International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

Real-Time Vegetable Storage and Monitoring System with User Alerts for a Smart Kitchen

N. Kavitha[1], Petikam Anisha Dev[2], Mamidanna Naga Sai Akhil[3]
[1]Assistant Professor, Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India
[2,3]Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India

I. ABSTRACT
Vegetable spoilage after harvest leads to big financial losses and nutritional waste in both home and business storage areas. The main problem is not that people don't know about it; it's that there isn't a continuous, objective, and cheap way to keep an eye on stored produce and let users know before damage becomes permanent. This paper discusses the design, implementation, and experimental validation of an intelligent IoT-based system for monitoring vegetable freshness and inventory, which is based on the ESP32 microcontroller. The system has a DHT11 sensor that measures the temperature and humidity in the air, a MQ-4 analogue gas sensor that finds methane and decomposition gas, a HX711 24-bit ADC with a strain-gauge load cell that measures weight, and an SSD1306 OLED display that shows the results in real time. The produce is stored on four racks that are set up to show how long it will last, from 1 to 3 days to 15 or more days. This makes it possible to keep track of when it will go bad automatically. The Telegram Bot API is an event-driven way to interact with users, enabling users to control their inventory using a conversational interface without necessarily downloading a special mobile application. All of the sensor data is sent at the same time to the Thinger.io IoT cloud platform, where it can be seen on a remote dashboard. The experiments showed that the system could reliably detect weight changes down to 70 grams, accurately classify gas-threshold freshness with transition-based spoilage alerts, manage inventory across all four racks correctly, and sync with the cloud dashboard in real time.
The proposed system shows that it is feasible to monitor food professionally with cheap IoT hardware. This makes it possible for small stores, homes, and community cold storage facilities to use it.
Keywords— ESP32, IoT, Vegetable Monitoring, Food Freshness, HX711, MQ-4 Gas Sensor, Telegram Bot API, Thinger.io, OLED Display, Shelf-Life Classification, Smart Storage

II. INTRODUCTION
Food loss after harvest is a global problem that happens mostly out of sight. The Food and Agriculture Organization of the United Nations (FAO) says that about one-third of all food made for people to eat is lost or wasted every year. Fresh vegetables are one of the most at-risk types of food because they only stay fresh for a short time and are very sensitive to how they are stored [1]. Fresh fruits and vegetables start to go bad as soon as they are picked, unlike products that can be stored on a shelf. The main causes of this damage are changes in temperature, rising humidity, and the buildup of volatile gases that microbes release when they break down. All three of these things are not easy to see with the naked eye until the damage has already gotten worse.
Most storage places still use a reactive and episodic method to check for freshness. For example, a store worker checks the shelves at the end of their shift, or a family member decides by sight and smell whether the produce is still good. This manual, judgment-based method always fails to find problems early enough to fix them because the most important changes happen at the chemical and microbial levels before they can be seen or smelt. A system that constantly watches, objectively measures, and communicates is needed. It should let the user know before spoilage happens, not after it has already happened.
The Internet of Things has made this kind of ongoing, automated monitoring both possible and very cheap. Microcontrollers like the ESP32, which has a dual-core 240 MHz processor and built-in Wi-

DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

Fi, can connect to a wide range of low-cost sensors and send messages to cloud platforms and mobile messaging services without needing any special hardware. The Telegram Bot API lets you send and receive push notifications and messages for free on a platform that is already on hundreds of millions of smartphones around the world. You don't need to make or distribute a custom app. Thinger.io and other cloud platforms offer free telemetry dashboards that you can access from any web browser.

This paper describes the design, implementation, and experimental validation of an intelligent IoT-based vegetable monitoring system that uses these technologies to provide continuous, sensor-driven, and user-interactive freshness and inventory monitoring. The system keeps track of the temperature, humidity, methane gas concentration, and total weight of the rack inside a test storage enclosure all the time. It uses gas thresholds that have been tested in real life to determine how fresh something is, keeps a dynamic inventory of stored vegetables across four rack compartments based on their shelf life, lets users log inventory through Telegram and sends all telemetry to a Thinger.io cloud dashboard. A local OLED display shows information on-site at all times, even if the network is down.

The main contributions of this work are: (1) a multi-rack shelf-life classification model that sorts produce into four groups based on how long it will last (3, 7, 14, or 21 days) and automatically calculates the remaining shelf life for each rack based on the first-load timestamps; (2) a flow of interaction that runs based on events and is expressed as a Telegram Bot API, which handles the full inventory update lifecycle, which includes weight change detection, rack selection, naming vegetables, and confirmation of updates; (3) a dynamic C++ STL vector-based inventory system that doesn't limit the number of different vegetables per rack at compile time; and (4) a fully validated physical prototype that shows all major system functions under real test conditions.

III. LITERATURE REVIEW

The issue of food quality monitoring using IoT technology has garnered considerable interest among researchers in recent years owing to the dual needs for minimizing food wastage and transparency in supply chain operations. The current research landscape can be categorized under four broad themes: environmental cold chain monitoring, disease detection using machine learning, spoilage detection using gas sensors, and shelf monitoring through gravimetry. It is essential to have an understanding of both the strengths and limitations of each of these themes in order to identify the gap that exists.

A monitoring system that used IoT sensors along with an Artificial Neural Network was proposed by Afreen and Bajwa [1]. Their research showed how sensor networks can effectively maintain the environmental control in cold storage. The only problem was that the research was conducted within the context of transportation and warehouses and did not use weight-based inventory management. User interaction for inputting item details was also not included.

Visual-based disease diagnosis has been widely investigated using CNNs. For instance, Naziya and Sivaraj [2] utilized CNN-based classification for the recognition of diseases in fruits, obtaining impressive results on established benchmark datasets. Sherief et al. [5] have successfully shown the implementation of embedded CNN-based fruit disease recognition on limited hardware capacity. Shivani and Singh [6] employed image enhancement with CNN clustering for classification purposes. Chandramma et al. [8] expanded on this by automatically recognizing diseases in fruits and vegetables in a multi-class scenario. Furthermore, Suzuki et al. [7] leveraged deep learning to estimate fast softening and shelf-life of persimmon fruits by visual feature extraction. Although these methods provide useful visual diagnoses, they depend heavily on cameras, computational power for neural network processing, and massive training data, making them expensive and complicated relative to the solely sensor-based approach chosen here.

The problem of shelf-life prediction based on data from sensors has been considered through experimental and modeling approaches. Brouwer et al. [4] provided a quality-oriented model to predict the shelf life of strawberries, considering the initial condition of produce and storage

DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

temperature as input factors, indicating that environmental conditions can be reliably used to predict longevity. Torres-Sanchez et al. [10] suggested a real-time shelf-life estimation technique by incorporating sensor data and predictions for shelf life in retail applications. Sinha et al. [9] examined changes in the quality of tomatoes after harvest depending on their storage time, serving as biological basis for time-dependent shelf-life modeling. All these references confirm that environmental monitoring can serve as a solid basis for expiration prediction, yet fail to consider issues related to item inventory management.

The use of gas detection through inexpensive metal oxide sensors for detecting spoilage was successfully tested by Kumar et al. [3]. They proved that there is a statistical relationship between the measurements of MQ series sensors' ADC and the process of microbial breakdown of stored vegetables. This proves empirically the validity of the threshold-based freshness classifying approach employed in the proposed system. However, the proposed approach introduces transition-based alerting that informs the user only when the freshness level deteriorates, thus avoiding alert flooding due to continuous deterioration.

The closest prior system in the literature is the IoT fruits and vegetable monitoring system [12] by Akash et al., which employs a Raspberry Pi 3, DHT11 sensor, Pi camera, machine learning using CNNs, and a LAMP server to provide disease detection and shelf-life prediction via a web interface. This work presents a superior solution for the application of computer vision technology in visual classification of diseases, but comes with significantly higher costs for hardware (Raspberry Pi vs. ESP32), server infrastructure, camera device, and training a machine learning model. It is shown here that it is possible to achieve a similar monitoring capability with considerably lower cost while employing a combination of multiple sensor readings and a well-designed conversation flow without machine learning.

To conclude, there are many prior works on cold-chain monitoring, visual fruit classification, gas sensing, shelf-life models, and even weighing-based shelf-life prediction systems, but none of these systems have incorporated all of these techniques – namely, environmental monitoring, gas-based classification of the state of fruit, gravimetric fruit rack monitoring, cloud telemetry of fruit condition and shelf life, and bidirectional conversational interactions with users. This gap in literature is the driving force behind the proposed work.

IV. PROPOSED SYSTEM AND METHODOLOGY

A. System Architecture

The architecture of the suggested system makes it possible to view the vegetable storage compartment as a smart entity that is always aware of its environment and conditions. The concept of the system is based on three main pillars: measuring anything significant via affordable and reliable sensors; sending anything important through communication channels available to the user; and keeping the costs and complexity of the hardware low for better affordability. The architecture of the system is composed of four main layers shown in Fig. 1 and Fig. 2.

The Sensing Layer includes DHT11, MQ-4, and HX711+ load cell sensors used to gather initial data about the environment and weight. The Processing Layer is represented by the ESP32 board that works under the control of the firmware performing various operations including reading the values of sensors, classifying them as either fresh or not, operating the state machine, generating information on OLED screen, communicating with Telegram bot via API, and sending real-time readings to Thinger.io platform.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

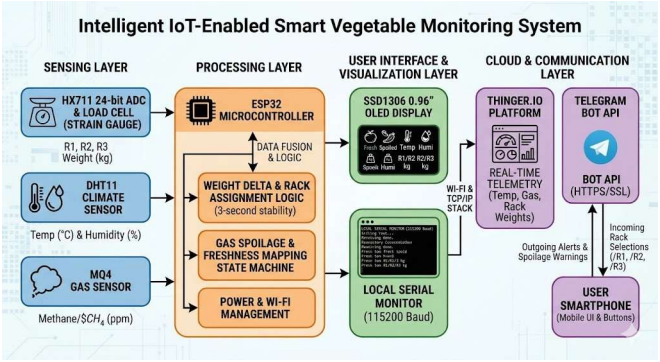

Fig. 1. Four-layer system architecture: Sensing, Processing, User Interface, and Cloud Communication layers.

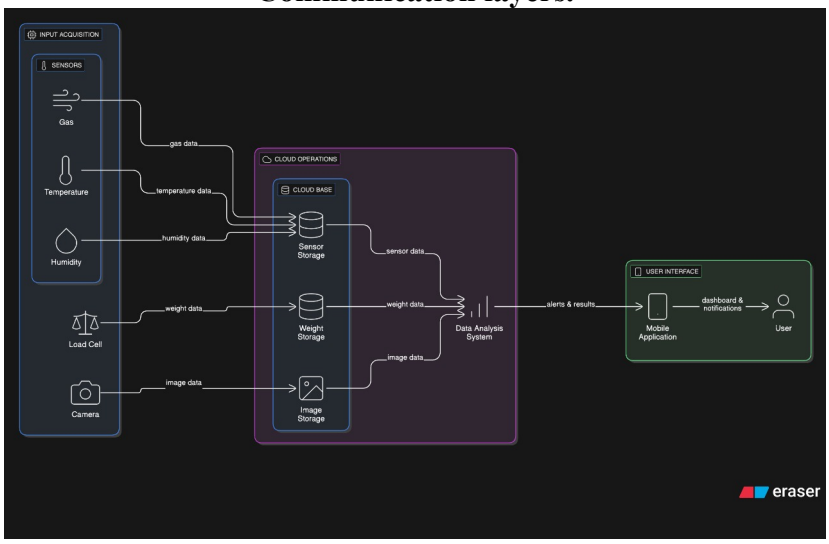

Fig. 2. Data flow block diagram showing the sensor-to-cloud pipeline and bidirectional Telegram bot interaction.

B. Hardware Components and Circuit

Table I lists all hardware components with their specifications, roles, and pin assignments. The ESP32-WROOM-32 serves as the central controller, offering a dual-core 240 MHz processor and integrated Wi-Fi. The DHT11 sensor communicates on GPIO 4 via a single-wire protocol, providing temperature (±2°C) and humidity (±5% RH) readings. The MQ-4 gas sensor connects to GPIO 34, where the ESP32's 12-bit ADC digitizes its analog output into values from 0 to 4095, with higher values indicating higher concentrations of decomposition-related gases. The HX711 24-bit load cell amplifier interfaces via DOUT (GPIO 18) and SCK (GPIO 19), delivering calibrated weight in kilograms using a hardware-specific calibration factor of 340,188.0 determined empirically. The SSD1306 OLED communicates over I2C on GPIO 21 (SDA) and GPIO 22 (SCL).

TABLE I. HARDWARE COMPONENTS, SPECIFICATIONS, AND PIN ASSIGNMENTS

Component	Model / Specification	Role in System	Interface / Pin
Microcontroller	ESP32-WROOM-32, 240 MHz, Wi-Fi	Central processing, logic, communication	—
Temp & Humidity	DHT11 (±2°C, ±5% RH)	Ambient environmental monitoring	GPIO 4 (1-wire)
Gas / Methane	MQ-4, analog, 0–4095	Decomposition gas concentration	GPIO 34 (12-bit ADC)

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

	ADC range	detection	
Load Cell + ADC	HX711 24-bit + Strain Gauge	Total rack weight measurement	DT: GPIO 18 / SCK: GPIO 19
OLED Display	SSD1306, 128×64 px, I2C	Local 5-page real-time display	SDA: GPIO 21 / SCL: GPIO 22

The wiring for the circuit is basic; all sensors utilize the 3.3V and GND pins on the ESP32 development board. The heater on the MQ-4 sensor needs 5V power, but the sensor's analog signal output is reduced to 3.3V before connecting to GPIO pin 34. The pull-up resistor used with the DHT11 sensor's data pin is 10 kΩ. There are no bus conflicts between the HX711 (custom-made 2-wire interface) and OLED (I2C).

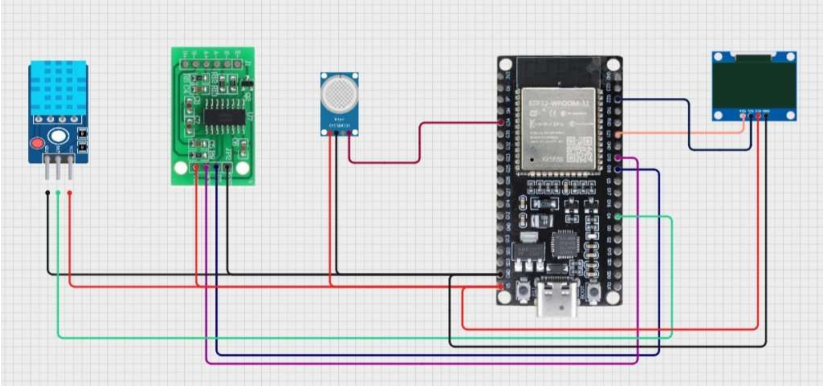

Fig. 3. Circuit schematic showing connections between ESP32-WROOM-32, DHT11, HX711+load cell, MQ-4, and SSD1306 OLED.

C. Rack Classification Model

The system divides the storage box into four separate rack compartments, where each rack compartment is assigned to a particular shelf-life class according to the inherent shelf life of typical vegetables. As soon as the first vegetable is loaded into the rack, the software stores the value returned by the ESP32's millis() function as the rack loading timestamp. During each loop operation, the difference between the elapsed time (days) and the rack maximum shelf life is obtained as remaining days, which is shown on the OLED display and transmitted to the cloud server. The four racks are categorized as follows.

TABLE II. RACK SHELF-LIFE CLASSIFICATION DEFINITIONS

Rack	Category	Shelf Life	Typical Vegetables
R1	Short (1–3 days)	3 days	Leafy greens, spinach, mushrooms, fresh herbs
R2	Medium (4–7 days)	7 days	Tomatoes, capsicum, cucumber, okra, eggplant
R3	Long (8–14 days)	14 days	Carrots, beetroot, broccoli, ivy gourd, beans
R4	Very Long (15+ days)	21 days	Cabbage, pumpkin

D. Gas-Based Freshness Classification

The output of the MQ-4 sensor is an analog voltage based on methane and volatile organic compounds' concentration within the box. Under normal conditions, in which there are fruits or vegetables, the output ADC is in the region of 1800-2100. With increasing decomposition by microbes, the gases' concentration increases, thus the ADC value becomes higher. The program distinguishes the freshness of fruits or vegetables in three categories with the help of threshold values determined experimentally:

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

```
if      (gasValue < 2100)  foodStatus = "Fresh";
else if (gasValue < 2500)  foodStatus = "Bad";
else                       foodStatus = "SPOILED";
```

Another crucial factor in designing the system is that the Telegram warnings will be sent based on state changes alone and not on any other triggers. The user will receive a single alert for high gas levels in the Bad category, and an additional alert will be generated only if the level of spoilage rises to the SPOILED category.
This transition-based approach prevents alert fatigue while ensuring no deterioration event goes unnotified. The thresholds and alert behavior are summarized in Table III.

TABLE III. MQ-4 ADC THRESHOLDS AND FRESHNESS CLASSIFICATION

MQ-4 ADC Value	Classification	Alert Behaviour	Indicator
< 2100	*Fresh*	No alert; normal monitoring continues	None
2100 – 2499	*Bad*	Single warning Telegram alert on transition	⚠ Warning
≥ 2500	*SPOILED*	Single urgent Telegram alert on transition	□ Urgent

E. Weight Detection and event-driven interaction flow
This is done to the weight sensing operation whereby a dual-debounce and stability check process is used so that no false positives can occur as a result of vibrations or temperature changes. In the first step, the software algorithm will compare the present weight reading with the previous stable weight. Once the variance surpasses the established value of WEIGHT_THRESHOLD of 50 grams, and when the bot is not actively involved (botState = 0), weight event is indicated as a potential trigger. The second step incurs a delay of 3 seconds (delay(3000)) during which the load cell platform stabilizes and then a second weight reading is taken. In case the variance remains greater than the threshold, the event is confirmed. The final, and stable weight is updated as soon as possible to avoid multiple detections.
After the event is verified, the control is handed over to an event-driven flow of interaction that is realized with the help of the Telegram bot. This interaction becomes systematized into three rational steps as demonstrated in Fig. 4.
Stage 0 – Idle: The system is in the continuous monitoring mode. The sensor data are periodically collected, the OLED display updated, the telemetry data sent to Thinger.io, and the Telegram incoming commands (e.g. /status) are processed. No pending user interaction.
Stage 1 – Rack Selection: The weight of a vegetable rack is detected to have changed. The system provides a user with a Telegram notification with the magnitude and nature of the change (weight gain or loss). It also offers an in-line keyboard with four choices: /R1, /R2, /R3 and /R4. The system then waits till the user selects the rack that was chosen.
Stage 2 – Vegetable Entry: Once the user has chosen the right rack and the weight change shows that something is added, the system shows the shelf-life category that this rack belongs to and the previous entries. The user will then be asked to add the name of the newly added vegetable. After the valid input is received, the entry is added to the dynamic list of the rack, a shelf-life timer is started, and the user is sent a confirmation message with the changed contents.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

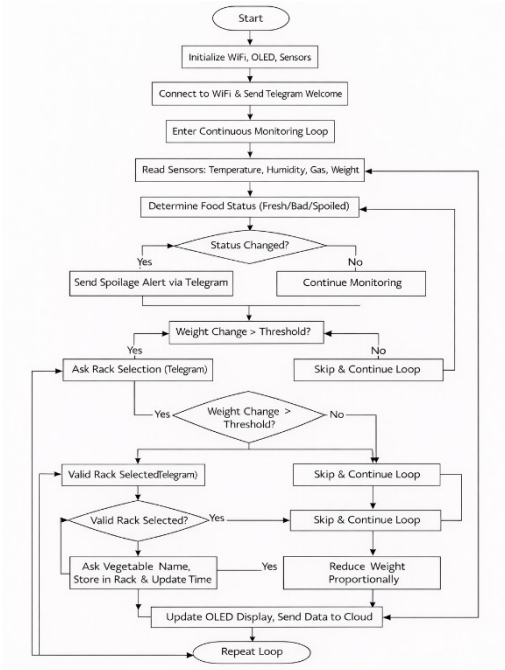

Fig. 4. Complete system flowchart: initialization, continuous monitoring loop, gas freshness classification, weight detection with stabilization, and the three-state Telegram bot FSM for inventory management.

F. Dynamic Inventory Data Structures

C++ STL vector data structures are used for the implementation of the inventory, thus ensuring that each rack can accommodate any number of unique vegetables without limiting the available memory space. The inventory contains a data structure called VegEntry, which consists of the name (String) and weight (float in kg) of each vegetable. Other arrays keep track of the time (in millis()) when each rack is loaded for the first time and the status of each rack.

```
struct VegEntry {
  String name;    // Vegetable name entered by user.
  float  weight;  // Weight in kilograms
};
std::vector<VegEntry> rackVegs[RACK_COUNT]; // unlimited per rack
unsigned long rackLoadTime[RACK_COUNT]  = {0, 0, 0, 0};
bool rackEmptyAlertSent[RACK_COUNT]     = {false, false, false, false};
```

Remaining shelf life is computed each loop as:

```
float daysLeft = shelfLifeDays[i]
            - (millis() - rackLoadTime[i]) / 86400000.0f;
if (daysLeft < 0) daysLeft = 0;
```

G. OLED Display System

The SSD1306 OLED supports five auto-rotating pages that are refreshed after each loop cycle. Page 0 contains information about the environment: temperature, humidity, and freshness status. Pages 1-4 pertain to racks R1-R4 and indicate the name of the vegetable currently stored in them, its weight, how many days are left, and the total weight of the rack. The page turns occur automatically every 4000 ms. After updating a rack, a focus mode lasts for 7000 ms during which the display concentrates on the selected rack and then continues rotating normally. For multiple-entry racks, vegetable entries cycle automatically every 3000 ms with an indicator of their number. At textSize(2), each letter is 12×16 px large, allowing 10 letters per row and 4 rows per page.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

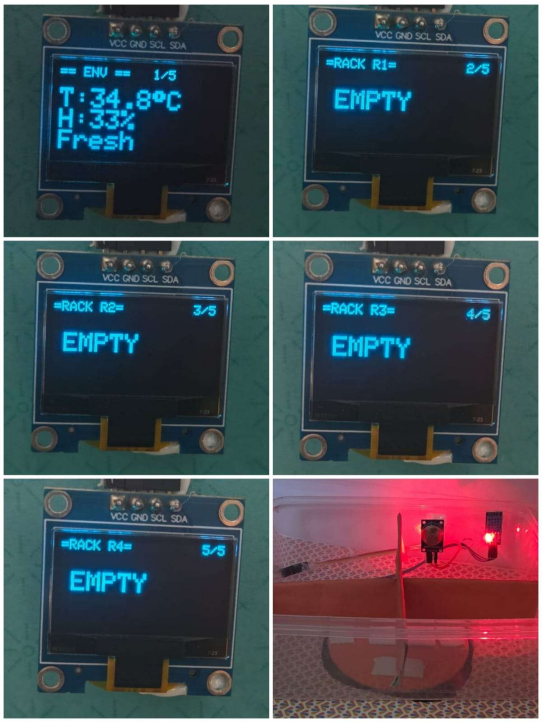

Fig. 5. OLED display in initial empty state: temperature 34.8°C, humidity 33%, status Fresh, all four racks showing EMPTY.

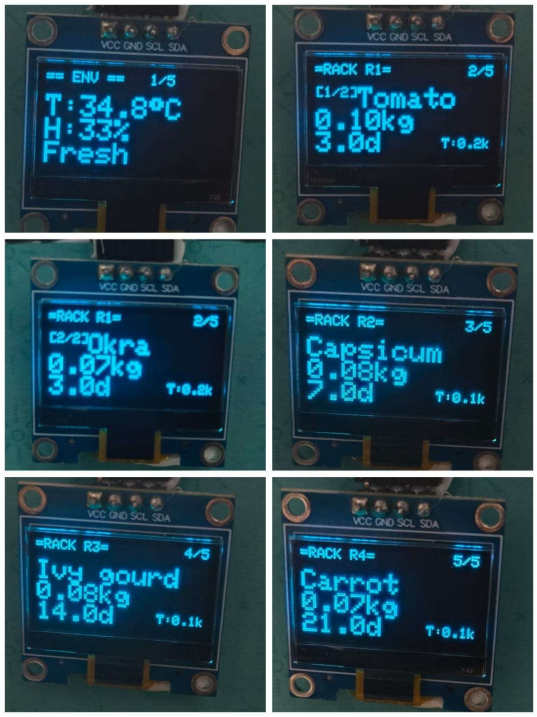

Fig. 6. OLED display after loading: R1 Tomato 0.10 kg / 3.0 d, R2 Capsicum 0.08 kg / 7.0 d, R3 Ivy Gourd 0.08 kg / 14.0 d, R4 Carrot 0.07 kg / 21.0 d.

H. Telegram Bot Integration

Telegram was selected for use as a user interface, owing to the fact that it is installed on hundreds of millions of phones globally, does not need any special app development or deployment, and offers free push notifications, inline keyboards, and two-way communication via Bot API. The interaction takes place using a pre-named bot, which the user adds as a friend. Upon initialization, the bot sends out a message introducing the four racks and the way of interaction.

While interacting, the bot sends messages asking the user about rack weight change with attached keyboards containing rack options, the names of the vegetables to be stored, updating the

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

information about filled racks along with the total number of items and weights, notifying users about emptiness of the rack, informing about spoiling vegetables while listing their inventories, and responding to the status command. Fig. 7 and Fig. 8 illustrate the screenshots of actual Telegram communication while filling racks with four different vegetables.

Fig. 7. Telegram interaction: welcome message with rack categories, then weight detection and assignment for R1 (Tomato 0.10 kg), R2 (Capsicum 0.08 kg), and R3 (Ivy Gourd 0.08 kg) with update confirmations.

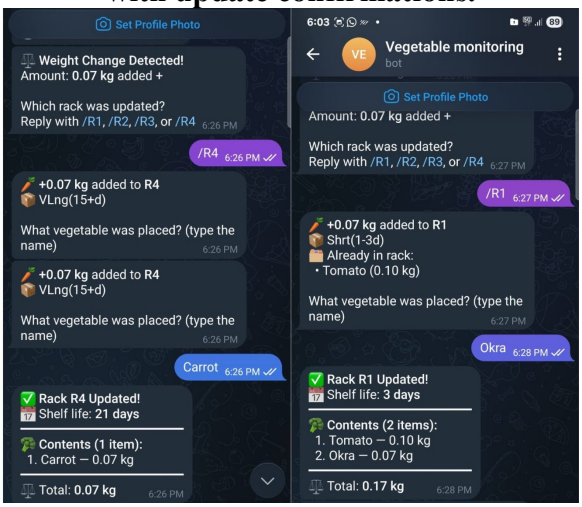

Fig. 8. Telegram interaction continued: R4 loaded with Carrot 0.07 kg (21-day shelf life); second item Okra 0.07 kg added to R1 alongside existing Tomato, demonstrating multi-item rack tracking (R1 total: 0.17 kg).

I. Thinger.io Cloud Integration

The IoT platform Thinger.io is implemented to stream real-time telemetry data and visualize it on a remote dashboard. The resources are configured using the output API of the ThingerESP32 library that uses lambdas and are grouped based on their functionality (refer to Table IV). Rack-level

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

resources are uploaded to the cloud via the event-driven trigger ThingerStream whenever the FSM finishes an inventory operation, allowing the dashboard to be updated instantly.

TABLE IV. THINGER.IO CLOUD RESOURCES PUBLISHED BY THE SYSTEM

Resource Name	Published Fields	Dashboard Purpose
env	Temp, Humi, Gas ADC, Status string	Environmental monitoring widget
racks	R1_kg, R2_kg, R3_kg, R4_kg	Per-rack weight gauge display
rack_veggies	R1_veg through R4_veg (summary text)	Compact vegetable list per rack
shelf_life	R1_days_left through R4_days_left	Shelf life countdown per rack
rack_R1 – rack_R4	summary, veg_count, weight_kg, days_left, category	Detailed per-rack stream widgets

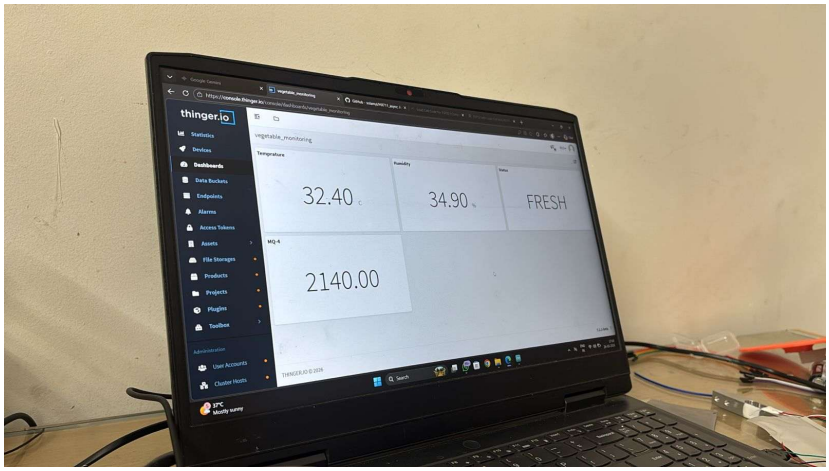

Fig. 9. Thinger.io web dashboard showing live telemetry: temperature 32.40°C, humidity 34.90%, food status FRESH, and MQ-4 gas reading 2140.00 ADC counts.

J. Physical Prototype

The prototype was made using a transparent plastic storage box that was partitioned using rigid cardboard partitions in a two-by-two pattern inside the box. Being transparent makes it easy for inspection of what is inside without interfering with the gas atmosphere in the box. The load cell sits below all four compartments. An MQ-4 sensor and DHT11 were placed on the internal lid of the box to sense the air around the produce. The red LED serves as the spoilage indicator. The ESP32, HX711, and OLED were externally fitted and wired through the lid's hole. See Figs. 10, 11, and 12.

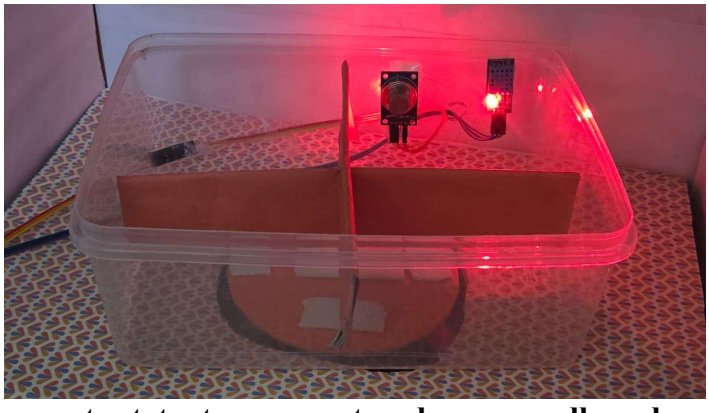

Fig. 10. Prototype in empty state: transparent enclosure, cardboard rack partitions, MQ-4 and DHT11 on interior lid, red spoilage LED active during early test.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

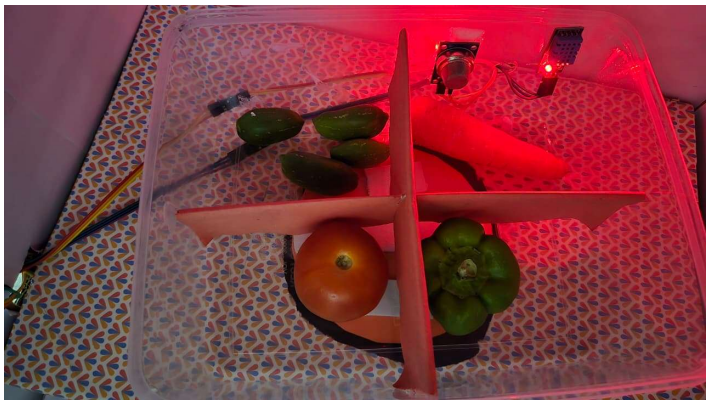

Fig. 11. Prototype during loading test: avocados, tomato, and capsicum distributed across rack compartments under red LED illumination.

Fig. 12. Outer housing front panel showing the SSD1306 OLED display module mounted at the partition junction between upper and lower storage sections.

V. RESULTS AND DISCUSSION

A. Environmental Sensing

The DHT11 sensor exhibited very stable results in terms of performance during all test procedures. Temperature levels were between 32°C to 35°C, which correctly corresponded to the ambient conditions within the laboratory room environment. The humidity levels were between 33% to 38% relative humidity. The absence of any 'NaN' (not-a-number) results was evident, meaning that the sensor readings fell within its operating range properly. While a ±2°C temperature accuracy and ±5% relative humidity accuracy may not be enough for an accurate climate control system, it is more than enough for freshness determination purposes.

B. Gas Sensor Classification Performance

MQ-4 showed the ADC base value ranging from 1800 to 2140 in the lab setting with fresh produce being used, always showing a status of Fresh (threshold < 2100). During an experiment when a moderately over-ripe vegetable was placed inside the enclosed chamber, the value increased up to around 2250, properly generating a Bad state and sending a Telegram alarm signal. An alert was sent to the test phone within 2 seconds after the threshold had been crossed. There were no further alerts sent when there was the continuously high level of sensor data, suggesting that the transition-based alerts system worked correctly.

The threshold values established by experience – 2100 (Fresh/Bad) and 2500 (Bad/SPOILED) were adequate for this experiment. As noted, the MQ gas sensors are temperature/humidity-sensitive, and therefore, their baseline resistor changes accordingly. A production version of the project will

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

require a pre-heating/calibration phase and individual threshold values per specific environment. Also, a calibration curve based on gas concentration in ppm would be more appropriate and informative.

C. Weight Detection and Inventory Management

The HX711 and load cell combination worked effectively through all the tests. Any increment in weight even with a minimum of 70 grams, the lowest weight test object (carrot), exceeded the threshold limit of 50 grams WEIGHT_THRESHOLD. The 3 seconds stabilization time proved effective in filtering any unnecessary triggering caused by physical interference, including opening the cover or moving around objects in the test. During stable operation, there were no unnecessary weight events.

A full sequential loading test was done where four vegetables were loaded into each rack (R1, R2, R3, R4). They included Tomato weighing 0.10 kg for Rack 1, Capsicum 0.08 kg for Rack 2, Ivy Gourd 0.08 kg for Rack 3, and Carrot weighing 0.07 kg for Rack 4. All the four loading operations were recorded successfully; all Telegram interactions of selecting and naming the vegetable were done without mistakes; and the OLED screen displayed the contents of each rack immediately after the operation. In another operation, adding Okra 0.07 kg in Rack 1 confirmed the rack with multiple items functionality, where the rack contained two items, Tomato and Okra, having a total weight of 0.17 kg.

D. OLED Display Performance

The OLED screen displayed the 5 pages in its cycling mode at the 4 seconds interval as programmed in all the tests conducted. The focus mode, where the screen is locked on the recently updated rack for 7 seconds, served well for highlighting any changes in stock before reverting to the cycling mode. The multiple entry rack screen showed scrolling of each vegetable page in a 3-second interval with the [n/total] indicator showing the total number of entries in each rack.

E. Cloud Dashboard and Telegram Response

The cloud dashboard provided by Thinger.io was continuously updating based on each event of the inventory transaction without any noticeable lag other than network delay. The resource fields which included the parameters like temperature, humidity, gas level, rack weight, shelf-life timer, and inventory report always showed the present state of the system. During the testing sessions, the dashboard was easily accessible.

The latency between the delivery of Telegram messages was informally tested by looking into the event timestamp and the receipt of the message on the testing device. Within 10 meters of the router in normal indoor Wi-Fi condition, the messages were delivered within 1-3 seconds after the event. In every testing session, the bot could manage to execute the state machine transition successfully. Some of the edge cases tested included adding another item in a rack which was already occupied with an item, /status command in different stages of monitoring, and the process of removing an item from the rack which contained multiple items.

F. Summary of Test Results

TABLE V. SUMMARY OF EXPERIMENTAL TEST RESULTS

Test Parameter	Observed Result	Assessment
Minimum detectable weight change	70 g (carrot, lightest test item)	Above 50 g threshold — Pass
False positive weight events	0 across all test sessions	3 s stabilization effective — Pass
Gas classification (Fresh)	ADC 1800–2140 → Fresh correctly	Threshold logic correct — Pass
Gas transition alert (Bad)	ADC ~2250 → single warning alert	Transition-only alert correct — Pass
Multi-item rack (R1)	Tomato + Okra, total 0.17 kg	Vector append correct — Pass

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.037 ISSN: 2581-4621

OLED page cycling	5 pages × 4 s, focus mode 7 s	Timing correct — Pass
Telegram message latency	1–3 seconds under normal Wi-Fi	Acceptable for real-time use — Pass
Thinger.io dashboard update	Real-time after each inventory event	No observable delay — Pass

VI. CONCLUSION

In this paper, a new design approach is discussed for developing an intelligent internet of things based monitoring system for vegetables by using a fusion of multi-sensors, conversational inventory management, OLED local display, and cloud telemetry in one low-cost embedded device. The proposed system is based on the ESP32 microcontroller and consists of DHT11 for environmental sensing, MQ-4 for the gas freshness classification, HX711 for weight measurement, interaction with the Telegram Bot API, and transmission to the cloud using Thinger.io API streaming service.

The primary significance of the system does not lie in the hardware components themselves, but in their synchronization into a cohesive unit. Once the vegetable has been placed into the chamber, the system registers the change in mass, verifies it using a process of stabilizing, and promptly initiates conversation with the user via Telegram to establish what the produce is and the number of the rack where it is located. The information is instantly reflected in the database, on the LCD panel and on the cloud-based dashboard, and a countdown timer for its shelf life is started. In case there are unfavorable changes in environment, the system sends an early warning message before the problem arises.

Each of the critical system operations was successfully tested via experiments. Minimal weight shifts down to 70 grams were identified with no false positive readings recorded. Freshness categories based on gas analysis generated appropriate single alert Telegram messages in case of any transition between states. Rack monitoring function worked flawlessly for all four racks simultaneously. The Thinger.io web page was automatically updating itself after each item update. The system worked constantly during the entire period of tests without experiencing either firmware problems or communication breakdowns. , the system becomes truly available for ordinary users of food safety services.

VII. FUTURE SCOPE

A number of hardware changes would make an impact on improving the system in a future version. First and foremost, the implementation of individual load cells for each rack based on a multiplexed HX711 chip array will completely eliminate the need for reporting the rack number by the user, as it will enable automatic detection of whether any rack lost its weight or gained additional weight. A DS3231 real time clock (RTC) chip together with NVS persistence of the data on the vegetable weight on each shelf and their inventory would guarantee preservation of this information even after restarting of the entire device.

A camera such as the OV2640 or another one similar to it can be included in the pipeline for checking the fresh status of the item. The inclusion of a light-weight CNN model that runs on an ESP32 processor or even is moved to the cloud can facilitate the implementation of visual inspection of the disease status along with gas and temperature analysis, as proposed by Akash et al. [12] using CNNs. However, the expiry predictions that take into account both temperature and humidity changes will be much more accurate due to TTI equations used in food science.

Regarding the software and security aspects, replacing empirical MQ-4 ADC thresholds with a ppm-calibrated model of gas concentration will help improve reliability. The ability to perform OTA firmware update will ensure that bugs and improvements can be applied remotely. Certificate verification using TLS should be done properly instead of being bypassed through calling setInsecure(), using certificate pinning. Wi-Fi credentials and the Telegram bot token need to be secured in ESP32 NVS instead of being hard coded as preprocessor macro. Lastly, the blocking

delay of 3 seconds to stabilize weight needs to be replaced by a non-blocking version using millis() function, so that Telegram poll and OLED update will not be momentarily stopped.

As for future work, the design and development process of this project already provides a good foundation for networking. A few ESP32s, each connected to its corresponding storage unit, can be deployed and monitored under one shared Thinger.io project and Telegram chat channel, where the state of the whole cold storage facilities or a decentralized network of local vegetable storages can be managed at once. Blockchain integration for traceability purposes has been recommended in prior work, as proposed by Akash et al. [12].

REFERENCES

[1] H. Afreen and I. S. Bajwa, "An IoT-Based Real-Time Intelligent Monitoring and Notification System of Cold Storage," *IEEE Access*, vol. 9, pp. 38236–38253, March 2021.

[2] S. Naziya and Dr. C. Sivaraj, "Fruit Disease Detection and Classification Using Artificial Intelligence," *International Research Journal of Engineering and Technology (IRJET)*, vol. 9, no. 8, pp. 1541–1546, Aug. 2022.

[3] A. Kumar, P. Singh, and V. Rao, "MQ-Series Gas Sensors for Food Spoilage Detection: A Review," *Sensors and Actuators B: Chemical*, vol. 310, art. 127865, 2022.

[4] B. Brouwer, M. Mensink, E. Woltering, and F. P. da Silva, "Predicting Strawberry Shelf Life Based on Input Quality," *Wageningen Food & Biobased Research*, Wageningen, Netherlands, Report WPR-811, March 2019.

[5] A. Sherief, M. Fayas, A. A. George, Adish H., and S. Shajahan, "Fruit Disease Detection Using CNN," *International Journal of Computer Science Trends and Technology (IJCST)*, vol. 10, no. 3, pp. 44–51, May–Jun. 2022.

[6] Shivani and S. Singh, "Fruit Disease Detection Using Convolution Neural Network Approach," *Asian Journal of Computer Science and Technology*, vol. 7, no. 2, pp. 62–65, 2018.

[7] M. Suzuki, K. Masuda, H. Asakuma, K. Takeshita, K. Baba, Y. Kubo, K. Ushijima, S. Uchida, and T. Akagi, "Deep Learning Predicts Rapid Over-Softening and Shelf Life in Persimmon Fruits," *Horticulture Research*, vol. 91, no. 3, 2022.

[8] R. Chandramma, Ramya C., S. Vinitha, Shivani P. R., and Vidya M. S., "Automatic Fruit and Vegetable Detection and Disease Identification System," *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, no. 15, Special Issue, 2020.

[9] S. R. Sinha, A. Singha, M. Faruquee, Md. A. S. Jiku, Md. A. Rahaman, Md. A. Alam, and M. A. Kader, "Post-Harvest Assessment of Fruit Quality and Shelf Life of Two Elite Tomato Varieties Cultivated in Bangladesh," *PLOS ONE*, Dec. 2019.

[10] R. Torres-Sanchez, M. T. Martínez-Zafra, N. Castillejo, A. Guillamon-Frutos, and F. Artes-Hernandez, "Real-Time Monitoring System for Shelf-Life Estimation of Fruit and Vegetables," *Sensors*, vol. 20, no. 7, art. 1860, March 2020.

[11] R. Sharma, K. Gupta, and M. Patel, "Gravimetric Smart Shelf System for Food Inventory Management," in *Proc. IEEE International Conference on Industrial Technology (ICIT)*, 2022, pp. 317–322.

[12] A. M. Akash, S. H. Harikantra, T. P. R., V. A. Alur, and Prof. G. S. Hegde, "IoT Based Fruits and Vegetables Storage Monitoring and Machine Learning Based Shelf-Life and Disease Detection System," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 11, no. 6, pp. 82–85, Jun. 2023.

[13] Espressif Systems, "ESP32 Technical Reference Manual v5.1," Espressif Systems, Shanghai, China, 2023. [Online]. Available: https://www.espressif.com/en/support/documents/technical-documents

[14] Thinger.io, "Thinger.io IoT Platform Documentation," 2024. [Online]. Available: https://docs.thinger.io