Project Report On

E-Paper Display-Based Shelf Updation System

Submitted in Partial Fulfillment for the award of

Post Graduate Diploma in DESD

Design (PG-DESD)

from

C-DAC, ACTS (Pune)

Guided by:

Mr. Rishabh Hardas

Presented by:

Ms. Bokil Arundhati Dhananjay	**PRN:250840130012**
Mr. Harshit Bhandari	**PRN:250840130019**
Ms. Kaushkey Kumari	**PRN:250840130021**
Ms. Sruthy P J	**PRN:250840130049**
Ms. Yadula Suguna	**PRN:250840130055**

Centre for Development of Advanced Computing (C-DAC), Pune

ACKNOWLEDGEMENT

This project "E-Paper Display-Based Shelf Updation System" was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We are very glad to mention the name of *Mr.Rishabh Hardas* for their valuable guidance to work on this project. Their guidance and support helped me to overcome various obstacles and intricacies during the course of project work.

Our heartfelt thanks goes to *Ms. Jubera Khan*(Course Coordinator, *PG-DESD*) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility, and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

1. Introduction

1. Introduction
2. Literature Survey
3. Aim of Project
4. Scope and Objective

2. THEORETICAL DESCRIPTION OF PROJECT

1. Internet of Things
2. MQTT
3. ESP32
4. BLE
5. E-Paper Technology
6. Electronic shelf Label system
7. Cloud and Shelf data
8. Data flow and Communication Model

3.SYSTEM ARCHITECTURE AND DESIGN

1.Overall System Architecture
2.Hardware Components
3.Software Components
4.Communication Protocol Design

4. HARDWARE IMPLEMENTATION

1. Hardware Requirements
2. Hardware Setup Overview
3. Circuit Diagram of ESP32 Gateway
4. Circuit Diagram of ESP32 Shelf Node
5. Interfacing ESP32 with E-Paper Display

5. SOFTWARE IMPLEMENTATION

1. Software Requirements
2. Gateway ESP32
3. Shelf Node ESP32
4. MQTT Topic Structure and Data Format

6. SYSTEM ARCHITECTURE AND DATA FLOW

1. Overall System Architecture
2. Cloud Layer Architecture
3. ESP32 Gateway Functional Architecture
4. ShelfEdge Architecture
5. Power Management
6. Product and Price Update Data flow
7. Firmware Update and OTA mechanism

7. TOOLS AND RESULTS

1. Cloud Dashboard
2. Local Data Update
3. System Logs

8. CONCLUSION

9. FUTURE SCOPE

10. REFERENCES

ABSTRACT

The Shelf Updation System using E-Paper Display is an Internet of Things (IoT) based solution developed to automate the process of updating shelf information such as product name, price, offers, and quantity in real time. In conventional retail environments, shelf labels are updated manually using paper tags, which is a time-consuming process and often leads to inconsistencies between billing systems and displayed prices. Such manual methods are inefficient, prone to human error, and difficult to manage across large-scale retail or warehouse environments.

To overcome these limitations, the proposed system introduces a cloud-controlled electronic shelf labeling mechanism that enables remote and instant updates. The system architecture consists of two ESP32 microcontrollers, a cloud platform, a Bluetooth communication module, and an e-paper display. Shelf data is updated from the cloud and transmitted to a gateway ESP32 (ESP32-B) using the lightweight MQTT protocol, which is well suited for IoT applications due to its low bandwidth consumption and reliable message delivery.

The gateway ESP32 acts as an intermediary between the cloud and the display unit. Upon receiving updated shelf data via MQTT, ESP32-B forwards the information wirelessly to another ESP32 (ESP32-A) using Bluetooth Low Energy (BLE). This short-range wireless communication ensures low power consumption while maintaining reliable data transfer between the gateway and the display controller.

ESP32-A processes the received data and communicates with the Waveshare 2.9-inch e-paper display using the SPI protocol. The e-paper display updates the shelf information and retains the displayed content even when power is removed, making it highly suitable for energy-efficient and battery-operated systems. Due to its paper-like readability and minimal power usage during refresh operations, the e-paper display significantly enhances the overall efficiency of the system.

The proposed Shelf Updation System provides a scalable, low-power, and cost-effective solution for modern retail and inventory management systems. By enabling real-time cloud-based updates and eliminating the need for manual label replacement, the system improves operational efficiency, reduces errors, and ensures consistency across multiple shelves. This solution is particularly suitable for applications in retail stores, supermarkets, warehouses, libraries, and smart inventory management systems.

Keywords: IoT, ESP32, MQTT, Bluetooth Low Energy, E-Paper Display, Electronic Shelf Label, Shelf Automation

CHAPTER 1: INTRODUCTION

1.1. Introduction:

In modern retail, warehouse, and inventory-driven environments, the accurate and timely display of product information plays a critical role in both operational efficiency and customer satisfaction. Shelf labels act as the primary interface between the store and the customer, conveying essential information such as product name, price, discounts, batch details, and stock availability. Despite advancements in backend inventory and billing systems, many retail environments still rely on traditional paper-based shelf labels, which require manual updates whenever product information changes. This process is inefficient, labor-intensive, and susceptible to errors, often leading to inconsistencies between displayed prices and billing systems.

As retail infrastructures grow in size and complexity, the limitations of manual shelf management become more pronounced. Frequent price changes, promotional offers, and dynamic inventory levels demand a system that can update shelf information quickly and accurately across multiple locations. Manual updates not only increase operational costs but also introduce delays and human errors that negatively impact customer trust. These challenges highlight the need for an automated, centralized, and reliable solution capable of updating shelf information in real time with minimal human intervention.

The rise in the Internet of Things (IoT) has enabled the seamless integration of cloud platforms with physical devices, allowing data to be monitored, controlled, and updated remotely. IoT-based systems provide scalability, flexibility, and real-time communication, making them well suited for applications such as electronic shelf labeling. By leveraging IoT technologies, shelf data can be maintained centrally on the cloud and distributed efficiently to edge devices, ensuring uniformity and accuracy across all display units.

Electronic Shelf Label (ESL) systems have gained significant attention as a replacement for traditional labeling methods. Among various display technologies, e-paper displays stand out due to their unique characteristics. E-paper displays consume power only during content refresh, retain displayed information even when power is removed, and offer excellent readability under ambient lighting conditions. These features make e-paper displays particularly suitable for shelf labeling applications, where displays need to remain active for long durations while consuming minimal power.

The system relies on cloud communication for central data control and remote updates. Lightweight messaging protocols such as MQTT are widely adopted in IoT applications due to their low overhead, publish–subscribe architecture, and reliable message delivery. Using MQTT, shelf information updated on the cloud can be instantly transmitted to gateway devices, ensuring timely and efficient data propagation without excessive network usage.

In the proposed Shelf Updation System, a gateway-based architecture is employed to manage communication between the cloud and the display units. A gateway ESP32 (ESP32-B) receives updated shelf information from the cloud using MQTT over Wi-Fi. This gateway acts as a bridge between the cloud infrastructure and the local shelf display network. The received data is then transmitted to another ESP32 (ESP32-A) using Bluetooth Low Energy (BLE) through the HM-10 module. BLE is chosen for its low power consumption and reliability in short-range wireless communication.

The display controller ESP32 processes the received shelf data and updates the Waveshare 2.9-inch e-paper display using the Serial Peripheral Interface (SPI) protocol. This modular design separates cloud communication, local data transmission, and display control, enhancing system scalability and maintainability. The proposed Shelf Updation System offers a low-power, scalable, and automated solution that minimizes manual effort, reduces errors, and ensures real-time consistency of shelf information. The system is well suited for deployment in retail stores, supermarkets, warehouses, libraries, and smart inventory management environments.

1.2. Literature survey

Electronic shelf labeling and smart display systems have gained increasing attention with the growth of retail automation and Internet of Things (IoT) technologies. The literature on shelf updation systems encompasses diverse approaches focusing on automation, wireless communication, power efficiency, scalability, and cloud-based data management. Researchers and developers have explored multiple technologies to replace traditional paper-based shelf labels, aiming to improve operational efficiency and ensure real-time synchronization of shelf information with centralized systems.

Traditional Shelf Labeling System:

Conventional shelf labeling systems rely on paper-based labels that are manually updated by store personnel. Literature highlights that while such systems are simple and cost-effective initially, they suffer from significant drawbacks including high labor costs, frequent human errors, delayed updates, and inconsistencies between shelf prices and billing systems. These limitations become more prominent in large retail environments where product prices and offers change frequently.

Electronic Shelf Label (ESL) Systems

Several studies discuss the adoption of Electronic Shelf Label systems as an alternative to paper labels. ESL systems use electronic displays to update product information remotely. Research indicates that early ESL implementations commonly used LCD or LED displays, which allowed dynamic updates but required continuous power supply. As a result, these systems faced challenges related to high energy consumption, frequent battery replacement, and increased maintenance costs.

IoT-Based Shelf Management Systems

With the advancement of IoT technologies, researchers have proposed shelf management systems that integrate cloud platforms with edge devices. IoT-based approaches enable centralized monitoring and control of shelf data, improving scalability and real-time update capabilities. Literature emphasizes that IoT solutions enhance flexibility and automation but must carefully address challenges such as network reliability, power consumption, and secure data transmission in distributed environments.

Wireless Communication Technologies

Various wireless communication protocols have been studied for use in electronic shelf labeling systems. Wi-Fi-based solutions offer high data rates but are often unsuitable for battery-powered shelf units due to high power consumption. Zigbee-based systems provide low power operation but have limited bandwidth and scalability constraints. Bluetooth Low Energy (BLE) has emerged as a promising alternative, offering reliable short-range communication with minimal power consumption, making it suitable for shelf-level data transmission.

Cloud-Based Communication and MQTT Protocol

Cloud-based communication forms a critical component of modern IoT systems. Literature frequently highlights MQTT as a lightweight and efficient messaging protocol for IoT applications. MQTT's publish–subscribe architecture enables reliable and scalable data exchange between cloud servers and edge devices while minimizing bandwidth usage. Studies indicate that MQTT is well suited for applications requiring real-time updates, such as electronic shelf labeling, especially when combined with gateway-based architectures.

E-Paper Display Technology

E-paper displays have been extensively studied for low-power display applications. Research demonstrates that e-paper technology consumes power only during display refresh and retains information even when power is removed. This makes e-paper displays highly suitable for electronic shelf labels where static information is displayed for long durations. However, literature also notes limitations such as slow refresh rates, which restrict their use to applications that do not require frequent animations or video content.

Scalability and Energy Efficiency Considerations

Scalability and energy efficiency are recurring themes in shelf automation literature. Researchers emphasize the need for modular architectures that allow multiple shelf units to be managed efficiently from a centralized system. Energy-efficient communication protocols, low-power microcontrollers, and display technologies are identified as key factors in achieving long-term, maintenance-free operation in large-scale deployments.

From the literature survey, it is evident that while existing shelf labeling and display systems provide partial automation, they often face challenges related to power consumption, scalability, and maintenance. The combination of cloud-based communication, low-power wireless technologies, and e-paper displays is identified as an effective approach to overcome these limitations. This motivates the development of a cloud-controlled shelf updation system that ensures real-time updates, low power consumption, and improved operational efficiency.

1.3. Aim of Project :

The aim of the E-Paper Display Based Shelf Updation System is to design and implement a smart, automated, and energy-efficient solution for updating shelf information such as product prices, descriptions, and availability in retail and inventory management environments. The proposed system seeks to replace traditional paper-based shelf labels with Electronic Shelf Labels (ESLs) using e-paper display technology, which consumes power only during data refresh and retains displayed information without continuous power supply.

The project focuses on developing a cloud-controlled shelf updation mechanism where shelf data is updated remotely through a centralized cloud platform. Any modification made on the cloud is transmitted to a gateway ESP32 (ESP32-B) using the MQTT communication protocol. This approach enables real-time and synchronized updates while ensuring lightweight, reliable, and scalable data transmission suitable for IoT applications.

Further, the system aims to establish low-power wireless communication between the gateway ESP32 and the shelf node ESP32 (ESP32-A) using a Bluetooth Low Energy (BLE) module (HM-10). This design reduces wiring complexity and power consumption while enabling efficient short-range communication between distributed shelf units.

The shelf node ESP32 processes the received data and updates the Waveshare 2.9-inch e-paper display accordingly. By leveraging the low-power characteristics of both BLE communication and e-paper display technology, the system ensures long operational life and minimal maintenance requirements, making it suitable for large-scale retail deployments.

Additionally, the project aims to demonstrate a modular and scalable IoT architecture that can support multiple shelf nodes through a single gateway, enabling easy expansion of the system. The implementation highlights the practical integration of cloud services, MQTT messaging, Bluetooth communication, and embedded systems for real-world automation.

Overall, the project aims to showcase an efficient smart shelf updation solution that minimizes manual effort, reduces human errors, improves data accuracy, and enhances operational efficiency in modern retail and inventory management systems.

1.4. Scope and Objectives:

Objectives:

- To design and implement a cloud-based shelf updation system that enables real-time modification of shelf information such as product name, price, and availability using the MQTT communication protocol.
- To develop a gateway-based communication architecture where a gateway ESP32 receives data from the cloud and forwards it to shelf node ESP32 devices using Bluetooth Low Energy , ensuring reliable and low-power wireless communication.
- To integrate and control a Waveshare 2.9-inch e-paper display using ESP32, enabling efficient display of shelf data with minimal power consumption and high readability.
- To demonstrate an energy-efficient and scalable IoT solution that reduces manual intervention, minimizes errors in shelf labeling, and supports future expansion for multiple shelf units in retail and inventory environments.

Scope:

The scope of the E-Paper Display Based Shelf Updation System covers multiple aspects of smart retail automation and IoT-based shelf management, focusing on low power operation, wireless communication, and real-time data updates.

- **Low-Power, Energy-Efficient Shelf Management:** The system utilizes bi-stable e-paper display technology, which consumes power only during content refresh and retains displayed information without continuous power supply. This makes the system highly energy-efficient and suitable for long-term deployment in retail shelves where frequent updates are required with minimal power consumption.
- **Cloud-Controlled Shelf Updation Using IoT:** The scope includes cloud-based control of shelf information, enabling product data such as price, name, and availability to be updated remotely. The MQTT protocol is used for lightweight and reliable message delivery from the cloud to the gateway ESP32 (ESP32-B), ensuring real-time and synchronized shelf updates.

- **Wireless Communication Between Gateway and Shelf Nodes:** The system employs Bluetooth Low Energy communication using the HM-10 module for data transfer between the gateway ESP32 and shelf node ESP32 (ESP32-A). This eliminates the need for wired connections, reduces installation complexity, and supports efficient short-range wireless communication within store environments.
- **Embedded System-Based Shelf Node Design:** Each shelf node consists of an ESP32 microcontroller interfaced with a Waveshare 2.9-inch e-paper display. The shelf node is responsible for receiving data from the gateway, processing it, and updating the display. This modular design allows independent operation of shelf units and improves system reliability.
- **Scalability and Modular Architecture:** The architecture supports scalability by allowing multiple shelf node ESP32 devices to communicate with a single gateway ESP32. Additional shelf displays can be added without significant changes to the system design, making it suitable for large retail stores, warehouses, and inventory management systems.
- **Operational Efficiency and Accuracy:** By automating shelf information updates, the system reduces manual effort, minimizes pricing and labeling errors, and ensures consistency between cloud data and shelf displays. This leads to improved operational efficiency, better inventory visibility, and enhanced customer experience.
- **Future Expansion and Enhancements:** The scope of the project can be extended to include advanced features such as over-the-air (OTA) firmware updates for ESP32 devices, integration with inventory management databases, and support for additional sensors. These enhancements can transform the system into a comprehensive smart retail shelf management solution.

CHAPTER 2: THEORETICAL DESCRIPTION OF PROJECT

2.1. Internet of Things(IoT)

- **What is IoT?**

  The Internet of Things (IoT) refers to a network of physical devices embedded with sensors, actuators, microcontrollers, and communication interfaces that enable them to collect, exchange, and act upon data over the internet. These devices are capable of interacting with each other and with centralized cloud platforms, allowing remote monitoring, control, and automation of real-world systems. IoT has become a key enabler in modern automation by bridging the gap between the physical and digital worlds. IoT systems are designed to be scalable and energy-efficient, making them suitable for large deployments with minimal maintenance
- **Use in Retail:**

  In retail, IoT plays a crucial role in enabling smart shelf systems, automated inventory tracking, and real-time pricing updates. By connecting shelf display units to a cloud platform, product information can be updated centrally and reflected instantly across multiple locations. This improves operational efficiency, ensures consistency in displayed data, and enhances customer experience.
- **Why IoT for shelf Updation System?**

  The Shelf Updation System using E-Paper Display is an application of IoT where embedded devices are connected to the cloud for remote data management. In this system, shelf information is updated through a cloud platform and transmitted to edge devices using lightweight communication protocols. The use of IoT enables seamless integration between cloud services and shelf display units, eliminating the need for manual label replacement.
- **Use case for this Project:**

  The E-Paper Display Based Shelf Updation System utilizes IoT to enable real-time, remote updating of shelf information from a centralized cloud platform. When data is updated on the cloud, it is transmitted to a gateway ESP32 using the MQTT protocol. The gateway forwards this data wirelessly to shelf node ESP32 devices via Bluetooth Low Energy. Each shelf node updates the e-paper display accordingly, ensuring synchronized and accurate shelf information. This IoT-based approach reduces manual effort, minimizes errors, and improves operational efficiency in retail environments.

2.2. MQTT

- **What is MQTT?**

  Message Queuing Telemetry Transport (MQTT) is a lightweight, publish–subscribe based messaging protocol designed specifically for resource-constrained devices and low-bandwidth, high-latency networks. It is widely used in Internet of Things (IoT) applications due to its simplicity, low overhead, and reliable message delivery mechanism. MQTT operates over the TCP/IP protocol stack and enables efficient communication between devices and cloud platforms.

- **Publisher-Subscriber Model:**

  The MQTT communication model is based on three main components: the publisher, the subscriber, and the broker. The publisher sends data messages to a specific topic, while subscribers receive messages by subscribing to those topics. The MQTT broker acts as an intermediary that manages message distribution between publishers and subscribers. This decoupled architecture allows devices to communicate without direct knowledge of each other, improving scalability and flexibility.
- **Why MQTT?**

  MQTT is its minimal data packet size, which significantly reduces bandwidth usage and power consumption. This makes it particularly suitable for embedded systems and wireless IoT devices such as microcontrollers. Additionally, MQTT supports retained messages and persistent sessions, enabling devices to receive the latest data even after reconnecting to the network. In the proposed Shelf Updation System, MQTT is used to transmit shelf data from the cloud platform to the gateway ESP32 (ESP32-B). When shelf information such as product price, discount and quantity is updated on the cloud, the data is published to a specific MQTT topic. The gateway ESP32 subscribes to this topic and receives the updated information in real time.

  The cloud publishes updated shelf data to MQTT topics, which are subscribed to by the gateway ESP32 (ESP32-B). MQTT's lightweight protocol design ensures low bandwidth usage and efficient communication for the proposed IoT system.

2.3. ESP32

The ESP32 microcontroller is a powerful and versatile system-on-chip widely used in Internet of Things (IoT) applications due to its integrated Wi-Fi and Bluetooth capabilities, low power consumption, and high processing performance.

2.3.1. ESP32 as Gateway (ESP32-B)

In the proposed system, the ESP32 functions as the gateway controller (ESP32-B), responsible for receiving updated shelf data from the cloud platform using the MQTT protocol. This ESP32 operates as an edge gateway device that maintains Wi-Fi connectivity and ensures reliable communication with the MQTT broker.

- **Wi-Fi Capability:**

  The built-in Wi-Fi capability of the ESP32 enables direct connectivity to the internet without the need for external network modules. This allows the gateway ESP32 to establish a stable connection with the cloud platform and communicate with the MQTT broker. Through Wi-Fi connectivity, the ESP32 can continuously receive updated shelf information from the cloud in real time, ensuring synchronized data delivery across the system.
- **MQTT Client:**

  As an MQTT client, the gateway ESP32 subscribes to predefined MQTT topics on the broker. Whenever shelf data such as product price or description is updated on the cloud, the corresponding message is published to these topics. The ESP32-B receives these messages instantly

and processes the data for further transmission. The lightweight nature of the MQTT protocol ensures minimal bandwidth usage and efficient

communication, making it suitable for embedded gateway applications.

- **Bridge between cloud and shelf node:**

  The gateway ESP32 acts as a bridge between the cloud and the shelf node ESP32 devices. After receiving data via MQTT, the gateway forwards the processed information to the shelf nodes using Bluetooth Low Energy communication through the HM-10 module. This gateway-based architecture enables separation of cloud connectivity and display control, improving system modularity and scalability.

2.3.2. ESP32 as Shelf Node Controller (ESP32-A)

In the proposed system, the ESP32 functions as the shelf node controller (ESP32-A), responsible for receiving shelf data, processing it, and updating the e-paper display. This ESP32 operates as an edge device located at the shelf level and plays a crucial role in ensuring accurate and timely display of product information.

- **BLE Communication:**

  The shelf node ESP32 communicates with the gateway ESP32 using Bluetooth Low Energy (BLE). BLE is chosen for its low power consumption, short-range reliability, and suitability for battery-operated embedded systems. Through BLE communication, ESP32-A receives updated shelf data forwarded by the gateway, enabling efficient wireless communication within the store environment.

- **E-Paper display Control:**

  The ESP32-A is also responsible for controlling the Waveshare 2.9-inch e-paper display. It interfaces with the display using the Serial Peripheral Interface (SPI) protocol, sending the processed data to update the display content. Due to the bi-stable nature of e-paper technology, the display requires power only during content refresh, making the system highly energy-efficient.

- **Data Processing:**

  The ESP32-A performs necessary data processing tasks such as data validation, formatting, and parsing. The processed information is structured according to the display requirements, ensuring correct representation of product details such as price, name, and availability. Local data handling at the shelf node minimizes communication overhead and enhances system responsiveness.

2.4. Bluetooth Low Energy (BLE):

- **What is BLE?**

  Bluetooth Low Energy (BLE) is a wireless communication technology designed for short-range data transmission with extremely low power consumption. Unlike classic Bluetooth, BLE is optimized for applications that require periodic data exchange while maintaining long battery life. Due to these characteristics, BLE is widely used in Internet of Things (IoT) and embedded systems.

- **Why BLE?**

  The use of BLE ensures efficient short-range communication with minimal energy consumption, which is essential for battery-operated shelf

  nodes. The lightweight data transmission over BLE complements the low-power nature of the e-paper display, contributing to the overall energy efficiency of the system.

  In the proposed Shelf Updation System, BLE communication is employed to transfer shelf data from the gateway ESP32 (ESP32-B) to the shelf node ESP32 (ESP32-A). After receiving updated shelf information from the cloud via MQTT, the gateway forwards the data wirelessly using BLE. This eliminates the need for wired connections between the gateway and shelf nodes, reducing installation complexity and improving system flexibility.

2.5. E-paper Display Technology

- **Working Principle:**

  E-paper display technology, also known as electronic ink display technology, is a low-power display solution designed to mimic the appearance of traditional printed paper. Unlike conventional LCD or LED displays, e-paper displays are bi-stable, meaning they consume power only during content updates and retain the displayed image even when power is removed. This characteristic makes e-paper displays highly suitable for applications that require infrequent updates and long-term information display.

- **Characteristics**:

  E-paper displays offer excellent readability under ambient and direct lighting conditions, including sunlight, due to their reflective nature. They provide a wide viewing angle and high contrast, making text and images easily visible from different perspectives.

- **Waveshare 2.9-Inch E-Paper Display in the Proposed System:**

  The Waveshare e-paper display interfaces with the shelf node ESP32 using the Serial Peripheral Interface (SPI) communication protocol. The ESP32 controls the display update process by sending formatted data to refresh the display content. Since updates occur only when new data is received from the cloud, the display remains static for long periods without consuming power.

2.6. Electronic Shelf Label System:

- **What is Electronic Shelf Updation System:**

  Electronic Shelf Label (ESL) systems are digital display solutions used in retail environments to present product information such as price, name, barcode, and promotional details directly on store shelves.

  ESL systems replace traditional paper-based labels and enable automatic updates through centralized control mechanisms. These systems improve operational efficiency by eliminating manual label replacement and ensuring consistent information across all shelves.

- **Working:**

  ESL system consists of a central management platform, wireless communication infrastructure, and display-equipped shelf nodes. The central platform manages product data and sends updates to shelf labels using wireless communication technologies. Shelf nodes receive this data and update the displayed information accordingly. This architecture allows real-time synchronization between backend inventory systems and shelf-level displays.

2.7. Cloud and Shelf Data:

The selection of Adafruit IO aligns with the project's objectives of simplicity, reliability, and efficient real-time data updates. It enables fast deployment, reduces development effort, and provides sufficient performance for transmitting shelf data such as product name, quantity, price, and offers. Thus, Adafruit IO serves as a practical and effective cloud solution for demonstrating an IoT-based electronic shelf labeling system.

- **What is Cloud Data Management?**

  Cloud platforms play a critical role in IoT systems by enabling centralized data storage, remote monitoring, and real-time device control. In smart retail applications, cloud-based data management allows shelf information to be updated from a single interface and reflected instantly across multiple display units. This eliminates the need for physical access to shelf devices and ensures consistent information delivery.

- **Shelf Data Representation in the Proposed System**

  In the proposed E-Paper Display Based Shelf Updation System, the cloud platform is used to store and manage product-related data such as product name, quantity, price, and offer details. This structured data is updated by the user through a cloud dashboard and transmitted to the gateway ESP32 using the MQTT protocol. Centralized data handling ensures accuracy and synchronization between backend data and shelf displays.

- **Adafruit IO as the Cloud Platform**

  Adafruit IO is selected as the cloud platform for this project due to its simplicity, ease of integration, and suitability for small- to medium-scale IoT applications. It provides built-in support for MQTT communication, allowing seamless data exchange with ESP32 devices. Adafruit IO offers an intuitive dashboard for real-time data updates, making it ideal for rapid development and demonstration of IoT-based shelf management systems.

- **Why Adafruit IO over AWS IOT Core?**

  AWS IoT Core is widely preferred for large-scale industrial IoT applications, it introduces higher complexity in terms of configuration, security policies, device certificates, and cloud resource management. For a prototype-level and educational implementation such as the proposed shelf updation system, AWS IoT Core may result in unnecessary overhead. Adafruit IO, on the other hand, offers a lightweight and developer-friendly environment that meets the functional requirements of the project without additional complexity.

2.8. Data Flow and Communication Model

- **Data Flow in the Proposed System:**

  The data flow in the proposed E-Paper Display Based Shelf Updation System follows a structured and hierarchical communication model. The system is designed to enable centralized management of shelf information through a cloud platform while ensuring efficient and reliable communication with shelf display units. A gateway-based architecture is employed to manage data transfer between the cloud and shelf nodes.

- **Cloud-to-Gateway Communication Using MQTT:**

  Shelf information such as product name, quantity, price, and offer details is updated on the cloud platform. This data is published to predefined MQTT topics. The gateway ESP32 (ESP32-B) subscribes to these topics and receives the updated data in real time using Wi-Fi connectivity. The MQTT protocol ensures lightweight, low-latency, and bandwidth-efficient communication between the cloud and the gateway device.

- **Gateway-to-Shelf Node Communication Using BLE:**

  After receiving data from the cloud, the gateway ESP32 forwards the information to the shelf node ESP32 (ESP32-A) using Bluetooth Low Energy (BLE) communication. The built-in BLE capability of the ESP32 eliminates the need for an external BLE module and reduces system complexity. BLE is well suited for short-range communication within indoor retail environments due to its low power consumption and reliable data transfer.

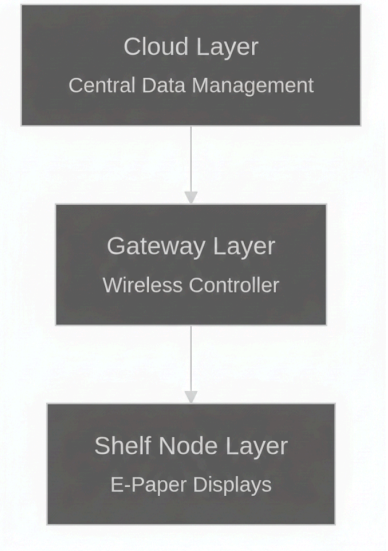
- **Shelf Node Data Processing and Display Update:**

  The shelf node ESP32 receives the transmitted data over BLE and performs necessary data processing such as validation and formatting. Once processed, the shelf information is sent to the e-paper display for updating the content. Since the e-paper display consumes power only during refresh operations, the display remains static for long durations without additional power usage.

CHAPTER 3: SYSTEM ARCHITECTURE AND DESIGN

3.1 Overall System Architecture:

The E-Paper Display Based Shelf Updation System follows a layered and modular architecture designed to enable real-time, wireless, and energy-efficient shelf updates. The system consists of a cloud layer, a gateway layer, and a shelf node layer. Shelf data such as product name, price, quantity, and offers are managed centrally on the cloud and transmitted to the shelf-level devices through a gateway controller. This architecture ensures centralized control, reduced manual intervention, and synchronized updates across multiple shelf units.



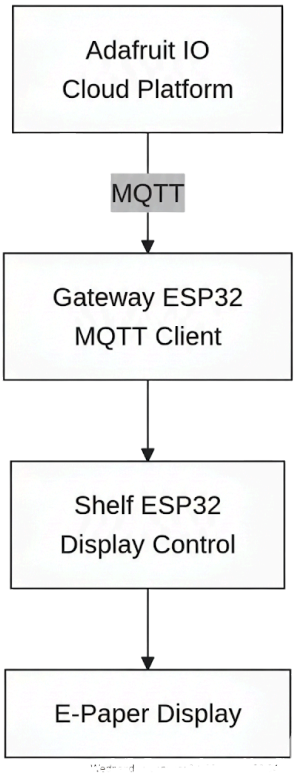
The diagram shows a three-layer architecture for the shelf update system. At the top is the cloud layer managing product data centrally. In the middle sits the gateway layer for wireless communication. At the bottom are the shelf nodes with e-paper displays. Arrows show data flowing downward from cloud to displays. This structure ensures centralized control and real-time updates.

3.2 Hardware Components

1. **ESP32 Gateway Node:** Connects to cloud via Wi-Fi and routes data to shelf nodes.
2. **ESP32 Shelf Node:** Receives update data and controls the e-paper display.
3. **Waveshare 2.9" E-Paper Display:** Shows product information with low power and high visibility.

3.3 Software Architecture:

The architecture uses cloud services and ESP32 firmware for modular, scalable communication. Adafruit IO publishes shelf data via MQTT; the gateway ESP32 subscribes to updates, while shelf nodes process data and drive e-paper displays. This separation ensures maintainability and independent updates.

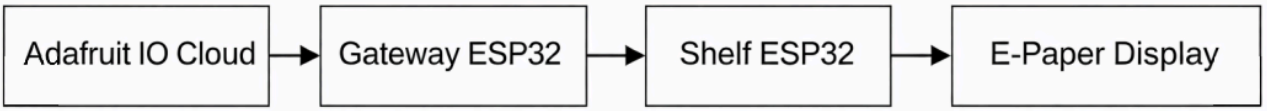


3.4. Communication Protocol Design:

- **Cloud to Gateway Communication:**
  MQTT over Wi-Fi is used between Adafruit IO and the gateway ESP32. Shelf data is published to specific MQTT topics, and the gateway subscribes to these topics to receive updates in real time. MQTT is chosen for its lightweight nature and low bandwidth usage.

- **Gateway to Shelf Node Communication:**
  The gateway ESP32 forwards received shelf data to the shelf node ESP32. This design reduces direct cloud dependency for shelf-level operations and improves system responsiveness.

- **Shelf Node to Display Communication:**
  The shelf node ESP32 communicates with the e-paper display using the SPI protocol. SPI ensures fast and reliable data transfer required for display updates.


Adafruit IO Cloud
Gateway ESP32
Shelf ESP32
E-Paper Display

CHAPTER 4: HARDWARE IMPLEMENTATION

This chapter describes the practical implementation of the proposed E-Paper Display Based Shelf Updation System. It focuses on the hardware design, circuit connections, and interfacing of system components required for real-time shelf information updates. The chapter explains how the ESP32 microcontrollers, communication interfaces, and the e-paper display are physically connected and configured to realize the system architecture discussed in the previous chapter. Detailed circuit diagrams and component-level explanations are provided to ensure clarity and reproducibility of the design.

4.1. Hardware Requirements:

Component	Specification
ESP32 Development Board	Dual-core MCU with Wi-Fi and Bluetooth support
Waveshare E-Paper Display	2.9-inch monochrome e-paper display
Connecting Wires	Jumper wires for SPI and power connections
USB Cable	For programming and power

4.2. Hardware Setup Overview

The hardware setup of the E-Paper Display Based Shelf Updation System is designed to support reliable wireless communication, low power operation, and efficient display control. The system consists of a gateway controller, a shelf node controller, an e-paper display module, and a suitable power supply. Each component plays a specific role in ensuring seamless data transfer from the cloud to the shelf display.

- **Gateway ESP32:**

  The gateway ESP32 acts as the central communication unit of the system. It connects to the cloud platform using Wi-Fi and subscribes to MQTT topics to receive updated shelf information. Upon receiving data, the gateway processes the messages and forwards the relevant information to the shelf node ESP32. This separation of gateway and shelf node functions improves scalability and reduces direct cloud dependency at the shelf level.

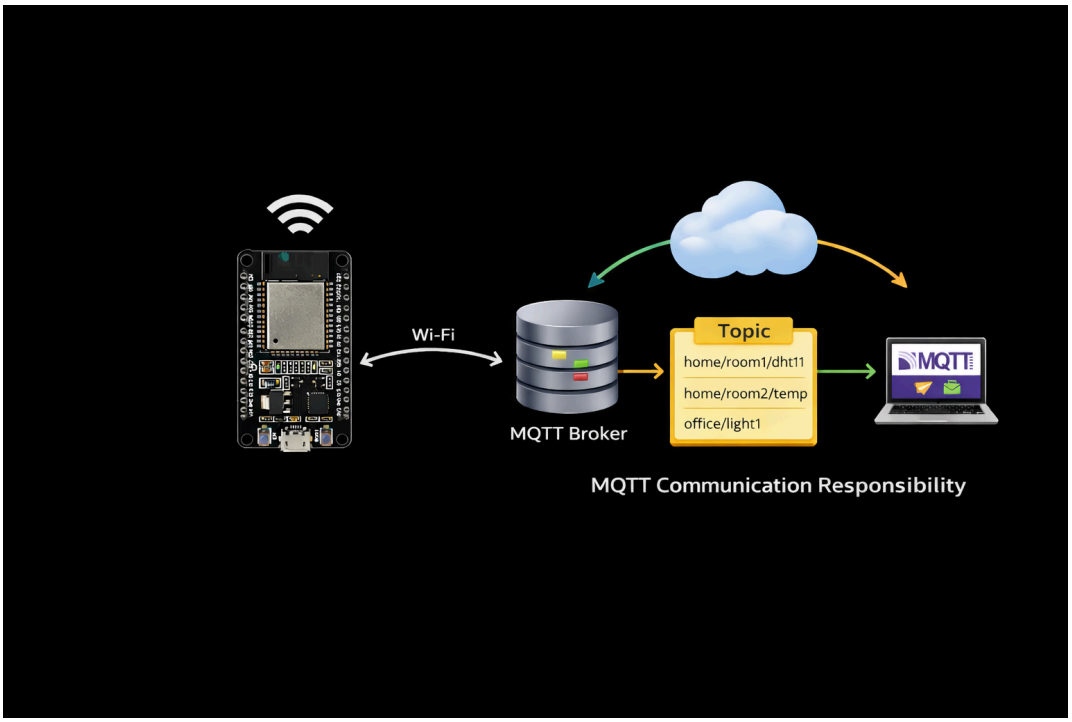
- **Shelf Node ESP32:**

The shelf node ESP32 is positioned at the shelf level and is responsible for processing received data and controlling the e-paper display. It handles data parsing, display formatting, and update operations. The shelf node operates in low-power modes when idle and wakes up only when new data is received, ensuring energy-efficient operation suitable for continuous deployment.

- **E-Paper Display:**

A Waveshare 2.9-inch e-paper display is used to present shelf information such as product name, price, quantity, and offers. The display is interfaced with the shelf node ESP32 using the SPI communication protocol. Due to its bi-stable nature, the e-paper display consumes power only during updates and retains the displayed content without continuous power, making it ideal for electronic shelf labeling applications.

4.3 Circuit Diagram of ESP32 Gateway (ESP32-B)



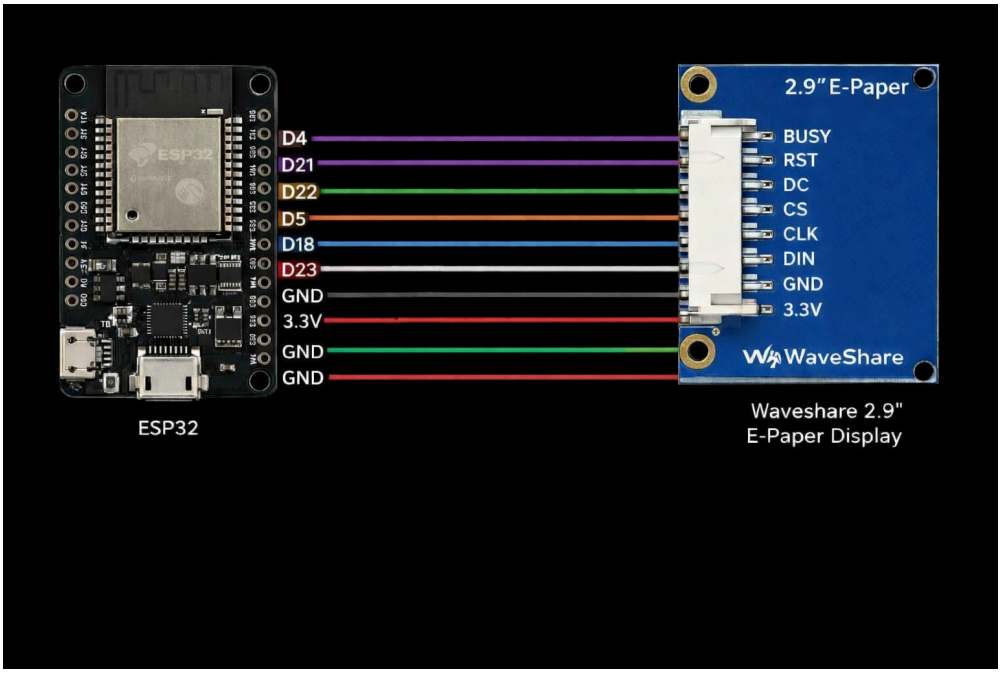
- **Wi-Fi Communication Role**

1. The ESP32 utilizes its built-in Wi-Fi module to connect directly to the cloud platform.
2. It establishes and maintains a stable internet connection for receiving real-time shelf updates.
3. No external Wi-Fi modules are required, reducing hardware complexity.

- **MQTT Communication Responsibility**

1. The gateway ESP32 acts as an MQTT client subscribing to specific topics on the cloud platform.
2. Shelf data including product name, price, quantity, and offers is received in structured MQTT messages.
3. Upon reception, the data is processed and forwarded to the shelf node ESP32 for display updates.

4.4. Circuit Diagram of Shelf Node ESP32 (ESP32-A)



Above fig. illustrates the interfacing of the ESP32 shelf node controller with the Waveshare 2.9-inch e-paper display using the Serial Peripheral Interface (SPI) protocol. The ESP32 controls the display by transmitting command and data signals required for refreshing the displayed shelf information.

ESP32 Pin	E-Paper Pin	Signal Type	Role / Function
GPIO 23 (D23)	DIN	SPI Data (MOSI)	Transfers display data and commands from ESP32 to e-paper display.
GPIO 18 (D18)	CLK	SPI Clock	Provides clock signals to synchronize SPI data transfer.
GPIO 5 (D5)	CS	Chip Select	Enables communication with e-paper display.
GPIO 22 (D22)	DC	Data/Command	Differentiate between display data and command instruction.
GPIO 21 (D21)	RST	Reset	Resets the e-paper display during initialization
GPIO 4 (D4)	BUSY	Status Signal	Indicates whether the display is busy during refresh
3.3V	3.3V	Power	Supplies operating voltage to the e-paper display
GND	GND	Ground	Provides common ground reference

- **Hardware-Level Operation:**
    1. The shelf node ESP32 remains in low-power mode while retaining previously displayed data on the e-paper display.
    2. Upon receiving updated shelf data, the ESP32 wakes up and initializes the SPI interface.
    3. The display is reset using the RST signal and checked for readiness through the BUSY pin.
    4. Command and data signals are transmitted to the e-paper display via SPI lines.
    5. The display refreshes the content and retains the updated information.
    6. After completion, the ESP32 returns to low-power mode to conserve energy.

4.5 Interfacing ESP32 with E-Paper Display

1. The ESP32 shelf node controller interfaces with the Waveshare 2.9-inch e-paper display using the Serial Peripheral Interface (SPI) communication protocol.
2. The ESP32 functions as the SPI master, while the e-paper display operates as the SPI slave device.
3. SPI communication lines DIN (MOSI) and CLK (SCK) are used to transmit display data and synchronize communication.
   Control signals CS, DC, RST, and BUSY are used to manage display selection, command/data differentiation, reset operations, and display status monitoring.

4. During initialization, the ESP32 resets the display and configures the SPI interface for proper operation.
5. Shelf data received by the ESP32 is formatted and transmitted to the display command and data sequences.
6. The BUSY pin is monitored to ensure the display completes its refresh cycle before further actions are taken.
7. After the update, the e-paper display retains the content without continuous power consumption.
8. The ESP32 then enters a low-power or idle state until the next update is received.

CHAPTER 5: SOFTWARE IMPLEMENTATION

The software implementation of the E-Paper Display Based Shelf Updation System requires a combination of development tools, platform support packages, and libraries to enable wireless communication, cloud integration, display control, and system reliability.

5.1. Software Requirements

Tools/Libraries	Specification
Arduino IDE	Used as the primary development environment for writing, compiling, and uploading firmware to the ESP32 microcontrollers**.**
Wi-Fi Library	Enables the gateway ESP32 to connect to the wireless network and communicate with the cloud platform.
MQTT Library (PubSubClient)	Used to implement MQTT publish–subscribe communication between the gateway ESP32 and the cloud (Adafruit IO). It supports lightweight, low-bandwidth data transfer suitable for IoT applications.
BLE Library (ESP32 BLE Arduino)	Enables Bluetooth Low Energy communication between the gateway ESP32 and the shelf node ESP32 for transmitting shelf update data.
GxEPD2 Library	Used to control the Waveshare 2.9-inch e-paper display. This library supports display initialization, text rendering, and partial or full refresh operations.
Adafruit GFX Library	Provides graphical functions such as text formatting, font rendering, and layout management for displaying shelf information clearly on the e-paper display.
Preferences	Used for non-volatile storage (NVS) to retain shelf data or system parameters across resets and power cycles.
Arduino OTA Library	Enables Over-The-Air firmware updates for the gateway ESP32, allowing software upgrades without physical access to the device.

5.2. Gateway ESP32

The gateway ESP32 firmware plays a central role in the proposed shelf updation system by acting as an interface between the cloud platform and the shelf node controllers. It is responsible for establishing network connectivity, managing cloud communication, handling user interaction through a web interface, and forwarding shelf data to the shelf node.

- **Wi-Fi Initialization and Network Connection**

  The gateway ESP32 initializes its Wi-Fi module at startup and connects to a preconfigured wireless network. The firmware continuously monitors the connection status and automatically reconnects in case of network failure, enabling reliable cloud communication.

- **MQTT Setup and Cloud Subscription**

  After establishing Wi-Fi connectivity, the gateway initializes an MQTT client and connects to the Adafruit IO broker. It subscribes to predefined topics carrying shelf information such as product name, price, quantity, and offers. MQTT messages are handled asynchronously to support real-time updates.

- **Receiving and Storing Shelf Data**

  Incoming MQTT data is parsed by the gateway firmware and stored in non-volatile storage (NVS) using the Preferences library. This ensures that shelf data is retained even after resets or power interruptions.

- **Trigger-Based Data Transmission to Shelf Node**

  Upon activation, the gateway establishes a BLE connection with the shelf node ESP32 and transmits the stored shelf data as a structured payload.

- **Web Dashboard and Local Control**

  The gateway hosts a lightweight web dashboard that allows users to monitor shelf data and manually trigger updates. The dashboard can be accessed using the device's local IP address or hostname.

- The gateway is designed to connect the ESP32 to a Wi-Fi network and an MQTT broker (Adafruit IO). Once connected, it subscribes to shelf-related feeds. When updated data is received (product info), the firmware parses and stores it. Upon a trigger (e.g., "push to shelf"), it scans for BLE devices and forwards the data. The gateway also hosts a web dashboard for manual control and supports OTA updates for future upgrades

5.3. Shelf Node ESP32

The shelf node ESP32 firmware is responsible for receiving updated shelf information from the gateway, processing the data, and updating the e-paper display efficiently. The firmware is optimized for low power operation and reliable display control.

- **BLE Communication and Data Reception**

  The shelf node initializes its Bluetooth Low Energy (BLE) module and waits for incoming connections from the gateway ESP32. Upon successful connection, it receives the shelf update payload containing product name, price, quantity, and offer details.

- **Data Parsing and Processing**

  The received BLE data is parsed and validated by the firmware. The extracted information is formatted into a suitable structure required for display rendering. This ensures consistent and accurate presentation of shelf information.

- **E-Paper Display Update**

  The shelf node controls the Waveshare 2.9-inch e-paper display using the SPI protocol. The firmware initializes the display, sends the formatted data as command and data sequences, and monitors the BUSY signal to ensure proper synchronization during refresh operations.

- **Low-Power Operation**

  After completing the display update, the shelf node ESP32 enters a low-power or idle state. Since the e-paper display retains content without power, the system minimizes energy consumption and supports long-term shelf deployment.

- **Overall Firmware Operation**

  The shelf node firmware operates in an event-driven manner, remaining inactive until new data is received. This design ensures energy efficiency, reliable updates, and stable shelf display performance.

- The shelf node firmware continuously scans for BLE data from the gateway. When a valid update arrives, the ESP32 parses the structured payload (product name, price, quantity, and discount) and formats it for display. Using the GxEPD2 and Adafruit GFX libraries, the information is drawn to the e-paper. After the update, the node enters deep sleep to conserve power, waking only on BLE trigger or scheduled RTC interval.

5.4 MQTT Topic Structure and Data Format

The communication between the cloud platform and the gateway ESP32 is implemented using the MQTT publish–subscribe model. Shelf-related information is published to predefined MQTT topics on Adafruit IO, enabling structured and efficient data transfer.

- **MQTT Topic Structure**

  Typical topics include:

  1. Product name
  2. Price
  3. Quantity
  4. Offer

  This topic-based approach allows individual shelf parameters to be updated independently without affecting other data fields.

- **Data Format**

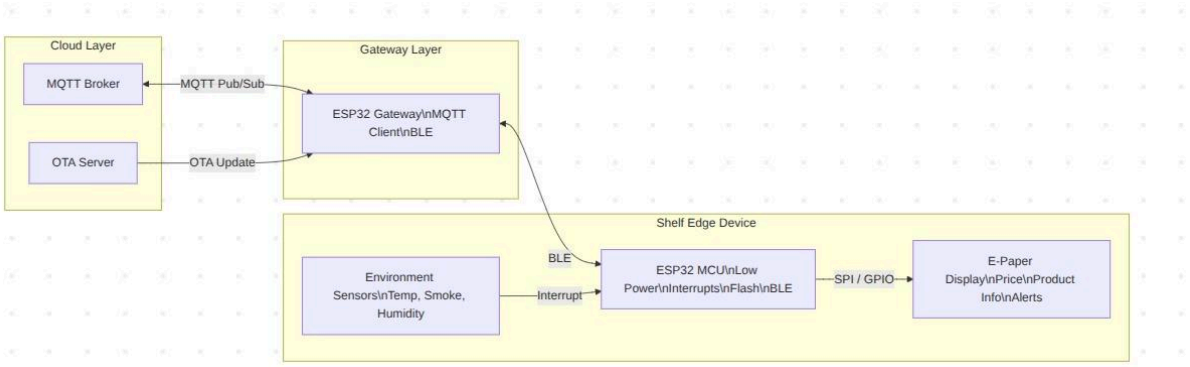
  The data received from the cloud is transmitted as simple text-based values. Upon receiving updates on the subscribed topics, the gateway firmware parses and stores the values in non-volatile storage. When an update trigger is activated, the stored data is combined into a structured payload and transmitted to the shelf node ESP32.

- The gateway subscribes to Adafruit IO feeds such as *shelf-prod* for product name, *shelf-price* for price, *shelf-qty* for quantity, and *shelf-disc* for discount. Once these are all received, and a push command is triggered via *shelf-send*, the gateway constructs a BLE payload.

CHAPTER 6: SYSTEM ARCHITECTURE AND DATA FLOW

This chapter provides a structured overview of the cloud layer, gateway unit, and shelf edge devices, along with their functional roles and interactions. It explains the communication mechanisms, including MQTT, BLE, and SPI, used for reliable data exchange and control. Additionally, power management strategies, event-driven data flow, and OTA update mechanisms are discussed to demonstrate system efficiency and scalability.

6.1. Overall System Architecture



This diagram presents the high-level architecture of the system. The system is divided into three main layers:

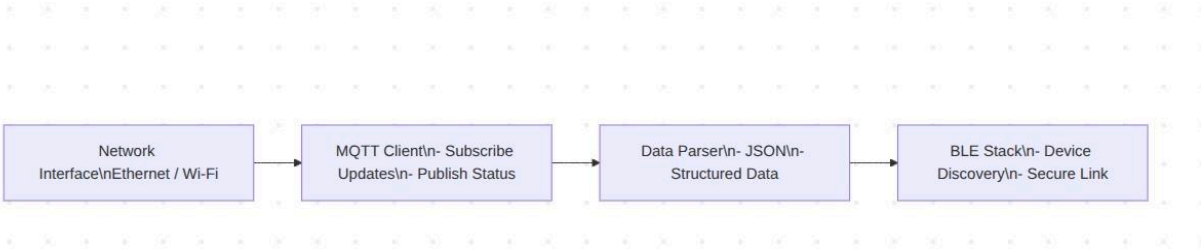
1. **Cloud Layer**
   Hosts the MQTT broker and OTA server. It manages product data, pricing updates, alerts, and firmware updates.
2. **Gateway Layer**
   An ESP32 gateway that connects to the cloud using MQTT over Wi-Fi/Ethernet and communicates with shelf devices using BLE.
3. **Shelf Edge Device**
   A low-power ESP32 device installed on each shelf, responsible for sensor monitoring and E-paper display updates.
   The gateway acts as an intermediary, reducing power consumption on shelf units and improving scalability.

6.2. Cloud Layer Architecture

The cloud layer is responsible for centralized control and data management:

- **MQTT Broker**
  Implements a publish/subscribe mechanism to distribute product updates, alerts, and acknowledgments.
- **OTA Server**
  Stores firmware binaries and manages version control for remote firmware updates.
- **Data Storage**
  Maintains product information, pricing data, and configuration parameters. This architecture enables remote updates, centralized monitoring, and easy system maintenance.

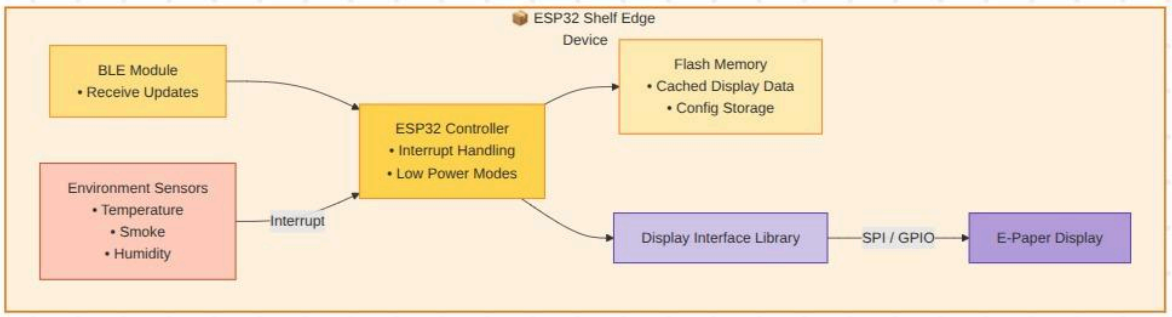
6.3. ESP32 Gateway Functional Architecture



The ESP32 gateway bridges cloud communication and shelf devices:

- **Network Interface**
  Connects to the cloud via Wi-Fi or Ethernet.
- **MQTT Client**
  Subscribes to update topics and publishes device status messages.
- **Data Parser**
  Parses structured JSON data received from the cloud.
- **BLE Stack**
  Handles device discovery, pairing, and secure BLE communication with shelf units. The gateway offloads networking complexity from shelf devices, allowing them to operate in low-power modes.

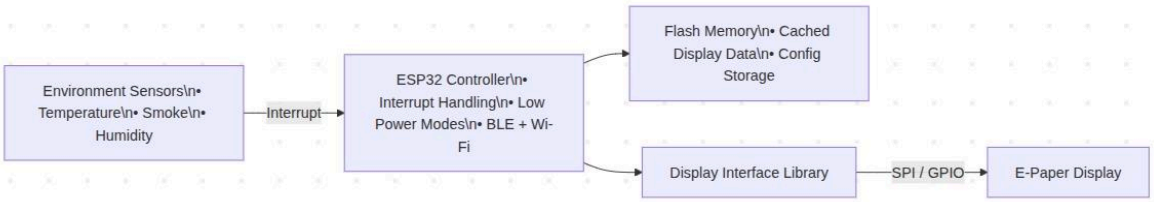
6.4. Shelf Edge Device Architecture



This diagram details the internal components of the shelf edge device:

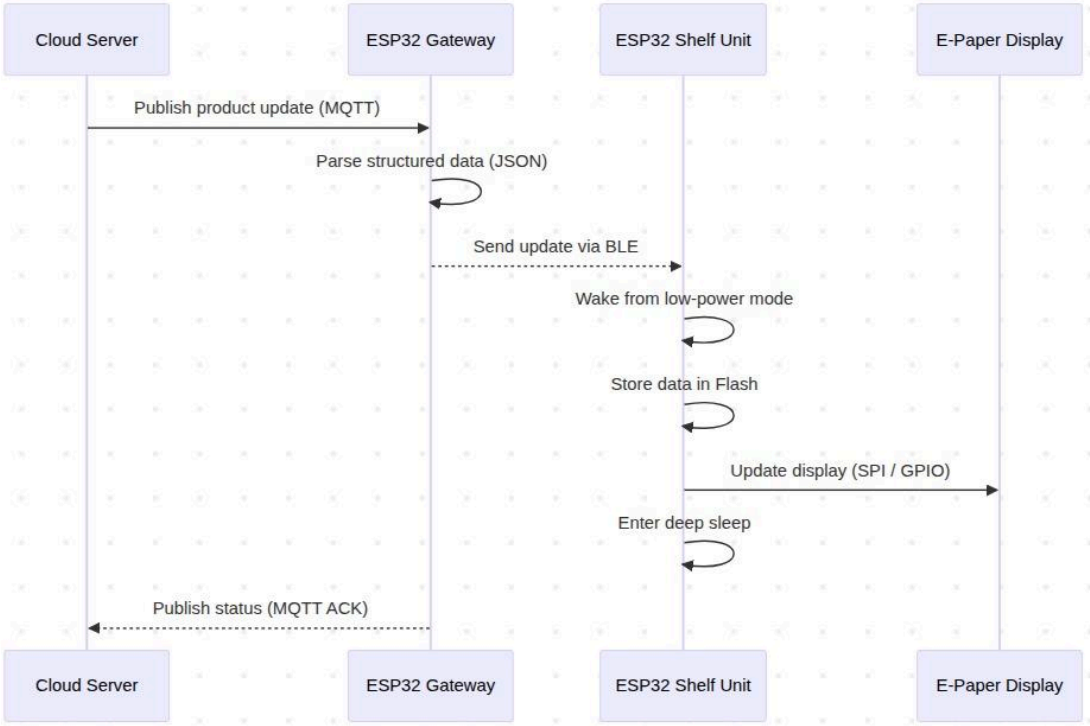
- **ESP32 Controller**
  Manages BLE communication, interrupt handling, and power modes.
- **Environmental Sensors**
  Measure temperature, smoke, and humidity levels.
- **Flash Memory**
  Stores cached display data and configuration parameters.
- **Display Interface Library**
  Handles communication with the E-paper display via SPI/GPIO.
- **E-Paper Display**
  Displays product price, information, and alerts with ultra-low power consumption.

6.5. Power Management and Interrupt Handling



- The system uses an interrupt-driven design to minimize power consumption:
  Shelf devices remain in deep sleep during idle conditions.
- Environmental sensors trigger an interrupt when a threshold is exceeded.
- The ESP32 immediately wakes up to process the event.
- This approach avoids continuous polling and significantly extends battery life.

6.6. Product and Price Update Data Flow



This diagram explains the normal update operation:

- Cloud server publishes a product or pricing update using MQTT.
- ESP32 gateway receives and parses the JSON payload.
- The update is transmitted to the shelf unit via BLE.
- The shelf unit wakes from low-power mode.
- Data is stored in flash memory.
- E-paper display is updated via SPI/GPIO.
- The shelf unit returns to deep sleep.
- Gateway publishes an acknowledgment to the cloud.

6.7. Firmware Update and OTA Mechanism

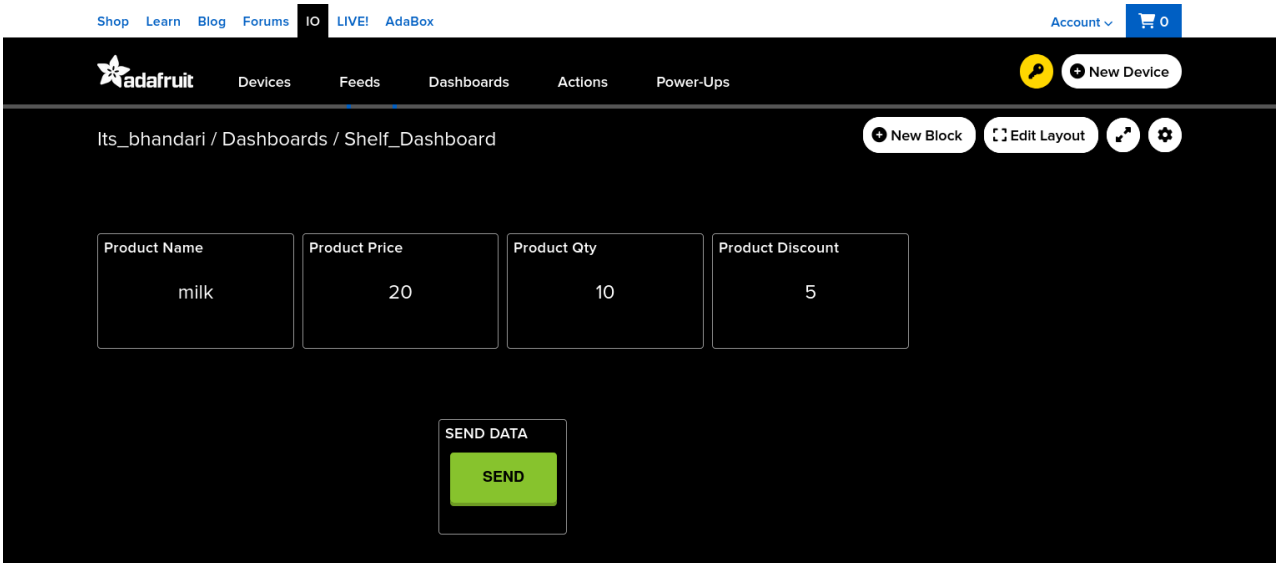
This diagram represents the OTA firmware update process:

- Firmware binaries are stored on the OTA server.
- Version information and update notifications are sent via MQTT.
- Devices download and apply firmware updates remotely.
- This eliminates manual intervention and ensures consistent firmware versions across devices.

CHAPTER 7: TOOLS AND RESULTS

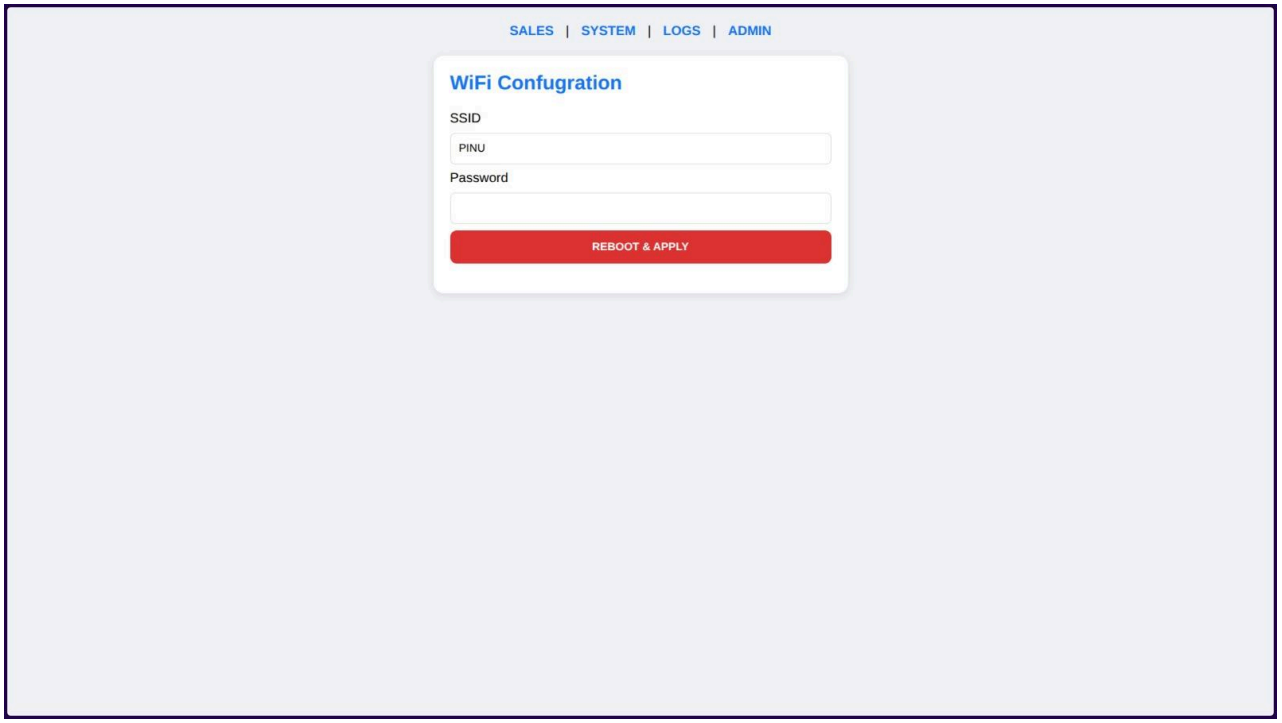
- **Tools used**
    1. Arduino IDE
    2. Adafruit IO

7.1. Cloud Dashboard for Shelf Data Update

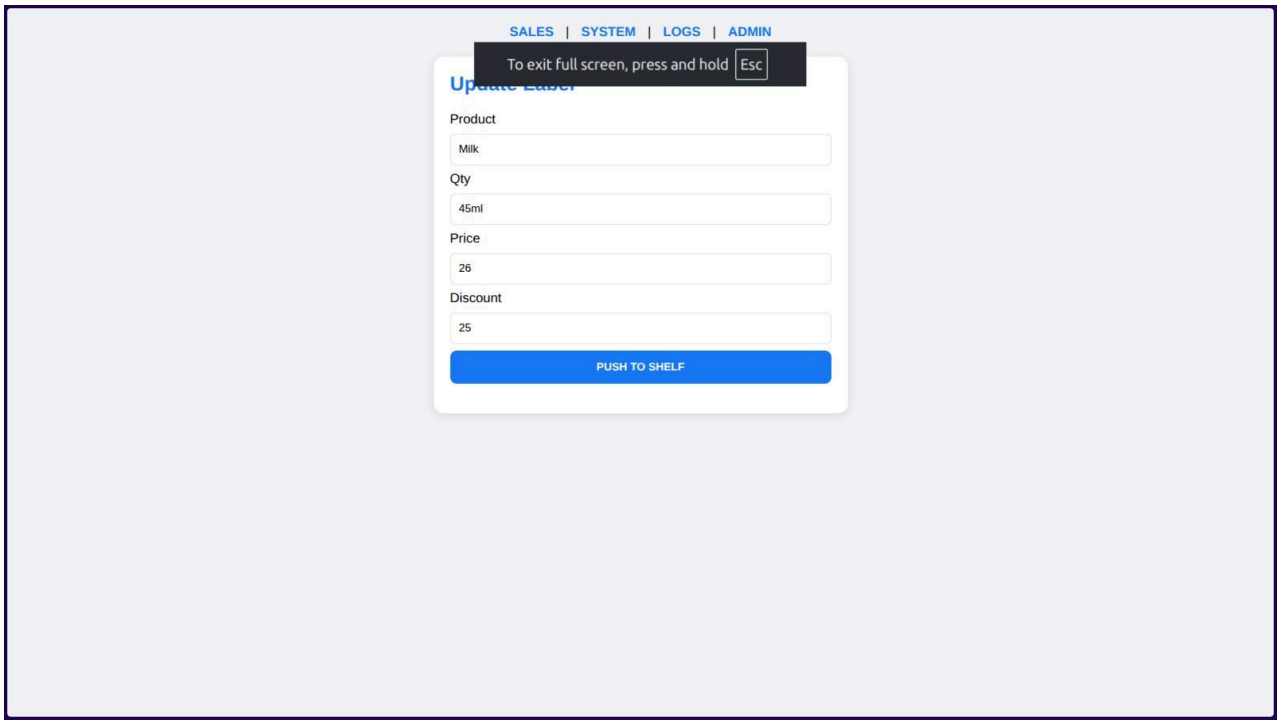


1. The above image shows the cloud-based dashboard used for updating shelf information in the proposed system.
2. The dashboard is developed using an IoT cloud platform and provides a user-friendly interface to modify product-related details such as product name, price, quantity, and discount.
3. Each parameter is represented using individual input blocks, allowing authorized users to update shelf data remotely.
4. Once the user enters the required values and presses the **SEND** button, the data is published to predefined MQTT topics on the cloud broker.
5. This dashboard acts as the central control point for shelf information management, enabling real-time updates without manual intervention at the physical shelf location.
6. In this mode of operation, the updated shelf data entered on the cloud dashboard is transmitted to the gateway ESP32 (ESP32-B) via the MQTT protocol. The gateway subscribes to the relevant topics and receives the updated information instantly.

7.2. Local Data Update



1. One of the local data update screenshots represents the **network configuration mode** of the system. This mode is used when the device is connected to a new or unknown Wi-Fi network. In this mode, the ESP32 creates a local configuration interface that allows the user to enter the Wi-Fi **SSID and password** without modifying the firmware.
2. Once the credentials are entered, the ESP32 stores the network details and attempts to connect to the specified Wi-Fi network.
3. This mechanism eliminates the need for reprogramming the device whenever it is deployed in a different location, making the system flexible and user-friendly.

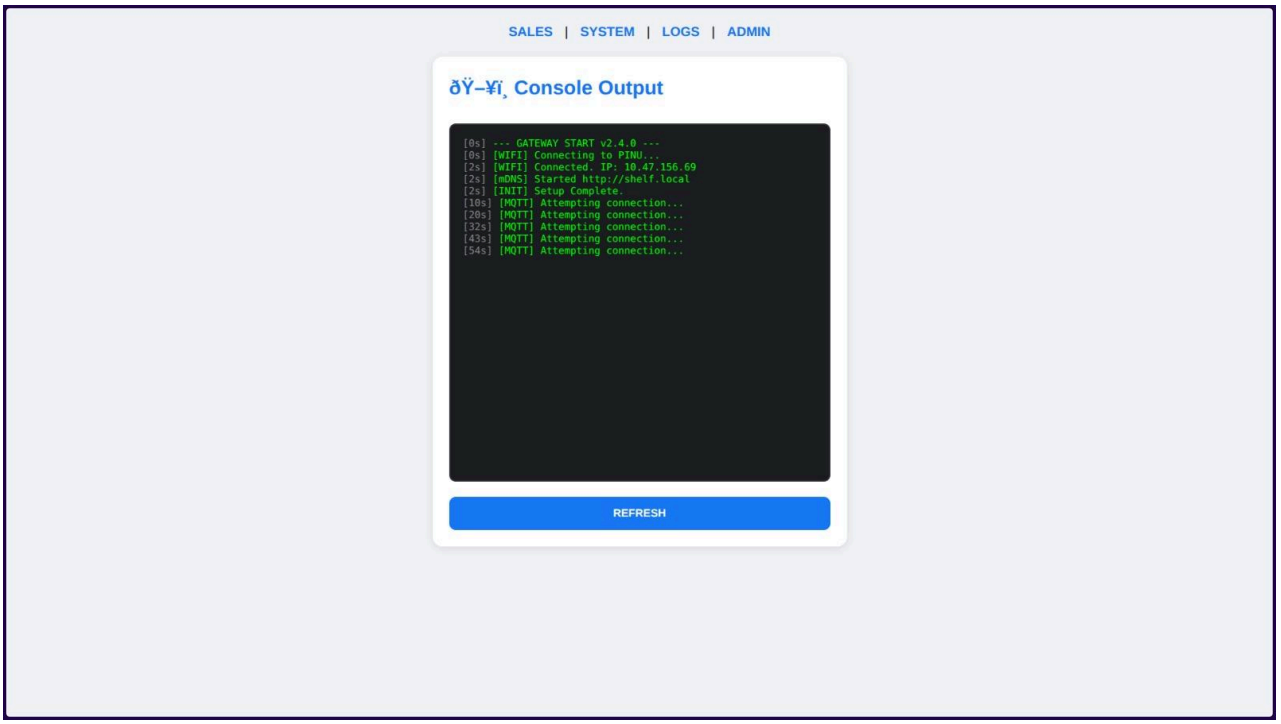


1. The second local update screenshot corresponds to the **local data push mode**, where shelf information such as product name, price, quantity, and discount is entered locally.

2. In this mode, the entered data is directly processed by the ESP32 and sent to the shelf node controller for updating the e-paper display. This ensures that the system remains operational even in offline scenarios and allows quick verification of display functionality during development.

- The system distinguishes between configuration data (Wi-Fi credentials) and operational data (shelf information). Wi-Fi credentials are processed and stored first to establish network connectivity. Once a stable connection is achieved, the system can either receive data from the cloud via MQTT or accept manual local data inputs for shelf updation.

7.3. System Logs



1. System logs are used to monitor and verify the real-time operation of the E-Paper Display Based Shelf Updation System.
2. The logs display information such as Wi-Fi connection status, successful MQTT subscription, reception of shelf data from the cloud, and Bluetooth Low Energy communication between the gateway ESP32 and shelf node ESP32.
3. Additionally, the logs confirm data processing and successful updates on the e-paper display. These logs are essential for debugging, system validation, and ensuring reliable end-to-end communication.

CHAPTER 8: CONCLUSION

This project successfully demonstrates the design and implementation of an E-Paper Display Based Shelf Updation System using ESP32 microcontrollers. The system addresses the limitations of traditional paper-based shelf labels by enabling real-time, remote, and automated updates of product information such as price, offers, and availability.

The proposed architecture utilizes a cloud–gateway–edge device model, where product data is managed centrally through a cloud platform and reliably delivered to shelf nodes via a gateway ESP32. Communication between the cloud and gateway is achieved using the MQTT protocol, while Bluetooth Low Energy (BLE) is used for short-range, low-power communication with shelf devices. The shelf node ESP32 interfaces with an e-paper display, ensuring low power consumption and high visibility.

Key features such as non-volatile data storage, web-based local control, OTA firmware updates, and mDNS-based access enhance the reliability, scalability, and maintainability of the system. Experimental results show that the system performs accurate and timely display updates while maintaining energy efficiency, making it suitable for smart retail and inventory management applications.

Overall, the project fulfills its objectives and demonstrates a practical, scalable, and cost-effective solution for smart shelf management in modern retail environments.

CHAPTER 9: FUTURE SCOPE

- **FOTA**
  In the future, the system can be enhanced by integrating Firmware Over-the-Air (FOTA) functionality. FOTA would allow remote firmware updates of both the gateway ESP32 and shelf node ESP32 without physical access. This enables easy deployment of bug fixes, feature updates, and performance improvements through the cloud. FOTA reduces maintenance effort and downtime, especially in large-scale deployments. It also improves system security by allowing timely firmware patches. Overall, FOTA would make the system more scalable, maintainable, and reliable for long-term use.

- **Multi-Shelf Expansion**
  The system can be extended to support a large number of shelf nodes connected to a single gateway, enabling full-store automation.

- **Enhanced Security Mechanisms**
  Implementation of encrypted MQTT communication (TLS/SSL), device authentication, and secure BLE pairing can improve system security.

- **Mobile Application Integration**
  A dedicated mobile application can be developed for real-time monitoring, control, and manual updates of shelf information.

- **Advanced Power Optimization**
  Further reduction in power consumption can be achieved by optimizing sleep cycles and adopting energy-harvesting techniques.

- **Analytics and Dynamic Pricing**
  Integration of analytics or AI-based modules can enable dynamic pricing, demand-based offers, and automated inventory alerts.