shows the following readings for the sensor in python console.

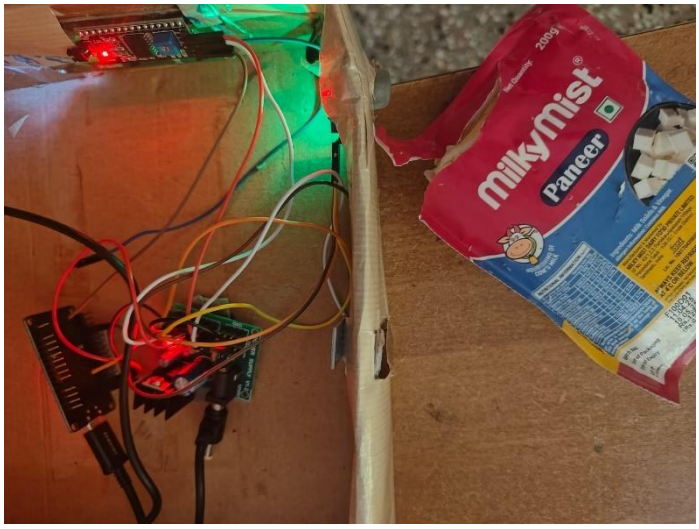

Fig 5.4 Spoiled Paneer as input for Packaged food

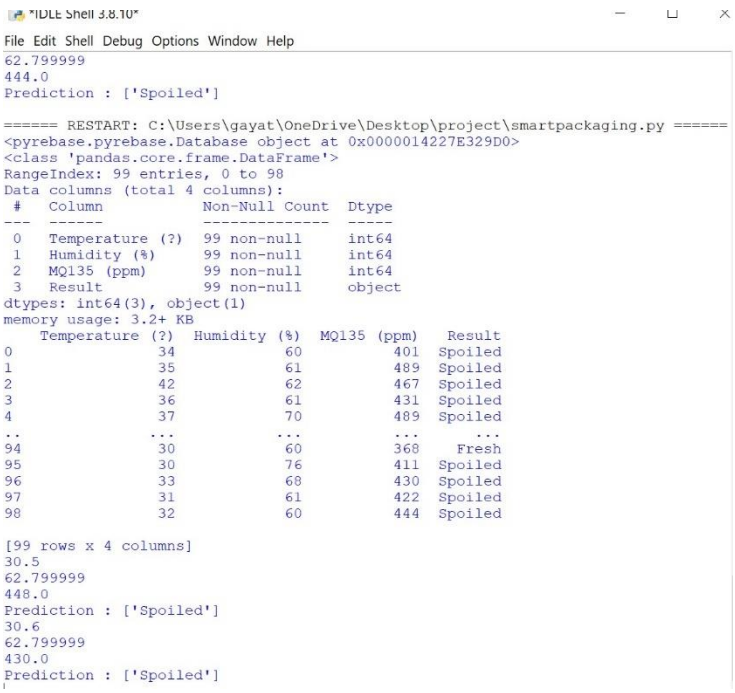

Fig 5.5 Status of Spoiled Paneer on Python Console

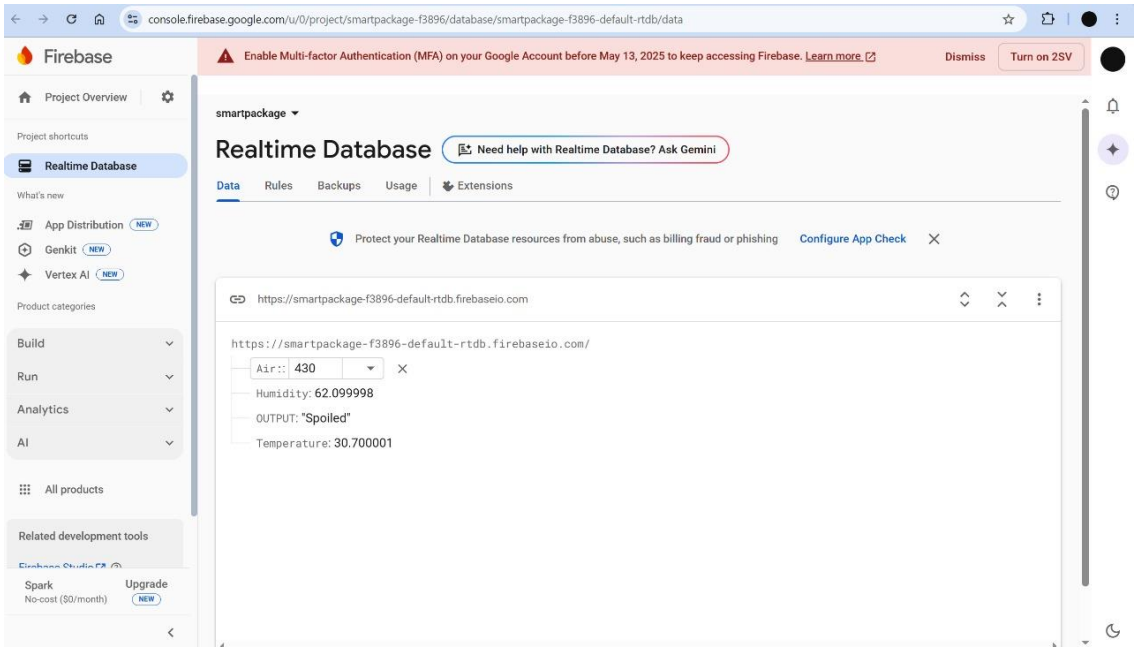

Fig 5.6 Status of Spoiled Paneer on Firebase

As the database gets updated with the real time values of the sensors, and according to the values the ML predicts if the food is spoiled or fresh, depending on a particular threshold values of the three variables. This status too is updated again in the database. The database is integrated with the MIT App, which will in-turn send timely alerts to the user.

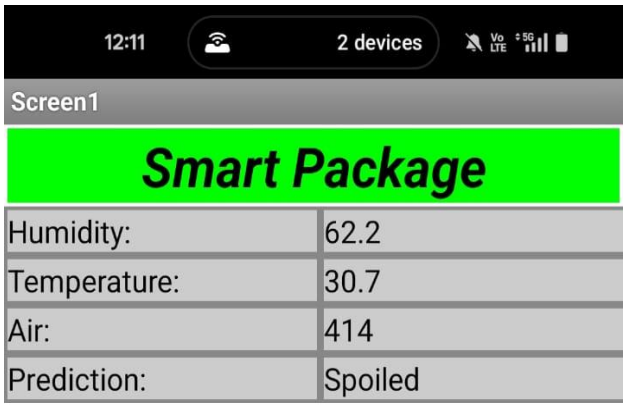

Fig 5.7 Live Status of Spoiled Paneer on MIT App

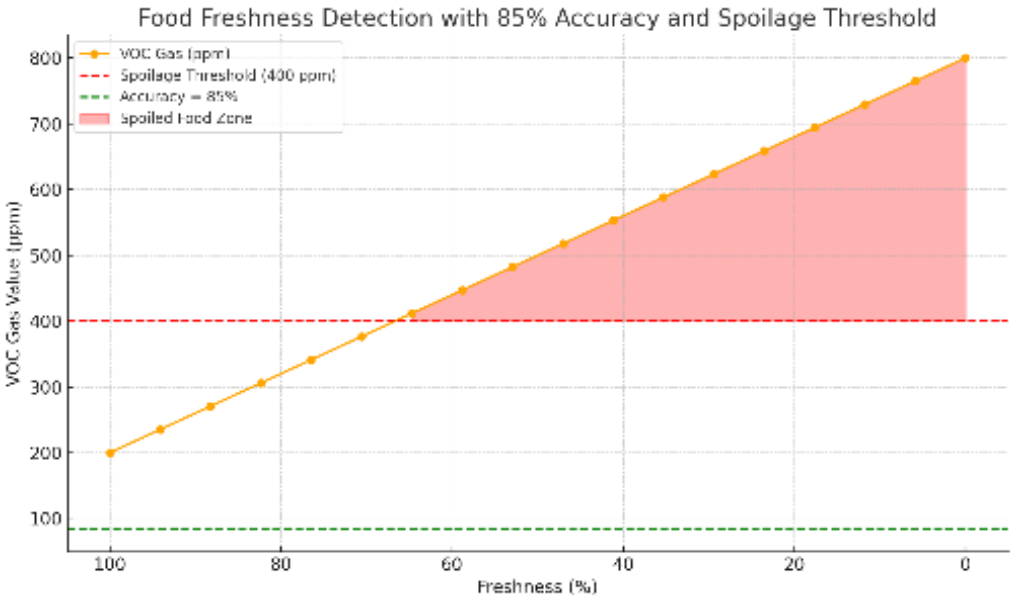

Fig.5.8 Freshness Vs VOC Gas values with Accuracy

Chapter VI

Applications, Advantages & Disadvantages

Chapter 6

6.1 Applications

The applications of Smart Grocery Packaging for Expiry Detection are diverse and impactful:

- **Household Food Monitoring**: Helps families monitor the freshness of stored food in real-time, reducing health risks and minimizing food waste.
- **Retail & Supermarkets**: Enables grocery stores to track the freshness of perishable items, ensuring timely removal of spoiled goods and improving customer trust.
- **Cold Chain Logistics**: Monitors environmental conditions during food transportation and storage, ensuring quality is maintained throughout the supply chain.
- **Restaurants & Cafeterias**: Assists kitchen staff in checking food safety before preparation and serving, promoting hygiene and compliance with food safety standards.
- **Food Packaging Industry**: Adds value to traditional packaging by embedding smart systems that enhance product quality assurance and traceability.
- **Food Donation Centers**: Ensures donated food is safe and consumable, boosting the efficiency and safety of food redistribution efforts.
- **Export & Import Industry**: Monitors exported food items over long transit times to prevent spoilage and ensure compliance with international food safety standards.
- **Research and Development**: Provides data for research into food spoilage patterns, preservation techniques, and sensor-based prediction models.
- **Health and Safety Compliance**: Assists regulatory bodies and food safety inspectors by providing real-time freshness data for compliance checks.

6.2 Advantages

The advantages of Smart Grocery Packaging includes:

- **Real-Time Monitoring**: Continuously tracks food conditions such as gas emissions, temperature, and humidity for up-to-date freshness assessment.
- **Early Spoilage Detection**: Predicts potential spoilage before it becomes visible, helping to prevent health risks and reduce food waste.
- **User Alerts**: Sends push notifications or SMS alerts to inform users when food is close to expiring or has spoiled.
- **Cloud Storage & Remote Access**: Stores sensor data on Firebase, allowing users to access information anytime from anywhere via a mobile app.
- **Cost-Effective**: Utilizes low-cost hardware components like ESP32 and MQ135, making the system affordable for both households and commercial use.
- **Scalable and Customizable**: Can be adapted for various food products and expanded with additional sensors for enhanced detection accuracy.
- **Improved Consumer Trust**: Ensures product quality and safety, fostering customer confidence in packaged food items.
- **Energy Efficient**: ESP32's low power consumption makes it suitable for long-term deployment in packaging environments.
- **Ease of Installation**: Wireless sensors are easier to install than wired systems, reducing installation time and costs.
- **Data Accessibility**: Enables easy access to monitoring data remotely through cloud-based platforms or IoT applications.
- **Environmental Impact:** Helps in preserving the environment by monitoring and maintaining the quality of water resources.
- **Customizable Solutions**: Offers flexibility in designing customized monitoring systems tailored to specific requirements and applications.

6.3 Disadvantages

While Smart Grocery Packaging offers many advantages, there are also some limitations to consider:

- **Limited Dataset Size**: Initial ML model is trained on a small sample size (100 samples), which may affect prediction accuracy in diverse conditions.
- **Internet Dependency**: Requires stable internet connectivity for real-time data synchronization with Firebase and remote access.
- **Sensor Limitations**: MQ135 and DHT22 have detection ranges that might not be suitable for all types of food or packaging environments.
- **Power Supply Constraints**: Continuous operation depends on reliable power sources, which can be a challenge for mobile or off-grid use cases.
- **Maintenance and Calibration**: Sensors may require periodic calibration and maintenance to ensure accurate readings over time.
- **Data Security Risks**: As with all IoT systems, potential vulnerabilities exist in transmitting and storing sensitive data.

- **Limited Edge Intelligence**: Current model offloads predictions to a server; onboard (edge) processing is not yet implemented on the ESP32.

Chapter VII

Conclusion and future scope

Chapter 7

Conclusion

The increasing global demand for safe, reliable, and long-lasting packaged food has intensified the need for advanced monitoring systems that go beyond traditional expiry labels. This project presents a comprehensive solution in the form of a smart food packaging system that harnesses the power of the Internet of Things (IoT) and machine learning (ML) to monitor, analyze, and predict food freshness in real time.

This innovative system offers numerous advantages over conventional food safety methods. It enhances consumer confidence by delivering transparent, real-time information about the freshness of their food. It also addresses a major public health concern by minimizing the risk of foodborne illnesses caused by spoiled food. Additionally, by enabling early detection of spoilage, the system helps reduce food wastage, which is a growing global issue with significant environmental and economic implications.

The proposed solution is not only cost-effective and energy-efficient but also modular and scalable. It has the potential to be adopted across various segments of the food industry—from household refrigerators to large-scale food processing and supply chain logistics. The architecture's flexibility allows it to be tailored to different food products and packaging environments, making it a practical tool for modern food safety management.

In conclusion, this smart packaging system represents a significant step toward intelligent, connected food monitoring solutions. It demonstrates how the convergence of IoT and machine learning can transform traditional practices into smart, proactive systems that align with the goals of sustainability, safety, and consumer empowerment in the food industry.

.

Future Scope

To further improve the intelligence, accuracy, and autonomy of the smart food packaging system, several advanced features and technical enhancements can be incorporated:

1. **On-Device Machine Learning (Edge AI):**
   Implementing TinyML models on the ESP32 would enable local, real-time prediction without relying on external servers. This reduces latency, enhances data privacy, and ensures operation even without internet connectivity.
2. **Multi-Class Spoilage Classification:**
   Instead of a binary "Fresh" or "Spoiled" label, future models can include multiple spoilage stages (e.g., *Fresh*, *Nearing Spoilage*, *Spoiled*) using probabilistic outputs or multi-class classifiers for more nuanced decision-making.
3. **Sensor Fusion:**
   Integrating additional sensors like:

- **CO_2/ethylene sensors** for fruit and vegetable spoilage,
- **pH sensors** for liquids or dairy,
- **Optical sensors** (color change detection), would allow cross-verification of spoilage indicators for improved reliability.

4. **Adaptive Learning Models:**
   Implement online or semi-supervised learning so the model can adapt over time with new data from different environments, packaging types, and food categories, improving its robustness and generalization.
5. **Self-Calibrating Sensors:**
   Incorporate smart calibration routines for gas and humidity sensors to counter drift over time, improving long-term accuracy without manual intervention.

References

References

[1] "Food storage monitoring system based on IoT", Noel John, Philipose Joseph, P S Anaz Muhammed, P Sidharth Sathiyandran, Dr. Godwinraj D, Ranjitha Rajan, IJPREMS, Jyothi College of Engineering, IEEE Journal, 2024.

[2] "The Freshness of Food Detection using the Internet of Things and Android App", Prof. Smita Thakare, Payal Jadhav, Gauri Wagh, Snehal Rajnor, Vidyarthi Griha's College of Engineering, IJARIIE-ISSN, 2022.

[3] "A Cloud-Based Smart Expiry System Using QR Code", Tareq Khan, School of Engineering Technology, Eastern Michigan University, IEEE International Conference on Electro/Information Technology (EIT), 2018.

[4] Monitoring Food Storage Humidity And Temperature Data Using Iot By Asif Bin Karim, Md Zahid Hassan, Md Masum Akanda, Avijit Mallik.

[5] Bing Jia and Yongjian Yang, "The design of food quality supervision platform based on the Internet of Things," Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE),2011,pp.263-266,doi 10.1109/TMEE.2011.6199193.

[6] M. Hasan et al., "IoT-Based Smart Food Monitoring System," *IEEE Access*, vol. 7, pp. 54775–54786, 2019.

[7] Zompanti, S. Grasso, M. Santonico and G. Pennazza, "A Sensor System for Non-Destructive Monitoring of Food Ripening Processes," 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, 2020.

[8] M. Abbatangelo, E. N. Carmona, V. Sberveglieri, E. Comini and G. Sberveglieri, "Overview of Iot Mox Chemical Sensors Arrays for Agri-Food Applications," 2019 IEEE International Symposium on Olfaction and Electronic Nose (ISOEN), 2019.

Appendix

ESP32 AI Thinker code:

```
// Include necessary libraries
#include <DHT.h>                    // Library for DHT sensor
#include <IOXhop_FirebaseESP32.h>   // Library for Firebase ESP32
#include <LiquidCrystal_I2C_Hangul.h>  // Library for LCD display over I2C
#include <Wire.h>                   // Library for I2C communication

// Initialize the LCD with I2C address 0x27, 16 columns, and 2 rows
LiquidCrystal_I2C_Hangul lcd(0x27, 16, 2);

// Define sensor pins
#define DHTPIN 4        // Digital pin connected to DHT22 data pin
#define MQ135_PIN 34    // Analog pin connected to MQ-135 sensor

// Define sensor type (DHT22 in this case)
#define DHTTYPE DHT22

// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE);

// Define Firebase project credentials
#define FIREBASE_HOST "smartpackage-f3896-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "DoVgdZGYciUs0fdPVWDc10LNIkWxb3iY7S7oNfND"

// Define Wi-Fi credentials
#define WIFI_SSID "Atria1"
#define WIFI_PASSWORD "Password06"

void setup() {
  Serial.begin(115200);       // Start serial communication at 115200 baud rate
  Serial.print("WELCOME");

  Serial2.begin(9600);        // Start secondary serial communication (optional, depending on use)

  dht.begin();              // Start DHT sensor
  lcd.init();               // Initialize the LCD
  lcd.backlight();          // Turn on LCD backlight
  lcd.print("Welcome");      // Display welcome message
  delay(2000);              // Wait for 2 seconds

  // Connect to Wi-Fi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to ");
```

```
  Serial.print(WIFI_SSID);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Connected to ");
  Serial.println(WIFI_SSID);
  Serial.print("IP Address is : ");
  Serial.println(WiFi.localIP()); // Display local IP address

  // Connect to Firebase
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  delay(500);
}

// Function to read air quality using MQ135 sensor
void air() {
  int sensorValue = analogRead(MQ135_PIN);  // Read analog value (0-4095)

  // Convert raw sensor value to voltage (0-3.3V for ESP32)
  float voltage = sensorValue * (3.3 / 4095.0);

  // Display air quality reading on Serial Monitor
  Serial.print("Analog Value: ");
  Serial.print(sensorValue);

  // Display air quality reading on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Air:");
  lcd.print(sensorValue);

  // Send air quality data to Firebase
  Firebase.setFloat("Air:", sensorValue);
  delay(200);

  // Display voltage on Serial Monitor
  Serial.print(" | Voltage: ");
  Serial.print(voltage);
  Serial.println(" V");

  delay(1000); // Wait for 1 second before next reading
}

// Function to read temperature and humidity from DHT22 sensor
```

```
void dht22() {
  delay(2000); // Wait for sensor to stabilize

  float temp = dht.readTemperature(); // Read temperature in Celsius
  float humidity = dht.readHumidity(); // Read humidity in %

  // Check if reading failed
  if (isnan(temp) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Display humidity and temperature on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Humidity:");
  lcd.print(humidity);
  lcd.setCursor(0, 1);
  lcd.print("Temperature");
  lcd.print(temp);

  // Display humidity and temperature on Serial Monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println(" °C");

  // Send humidity and temperature data to Firebase
  Firebase.setFloat("Humidity", humidity);
  delay(200);
  Firebase.setFloat("Temperature", temp);
  delay(200);
}

// Main loop to continuously read and send sensor data
void loop() {
  air();    // Read and send air quality data
  dht22();  // Read and send temperature and humidity data
}
```

Python Machine Learning Code (KNN Algorithm):

```
# Clear the console screen
import os
os.system('cls')

# Import necessary libraries
import json, requests, time
import pyrebase
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsClassifier

# Firebase configuration
config = {
   "apiKey": "DoVgdZGYciUs0fdPVWDc10LNIkWxb3iY7S7oNfND",
   "authDomain": "smartpackage-f3896-default-rtdb.firebaseio.com",
   "databaseURL": "https://smartpackage-f3896-default-rtdb.firebaseio.com",
   "storageBucket": "smartpackage-f3896-default"
}

# Initialize Firebase connection
firebase = pyrebase.initialize_app(config)
db = firebase.database()
print(db)  # Print database object to confirm connection

# Load dataset
ad = pd.read_csv("./dataset.csv")

# Display dataset information
ad.shape      # Check the shape (rows, columns) of dataset
ad.info()     # Display datatype information
ad.describe() # Statistical summary
ad.head()     # Display first 5 rows
print(ad)     # Print entire dataset (optional)

# Prepare features and labels
y = ad['Result'].values              # Output labels (Fresh/Spoiled)
X = ad.drop('Result', axis=1).values   # Input features (Temperature, Humidity, Gas)
```

```
# Train KNN Model
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X, y)                    # Train the KNN model on the dataset

# Test KNN model prediction on training data (optional)
y_pred = knn.predict(X)

# Infinite loop to fetch real-time data from Firebase, predict, and update output
while True:
    # Read Temperature from Firebase
    temperature = db.child("Temperature").get()
    a = temperature.val()
    print(a)

    # Read Humidity from Firebase
    humidity = db.child("Humidity").get()
    b = humidity.val()
    print(b)

    # Read Gas concentration (Air quality) from Firebase
    mq135 = db.child("Air:").get()
    c = mq135.val()
    print(c)

    # Prepare a new feature vector for prediction
    x_new = [a, b, c]

    # Predict freshness using the trained KNN model
    new_pred = knn.predict([x_new])
    type(new_pred)  # Checking type (optional)

    # Print prediction result
    print("Prediction : {}".format(new_pred))

    # Update the prediction output to Firebase
    db.update({"OUTPUT": new_pred[0]})

    # Wait for 5 seconds before repeating the cycle
    time.sleep(5)
```