TECHNICAL PROJECT REPORT

IoT-ENABLED DIGITAL NOTICE BOARD

Realtime Smart Notice Management and
IoT Display System

Flutter | Firebase | ESP32 | HUB75E

Department of Electronics and Communication Engineering
ABVGIET, Pragati Nagar, Himachal Pradesh

Date of Release:
May 2026

ABSTRACT

The proposed “IoT-Enabled Digital Notice Board” system was developed to provide realtime digital communication using IoT technology, cloud computing, embedded systems, and mobile application development. Traditional paper-based notice boards suffer from several limitations such as delayed information updates, manual maintenance, excessive paper usage, and lack of remote accessibility. The proposed system overcomes these limitations by implementing a cloud-connected realtime digital notice management platform.

The system integrates ESP32 microcontroller, Firebase cloud services, Flutter mobile application framework, and HUB75E RGB LED matrix display technology. The Flutter mobile application enables authorized users to remotely create, manage, and publish notices through internet connectivity. Firebase Authentication provides secure user login and email verification functionality, while Cloud Firestore and Firebase Realtime Database provide cloud-based storage and realtime synchronization between the mobile application and ESP32 embedded controller.

The ESP32 continuously monitors Firebase Realtime Database through WiFi communication and updates the HUB75E RGB LED matrix display whenever new notice data or display settings are synchronized. The system also supports scrolling text display, brightness control, color customization, border configuration, realtime display synchronization, and automatic notice expiry handling using NTP time synchronization.

DMA-based rendering architecture was implemented to improve LED matrix refresh stability and smooth scrolling performance during simultaneous realtime cloud communication operations. The proposed system successfully achieved stable realtime synchronization, secure cloud communication, remote notice management, and efficient embedded display control.

The developed system provides a low-cost, scalable, paperless, and efficient digital communication solution suitable for educational institutions, offices, and public information display applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The rapid advancement of Internet of Things (IoT) technology and digital communication systems has significantly transformed the way information is managed and distributed in educational institutions, organizations, hospitals, transportation systems, and public environments. Traditional notice boards that depend on manually updated paper notices are still widely used for displaying announcements, schedules, circulars, and important information. However, such systems suffer from several limitations including delayed information updates, excessive paper usage, manual maintenance, lack of realtime communication, and absence of remote accessibility.

In modern communication environments, there is an increasing requirement for realtime information delivery systems capable of providing fast, efficient, and centralized communication. The development of wireless communication, embedded systems, cloud computing, and mobile application technologies has enabled the implementation of smart digital display systems that overcome the limitations of traditional notice boards.

IoT-based digital notice board systems provide centralized information management, wireless notice synchronization, remote accessibility, and realtime display capabilities through internet connectivity. These systems allow authorized users to remotely publish and manage notices using cloud-connected platforms while embedded display controllers automatically receive and display updated information in realtime.

The proposed **"IoT-Enabled Digital Notice Board"** system has been developed using ESP32 microcontroller, Firebase cloud services, Flutter mobile application framework, and HUB75E RGB LED matrix display technology. The system enables authorized users to remotely create, manage, and publish notices through a mobile application interface. Firebase cloud services are

used for secure authentication, cloud storage, and realtime communication between the mobile application and ESP32 embedded controller.

The ESP32 microcontroller continuously monitors Firebase Realtime Database through WiFi communication and updates the HUB75E RGB LED matrix display whenever new notice data is synchronized. The system also supports configurable display settings, scrolling text display, dynamic brightness control, border customization, realtime cloud synchronization, and automatic notice expiry handling using NTP time synchronization.

The proposed system demonstrates practical integration of IoT technology, embedded systems, cloud databases, and mobile application development into a low-cost, scalable, and efficient digital communication platform suitable for educational institutions and organizational environments.

1.2 Problem Statement

Traditional notice boards used in educational institutions, offices, and public environments primarily rely on manually updated paper notices for information dissemination. Such systems are inefficient, time-consuming, and incapable of providing realtime communication. Every notice update requires physical printing and manual replacement, resulting in operational delays and increased maintenance effort.

Conventional notice boards also suffer from several practical limitations such as excessive paper usage, limited visibility, difficulty in managing urgent announcements, and inability to remotely update displayed information. In large institutions, important notices may not reach the intended audience on time due to delays in manual notice handling and physical information distribution.

Existing digital display systems often depend on expensive hardware infrastructure, dedicated backend servers, or complex communication modules, increasing deployment cost and maintenance complexity. Some systems also lack efficient realtime synchronization and centralized cloud management capability.

Therefore, there is a requirement for a smart, low-cost, cloud-connected, and realtime digital notice board system capable of remotely managing and displaying notices through wireless communication with minimal operational complexity. The proposed "IoT-Enabled Digital Notice Board" system addresses these limitations through integration of IoT technology, Firebase cloud services, ESP32 embedded controller, and Flutter mobile application framework.

1.3 Objectives of the Project

The primary objective of the proposed system is to develop a realtime cloud-connected digital notice board using IoT and embedded communication technologies.

The major objectives of the project are as follows:

- To develop a wireless digital notice board using ESP32 and HUB75E RGB LED matrix display.
- To implement realtime cloud communication using Firebase services.
- To develop a Flutter-based mobile application for remote notice management.
- To implement secure user authentication and email verification.
- To establish realtime synchronization between Firebase and ESP32.
- To implement scrolling text displays for long notice messages.
- To provide configurable display settings such as brightness and scrolling speed.
- To implement automatic notice expiry handling using NTP time synchronization.
- To provide fallback clock mode operation when no active notice is available.
- To develop a scalable and modular system architecture for future expansion.

1.4 Scope of the Project

The scope of the proposed system includes the development of a cloud-connected embedded display platform capable of providing realtime digital notice broadcasting through wireless communication.

The project includes development and integration of the following major components:

- Flutter mobile application for notice management
- Firebase cloud integration
- ESP32 firmware development
- HUB75E RGB LED matrix display integration
- Realtime notice synchronization
- User authentication system
- Display configuration management
- Automatic notice expiry handling
- NTP-based clock synchronization

The developed system supports remote notice publishing through internet connectivity and realtime synchronization between Firebase cloud services and ESP32 embedded controller.

The current implementation is designed for a single ESP32-controlled LED matrix display system. However, the architecture can be extended in future for multiple display boards, centralized notice management, and larger institutional communication systems.

1.5 Proposed System Overview

The proposed “IoT-Enabled Digital Notice Board” system is designed as a smart realtime communication platform capable of remotely managing and displaying digital notices using IoT and cloud technologies.

The system consists of three major components:

1. Flutter Mobile Application
2. Firebase Cloud Services
3. ESP32 and HUB75E RGB LED Matrix Display System

The Flutter-based mobile application provides an interface through which authorized users can create, manage, and publish notices remotely. Firebase services are used for cloud-based data storage and realtime synchronization between the mobile application and embedded display controller.

A hybrid Firebase architecture has been implemented in which Cloud Firestore is used for permanent notice storage and historical data management, while Firebase Realtime Database is used for low latency realtime communication with the ESP32 microcontroller.

The ESP32 continuously monitors the cloud database through WiFi communication and updates the LED matrix display whenever new notice data is synchronized. The system supports scrolling text display, configurable brightness, display customization, border styles, realtime updates, and automatic notice expiry management.

An autonomous notice expiry mechanism has also been implemented using epoch timestamp comparison and NTP time synchronization. Expired notices are automatically removed from the display without requiring dedicated backend servers or manual intervention. When no active notice is available, the system automatically switches to realtime clock display mode.

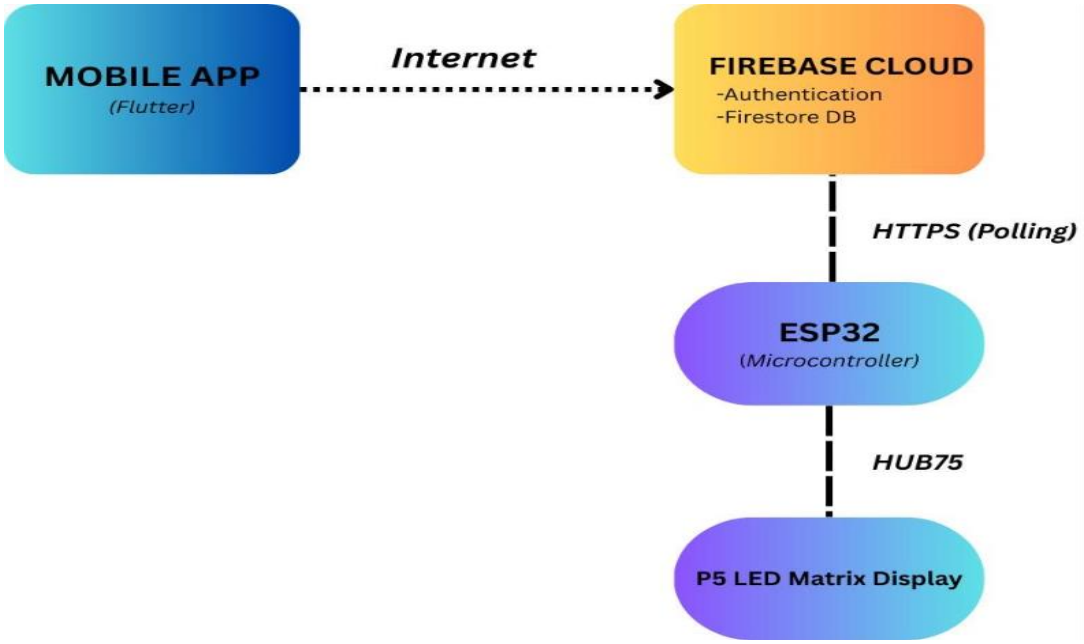


Figure 1.1 Overall Proposed system

1.6 Advantages of the System

The proposed system provides several advantages compared to traditional notice boards and conventional digital display systems.

The major advantages are:

- Realtime wireless communication
- Remote notice management
- Paperless information display
- Secure cloud-based authentication
- Low operational cost
- Centralized notice management
- Dynamic display customization
- Automatic notice expiry handling
- Realtime cloud synchronization
- Scalable and modular architecture
- Low-cost IoT implementation

1.7 Applications of the System

The proposed system can be used in various realtime information management environments such as:

- Schools and Colleges
- Offices and Organizations
- Hospitals
- Railway Stations
- Shopping Malls
- Libraries
- Smart Campus Systems
- Public Information Display Systems

1.8 Organization of the Report

The project report is organized into multiple chapters describing different stages of system development and implementation.

- **Chapter 1** presents the introduction, objectives, scope, and overview of the proposed system.
- **Chapter 2** presents a literature survey and comparative analysis of existing systems.
- **Chapter 3** explains system requirements and overall architecture of the proposed system.
- **Chapter 4** discusses hardware design and implementation of ESP32 and HUB75E display systems.
- **Chapter 5** explains software development, Firebase integration, and mobile application implementation.
- **Chapter 6** presents testing, validation, and performance analysis of the proposed system.
- **Chapter 7** presents the Conclusion and future Scope of the project.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Digital notice board systems have become an important communication solution for educational institutions, offices, transportation systems, hospitals, and public information environments. The rapid growth of wireless communication, embedded systems, cloud computing, and Internet of Things (IoT) technologies has significantly improved the efficiency of digital information management systems.

Traditional paper-based notice boards require manual updates and physical maintenance, resulting in delayed communication and increased operational effort. To overcome these limitations, several electronic notice board systems based on GSM, Bluetooth, Wi-Fi, and IoT technologies have been developed by researchers and industries.

This chapter presents a study of traditional notice board systems, existing wireless communication-based digital notice boards, IoT-enabled display systems, and related technologies used for realtime information management. A comparative analysis of different communication approaches is also presented to identify the advantages and limitations of existing systems and justify the development of the proposed system.

2.2 Traditional Notice Board Systems

Traditional notice boards are commonly used in schools, colleges, offices, and public institutions for displaying announcements, schedules, and important information. These systems rely on manually printed paper notices attached to physical boards.

Although traditional notice boards are simple and inexpensive, they suffer several limitations. Every notice update requires manual printing and physical replacement of existing notices. This process consumes time, paper resources, and human effort.

Traditional systems also lack realtime communication capability and remote accessibility. Important notices may not reach users immediately due to delays in manual updating and information distribution. In environments where notices change frequently, manual notice management becomes inefficient and difficult to maintain.

The limitations of conventional notice boards created the requirement for automated and wireless digital information display systems capable of providing realtime updates and centralized management.

2.3 GSM-Based Digital Notice Board Systems

GSM-based digital notice board systems were developed to provide remote notice updating capability using SMS communication. In such systems, GSM modules are connected with embedded controllers to receive notice data through mobile communication networks.

Whenever a user sends an SMS containing notice information, the GSM module receives the message and forwards the data to the embedded controller, which updates the display accordingly. These systems improved communication flexibility compared to traditional notice boards by enabling remote notice publishing without physical access to the display unit.

However, GSM-based systems have several disadvantages including communication delay, dependency on mobile network availability, SMS transmission cost, and limited scalability. Such systems also lack centralized cloud-based data management and efficient realtime synchronization capability.

The operational reliability of GSM-based systems is highly dependent on network signal strength and SMS delivery performance, making them less suitable for modern realtime communication environments.

2.4 Bluetooth-Based Digital Notice Board Systems

Bluetooth-based digital notice board systems use short-range wireless communication for transmitting notices between mobile devices and embedded display controllers.

In these systems, users connect their smartphones or computers directly to the embedded display controller through Bluetooth communication and send notice information wirelessly. Bluetooth communication eliminated SMS-related operational costs and improved local communication speed compared to GSM-based systems.

Although Bluetooth-based systems are suitable for small-scale applications, they suffer several limitations. The communication range of Bluetooth is limited, and users must remain physically close to the display system to update notices. These systems also lack internet-based remote accessibility and centralized cloud communication capability.

Bluetooth-based systems are therefore unsuitable for large-scale institutional communication environments where realtime remote information management is required.

2.5 IoT-Based Digital Notice Board Systems

IoT-based digital notice board systems use internet connectivity and cloud communication technologies for realtime notice synchronization and remote information management. These systems commonly integrate embedded controllers, wireless communication modules, cloud databases, and mobile applications to establish centralized digital communication platforms.

Modern IoT-based systems provide several advantages including:

- Realtime cloud synchronization
- Remote accessibility
- Centralized management
- Wireless communication
- Mobile application integration
- Dynamic display configuration

The ESP32 microcontroller has become widely used in IoT applications due to its integrated Wi-Fi capability, dual-core processing architecture, low cost, and compatibility with cloud communication libraries. Firebase cloud services are also commonly used in IoT systems because they simplify backend development and provide efficient realtime database synchronization.

IoT-based digital notice boards significantly improve communication efficiency compared to traditional and earlier wireless systems. However, practical implementation challenges still exist in areas such as realtime synchronization stability, embedded rendering performance, memory management, and cloud communication reliability.

2.6 Comparative Analysis of Existing Systems

Different digital notice board technologies provide different levels of communication capability, operational efficiency, and scalability. Table 2.1 presents a comparative analysis of traditional, GSM-based, Bluetooth-based, and IoT-based digital notice board systems.

Table 2.1 Comparative Analysis of Existing Notice Board Systems

Feature	Traditional System	GSM-Based System	Bluetooth-Based System	IoT-Based System
Remote Notice Update	No	Yes	Limited	Yes
Internet Connectivity	No	No	No	Yes

Realtime Synchronization	No	Limited	Limited	Yes
Communication Range	Physical Access	Wide Area	Short Range	Global
Operational Cost	Medium	High	Low	Low
Centralized Management	No	No	No	Yes
Mobile Application Support	No	Limited	Limited	Yes
Automatic Notice Management	No	No	No	Yes
Scalability	Low	Medium	Low	High

The comparative analysis shows that IoT-based systems provide better realtime communication, remote accessibility, centralized management, and scalability compared to traditional, GSM-based, and Bluetooth-based systems.

2.7 Limitations of Existing Systems

The study of existing digital notice board systems identified several common limitations that reduce operational efficiency and scalability.

The major limitations include:

- Lack of realtime cloud synchronization
- Limited remote accessibility
- Communication delay in GSM systems
- Limited communication range in Bluetooth systems
- High operational dependency on manual processes
- Absence of centralized cloud management

- Limited display customization capability
- Lack of automatic notice expiry handling
- Reduced scalability for large institutional environments

These limitations highlight the requirement for an efficient IoT-enabled digital notice board system capable of providing realtime communication, centralized cloud synchronization, remote accessibility, and dynamic display management using modern embedded and cloud technologies.

2.8 Summary

This chapter presented a literature survey of traditional notice boards, GSM-based systems, Bluetooth-based systems, and IoT-enabled digital notice board technologies. The study showed that IoT-based systems provide significant advantages in terms of realtime synchronization, cloud communication, remote management, and scalability.

The analysis of existing systems also identified several practical limitations related to communication efficiency, centralized management, and realtime display synchronization. The proposed “IoT-Enabled Digital Notice Board” system addresses these limitations through integration of ESP32 microcontroller, Firebase cloud services, Flutter mobile application framework, and HUB75E RGB LED matrix display technology.

CHAPTER 3

SYSTEM ANALYSIS AND ARCHITECTURE

3.1 Introduction

System analysis and architecture design play an important role in the development of IoT-based embedded communication systems. Proper system planning improves communication efficiency, modularity, scalability, realtime synchronization capability, and maintainability of the complete system.

The proposed "IoT-Enabled Digital Notice Board" system integrates Flutter mobile application, Firebase cloud services, ESP32 embedded controller, and HUB75E RGB LED matrix display technology to establish a centralized realtime communication platform.

This chapter discusses the functional requirements, non-functional requirements, hardware requirements, software requirements, overall system architecture, Firebase communication architecture, data flow, and workflow of the proposed system. The architectural design defines the interaction between cloud services, embedded hardware, and mobile application components used in the system implementation.

3.2 Functional Requirements

Functional requirements define the major operational features and system functionalities implemented in the proposed digital notice board system.

The major functional requirements are as follows:

- User registration and login authentication

- Email verification for authorized access
- Realtime notice publishing
- Cloud-based notice storage
- Realtime notice synchronization
- Scrolling text display on LED matrix
- Display brightness adjustment
- Scroll speed control
- Border style customization
- Text color configuration
- Automatic notice expiry handling
- Clock mode operation
- Wi-Fi communication
- Firebase cloud synchronization
- Realtime display updates

Table 3.1 Functional Requirements of the Proposed System

S. No.	Functional Requirement	Description
1	User Authentication	Secure login and registration
2	Notice Publishing	Realtime notice creation
3	Cloud Synchronization	Firebase communication
4	Display Management	LED matrix control
5	Settings Configuration	Display customization
6	Notice Expiry Handling	Automatic notice removal
7	Clock Mode Operation	Date and time display
8	Wi-Fi Communication	Internet connectivity

3.3 Non-Functional Requirements

Non-functional requirements define system quality attributes such as reliability, scalability, usability, performance, and maintainability.

The major non-functional requirements are:

- Low communication delay
- Stable realtime synchronization
- Secure cloud communication
- Responsive mobile application interface
- Efficient memory utilization
- Low operational cost
- Continuous display operation
- Reliable Wi-Fi connectivity
- Modular software architecture
- Scalable system implementation

Table 3.2 Non-Functional Requirements

S. No.	Requirement	Description
1	Reliability	Stable system operation
2	Scalability	Future feature expansion
3	Performance	Fast realtime synchronization
4	Security	Secure authentication
5	Usability	User-friendly interface
6	Maintainability	Modular architecture

3.4 Hardware Requirements

The hardware components used in the proposed system provide embedded processing, wireless communication, and realtime display functionality.

The major hardware components are:

- ESP32 microcontroller
- HUB75E RGB LED matrix display
- Power supply unit
- USB programming interface
- Wi-Fi communication module (integrated in ESP32)

ESP32 is used as the primary embedded controller responsible for Firebase communication, Wi-Fi synchronization, display rendering, and realtime notice processing.

The HUB75E RGB LED matrix display is used for displaying notices, scrolling text, and clock information in realtime.

Table 3.3 Hardware Components

S. No.	Hardware Component	Purpose
1	ESP32 Microcontroller	Embedded control and Wi-Fi communication
2	HUB75E RGB LED Matrix	Notice display
3	Power Supply Unit	System power source
4	USB Interface	Firmware programming
5	Wi-Fi Connectivity	Internet communication

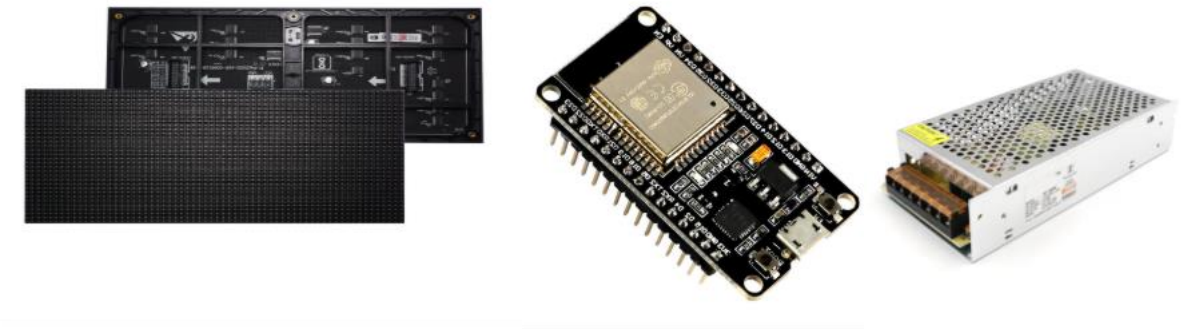


Figure 3.1 Main Hardware Components

3.5 Software Requirements

The software components used in the proposed system provide mobile application development, cloud communication, firmware programming, and realtime database synchronization.

The major software tools and technologies are:

- Flutter Framework
- Dart Programming Language
- Firebase Authentication
- Firebase Cloud Firestore
- Firebase Realtime Database
- Arduino IDE
- ESP32 Firebase Client Library
- HUB75 DMA Display Library

Table 3.4 Software Requirements

S. No.	Software Tool / Technology	Purpose
1	Flutter	Mobile application development
2	Dart	Application programming
3	Firebase	Cloud communication
4	Arduino IDE	ESP32 firmware development
5	Firebase ESP Client	Firebase communication
6	HUB75 DMA Library	LED matrix rendering

3.6 Overall System Architecture

The proposed system follows a cloud-connected embedded communication architecture consisting of a mobile application layer, cloud communication layer, and embedded display layer.

The Flutter mobile application acts as the primary user interface through which authorized users can publish and manage notices remotely. Firebase cloud services establish realtime communication and centralized data synchronization between the application and the ESP32 controller.

Firebase Authentication manages secure user access control, while Cloud Firestore and Firebase Realtime Database handle structured cloud storage and low-latency synchronization respectively.

The ESP32 microcontroller continuously monitors the Firebase Realtime Database through Wi-Fi communication and updates the HUB75E RGB LED matrix display whenever new notice data or display settings are synchronized.

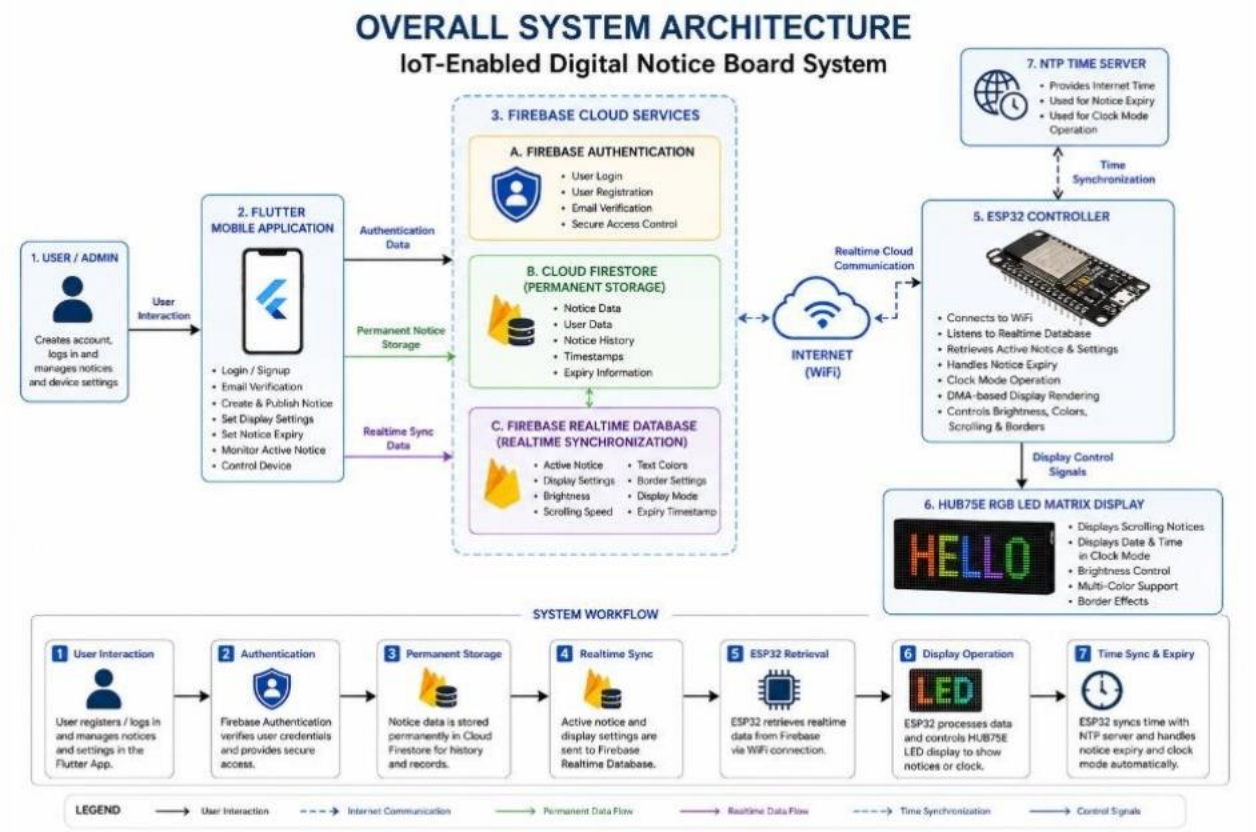


Figure 3.2 Overall System Architecture

3.7 Firebase Communication Architecture

A hybrid Firebase architecture was implemented using Firebase Authentication, Cloud Firestore, and Firebase Realtime Database. Each Firebase service performs a dedicated role within the system communication architecture.

Firebase Authentication manages secure login, registration, email verification, and password recovery operations. Cloud Firestore is used for permanent notice storage and historical data management. Firebase Realtime Database is used for low-latency synchronization between the Flutter application and ESP32 controller.

Whenever notice data or display settings are updated through the mobile application, the changes are instantly synchronized with ESP32 through realtime database communication.

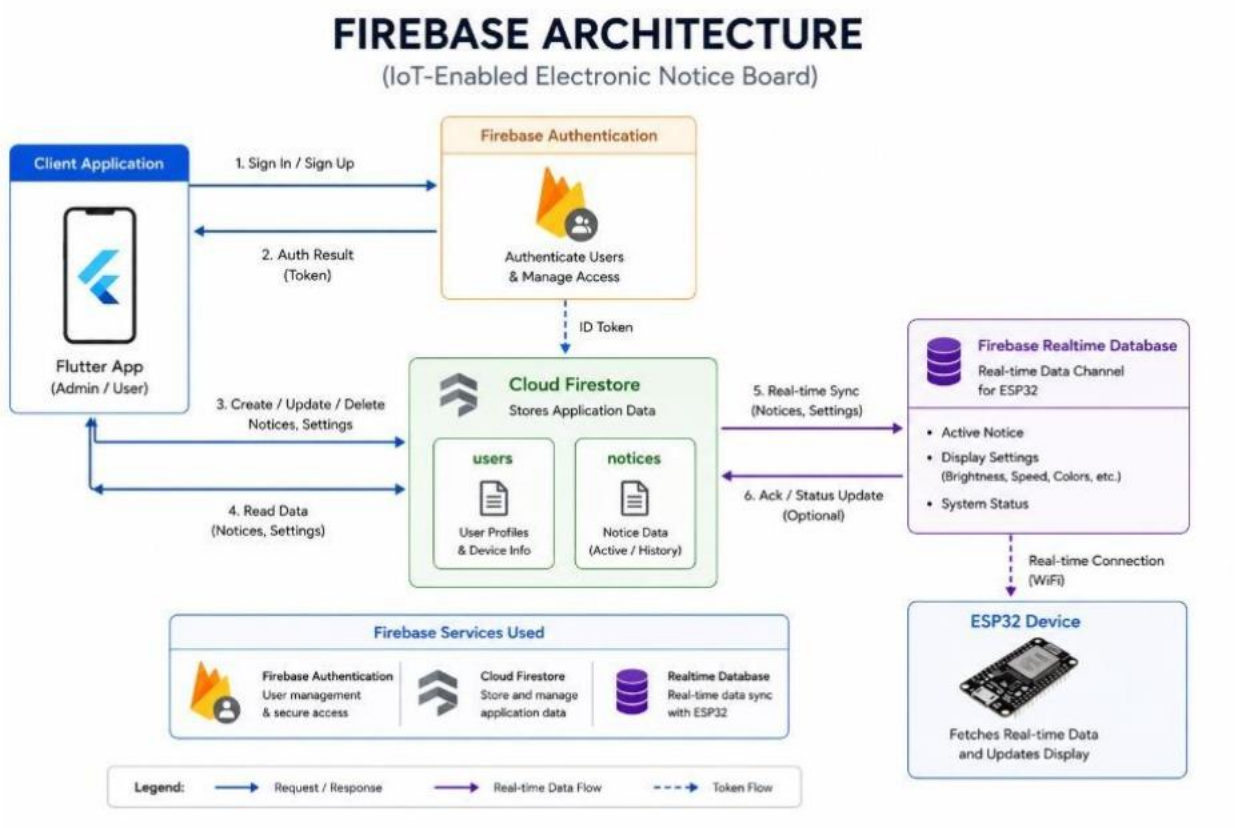


Figure 3.3 Firebase Communication Architecture

3.8 System Workflow

The operational workflow of the proposed system begins with user authentication through the Flutter mobile application. After successful login and email verification, authorized users can publish notices and modify display settings remotely.

The mobile application synchronizes updated notice data with Firebase cloud services. ESP32 continuously monitors the Firebase Realtime Database through Wi-Fi communication and fetches updated notice information whenever changes are detected.

The ESP32 processes the synchronized data and updates the HUB75E RGB LED matrix display accordingly. If no active notice is available, the system automatically switches to clock mode using NTP for synchronized internet time.

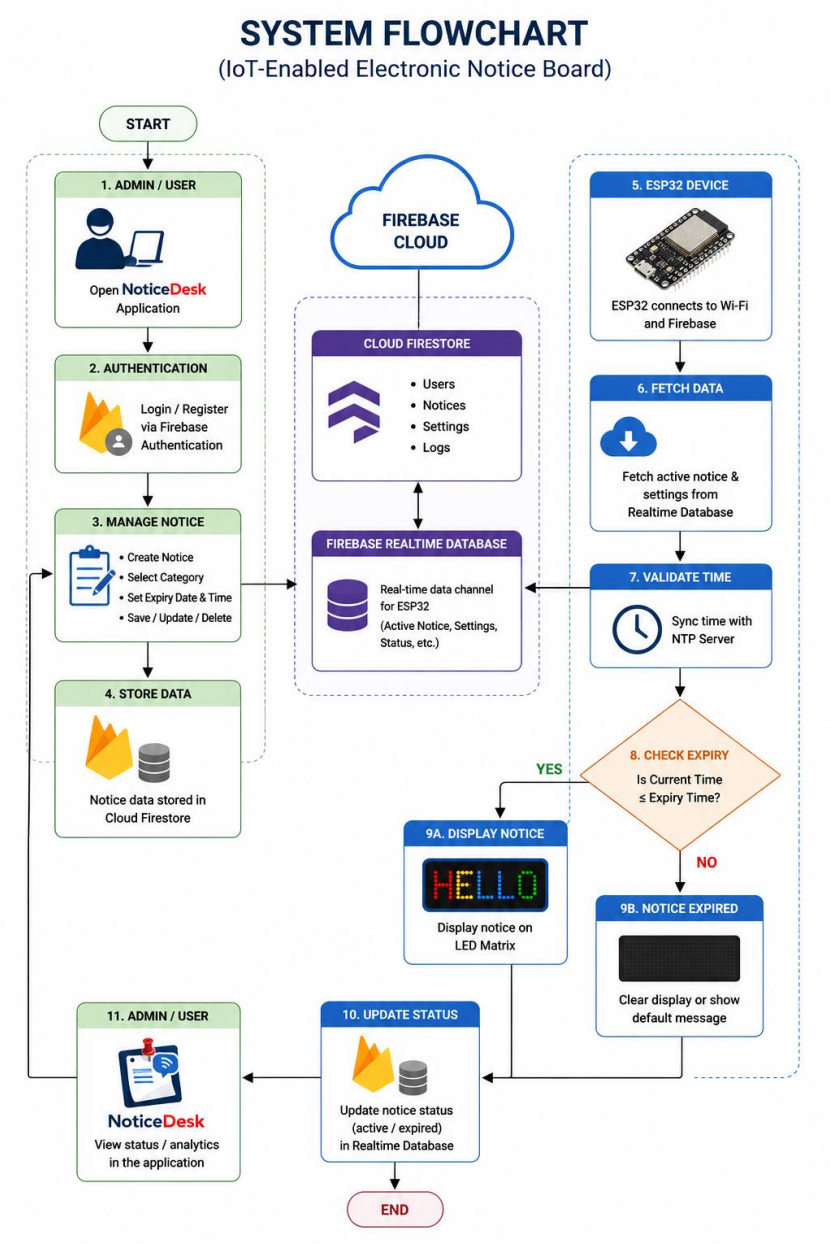


Figure 3.4 System flowchart of the Proposed System

3.9 Data Flow Diagram

The data flow diagram represents the movement of notice data between the mobile application, Firebase cloud services, and ESP32 display controller.

The user publishes notices through the mobile application interface. Firebase cloud services store and synchronize the notice data, while ESP32 fetches the updated information through realtime database communication and updates the LED matrix to display accordingly.

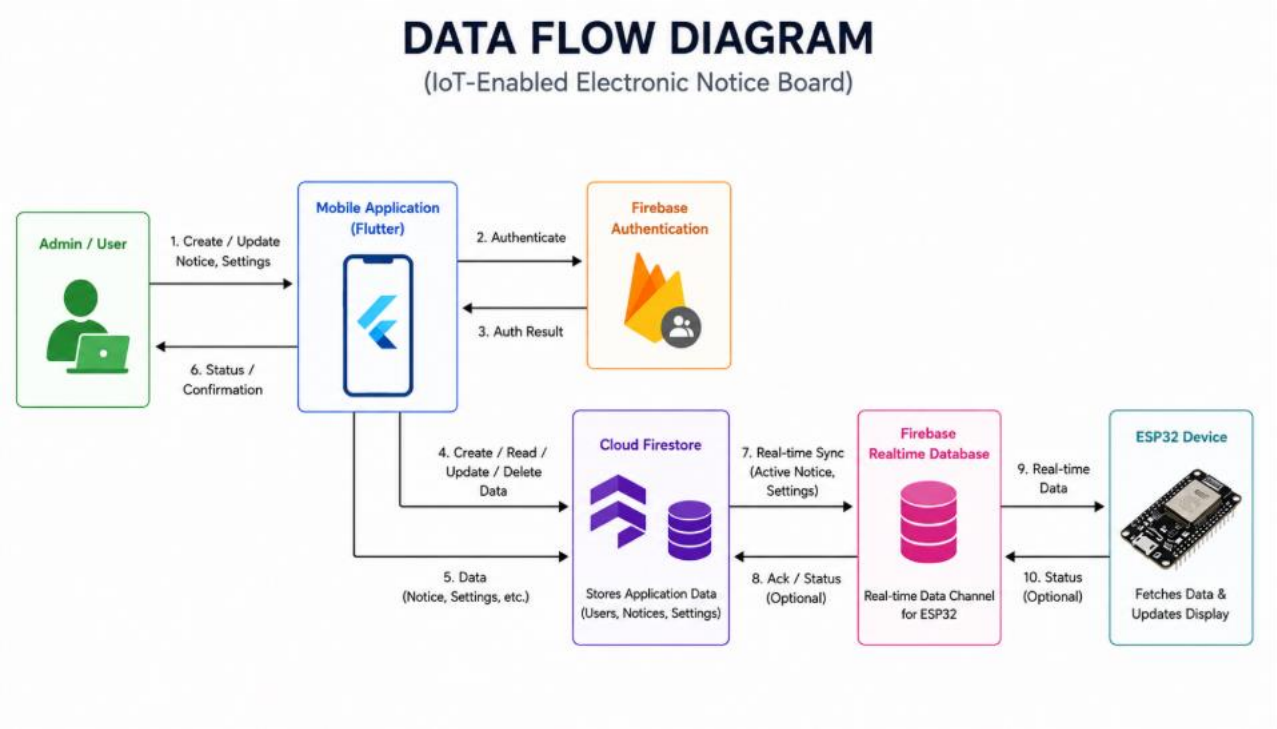


Figure 3.5 Data Flow Diagram of the Proposed System

3.10 Summary

This chapter presented the system analysis and architecture of the proposed “IoT-Enabled Digital Notice Board” system. Functional requirements, non-functional requirements, hardware requirements, software requirements, Firebase communication architecture, data flow, and operational workflow were discussed.

The architectural design establishes efficient integration between Flutter mobile application, Firebase cloud services, ESP32 embedded controller, and HUB75E RGB LED matrix display for realtime digital notice management and cloud-based synchronization.

CHAPTER 4

HARDWARE DESIGN AND IMPLEMENTATION

4.1 Introduction

The hardware implementation of the proposed "IoT-Enabled Digital Notice Board" system focuses on realtime cloud communication, embedded processing, and LED matrix display control using low-cost IoT hardware components. The hardware architecture was designed to provide stable wireless communication, efficient display rendering, realtime synchronization, and continuous display operation.

The major hardware components used in the proposed system are the ESP32 microcontroller, HUB75E RGB LED matrix display, power supply unit, and communication interfaces. ESP32 performs cloud communication, Firebase synchronization, realtime data processing, NTP synchronization, and LED matrix rendering operations, while the HUB75E RGB LED matrix panel acts as the primary visual output device.

This chapter discusses the hardware components, ESP32 microcontroller architecture, HUB75E RGB LED matrix display, ESP32 and HUB75E interfacing, DMA-based rendering implementation, Wi-Fi communication, NTP synchronization, driver IC analysis, and practical hardware challenges encountered during system implementation.

4.2 ESP32 Microcontroller

ESP32 is a low-cost and high-performance microcontroller widely used in IoT and embedded communication applications. It provides integrated WiFi and Bluetooth connectivity along with dual-core processing capability, making it suitable for realtime cloud communication and LED matrix display control.

The ESP32 microcontroller performs the following major operations within the proposed system:

- Firebase cloud communication
- Wi-Fi synchronization
- HUB75E display rendering
- NTP time synchronization
- Realtime notice processing
- Display settings synchronization
- Notice expiry handling
- Command processing

The integrated wireless communication capability of ESP32 eliminates the requirement for external communication modules and simplifies system implementation.

Table 4.1 ESP32 Technical Specifications

S. No.	Parameter	Specification
1	Processor	Dual-Core Xtensa LX6
2	Clock Frequency	Up to 240 MHz
3	Wireless Connectivity	WiFi and Bluetooth
4	SRAM	520 KB
5	Flash Memory	4 MB
6	Operating Voltage	3.3V
7	GPIO Support	Multiple GPIO Pins
8	Communication Interfaces	UART, SPI, I2C, PWM

Figure 4.1 ESP32 Development Board

4.3 HUB75E RGB LED Matrix Display

The HUB75E RGB LED matrix display is used as the primary visual output device for displaying realtime notices and clock information. The display supports RGB color rendering and high-speed parallel communication using HUB75E interface architecture.

The LED matrix display is capable of displaying scrolling text, dynamic color output, realtime notice updates, and display customization features. ESP32 directly controls the HUB75E display using GPIO-based parallel communication and DMA-based rendering techniques.

The proposed system supports scrolling notice display, dynamic brightness control, border customization, text color management, and realtime synchronization using the HUB75E display panel.

Table 4.2 HUB75E RGB LED Matrix Specifications

S. No.	Parameter	Specification
1	Display Type	RGB LED Matrix
2	Interface Type	HUB75E
3	Resolution	80 × 40 Pixels
4	Display Colors	RGB
5	Operating Voltage	5V
6	Communication Type	Parallel GPIO Interface
7	Display Mode	Static and Scrolling

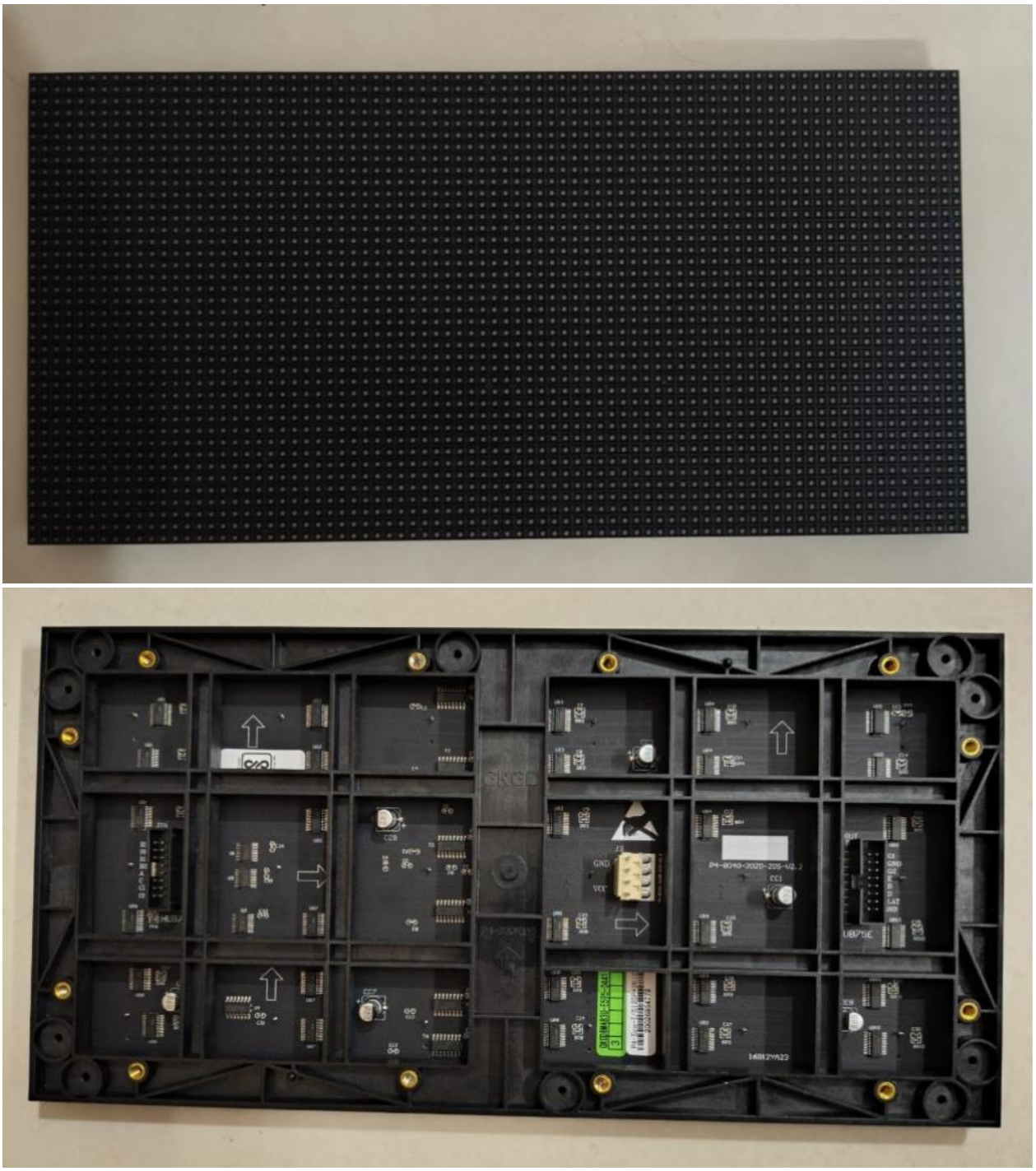


Figure 4.2 HUB75E RGB LED Matrix Display

4.4 ESP32 and HUB75E Interface Overview

The ESP32 microcontroller communicates with the HUB75E RGB LED matrix display using GPIO-based parallel communication architecture. Multiple GPIO pins are used for RGB data transmission, row addressing, clock synchronization, latch control, and output enable operations.

The communication architecture enables realtime display rendering and stable synchronization between ESP32 and the LED matrix display. DMA-based rendering is used to improve display refresh stability and reduce processor overhead during continuous display operation.

The ESP32 receives updated notice data from Firebase cloud services and processes the information before rendering the content on the HUB75E RGB LED matrix display.

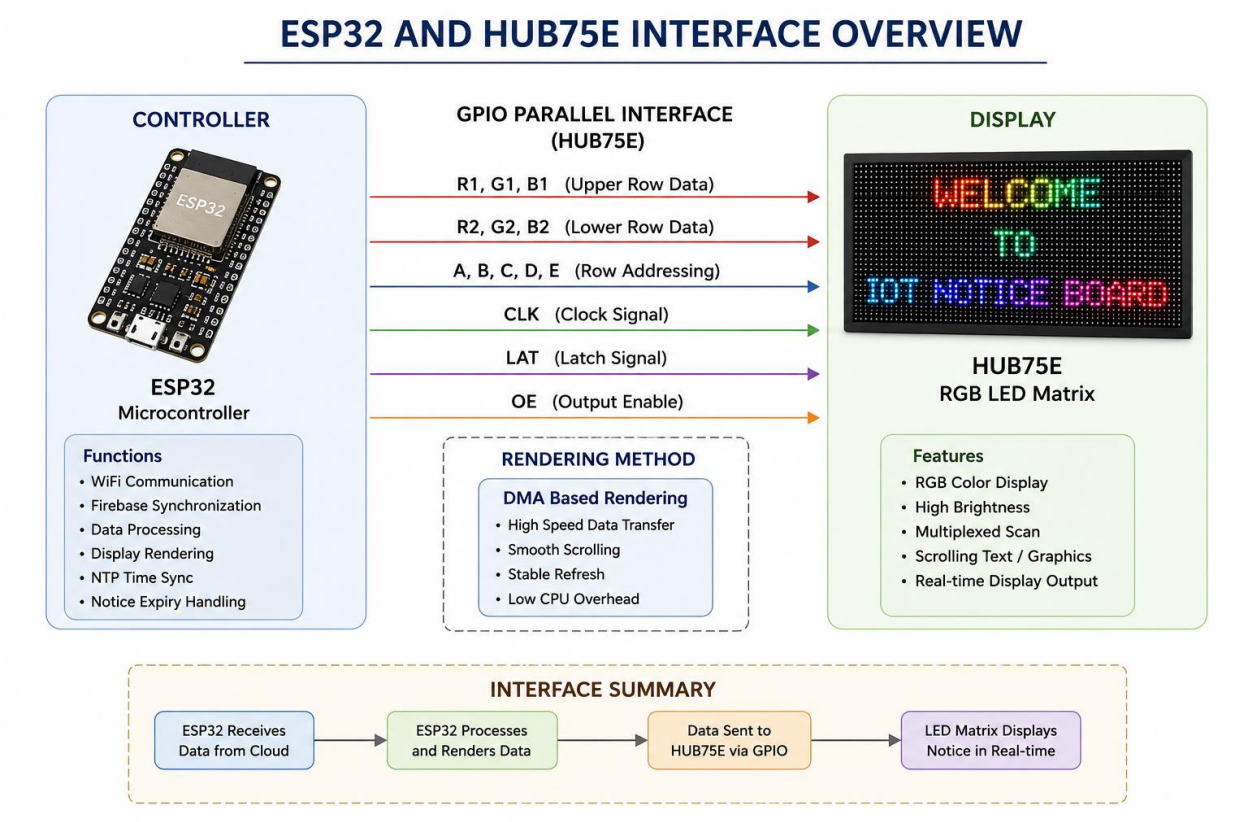


Figure 4.3 ESP32 and HUB75E Interface Overview

4.5 ESP32 and HUB75E Interface Connections

The HUB75E RGB LED matrix display requires multiple GPIO connections for RGB data transmission and display synchronization. Separate GPIO pins are used for upper-row and lower-row RGB data communication along with row address selection and control signals.

The ESP32 GPIO pins were configured according to HUB75E interface requirements to maintain stable display rendering and realtime synchronization.

Table 4.3 ESP32 and HUB75E Interface Connections

Signal Name	ESP32 GPIO Pin	Function
R1	GPIO 25	Upper Red Data
G1	GPIO 26	Upper Green Data
B1	GPIO 27	Upper Blue Data
R2	GPIO 14	Lower Red Data
G2	GPIO 12	Lower Green Data
B2	GPIO 13	Lower Blue Data
A	GPIO 23	Row Address Selection
B	GPIO 19	Row Address Selection
C	GPIO 5	Row Address Selection
D	GPIO 2	Row Address Selection
E	GPIO32 15	Extended Row Address
LAT	GPIO 4	Latch Signal
OE	GPIO 16	Output Enable
CLK	GPIO 18	Clock Signal

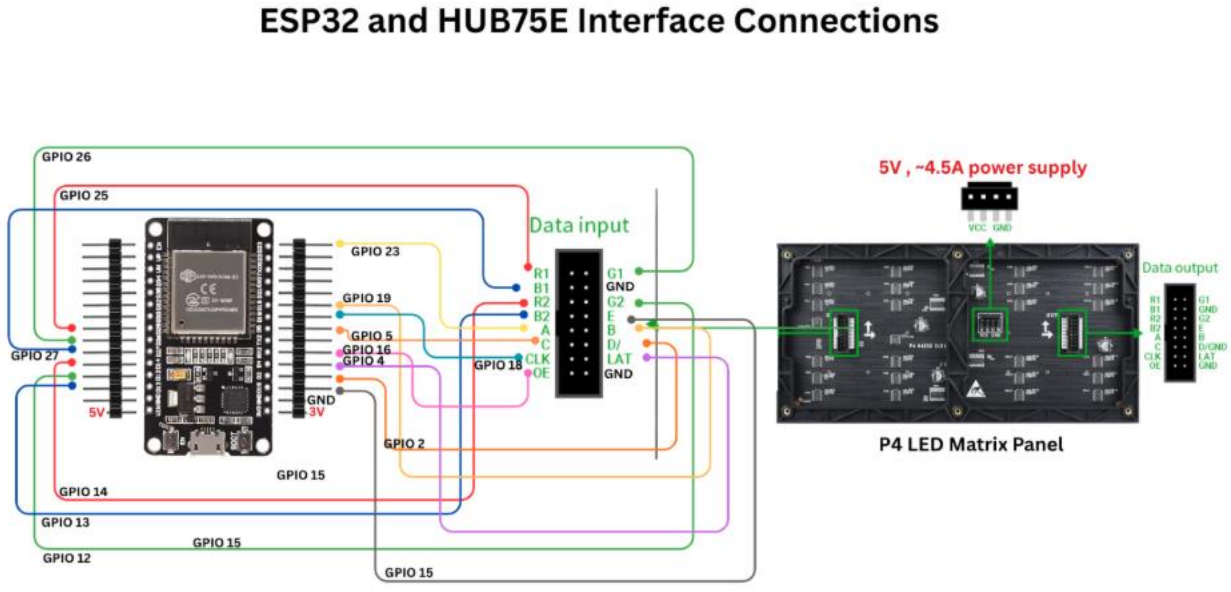


Figure 4.4 ESP32 and HUB75E Interface Connections

4.6 DMA-Based Rendering

Direct Memory Access (DMA)-based rendering was implemented to improve LED matrix refresh stability and reduce processor overhead during continuous display operation.

The HUB75E RGB LED matrix requires high-speed refresh operations for stable display rendering. Traditional software-based refresh methods may produce display flickering and unstable rendering during simultaneous cloud communication and display synchronization.

The ESP32-HUB75-MatrixPanel-I2S-DMA library was used to implement DMA-based rendering architecture. DMA allows display data to be transferred directly from memory to the LED matrix interface without continuous CPU intervention.

DMA-based rendering improved:

- Display refresh stability
- Scrolling smoothness
- Realtime rendering performance
- Multitasking capability

4.7 Wi-Fi Communication and NTP Synchronization

The ESP32 uses integrated Wi-Fi capability to establish communication with Firebase cloud services through internet connectivity. Wi-Fi communication is responsible for Firebase synchronization, realtime notice fetching, settings of synchronization, and cloud-based command handling.

Network Time Protocol (NTP) synchronization was implemented to obtain accurate internet date and time information. NTP synchronization is used for notice expiry handling, timestamp validation, and clock mode operation.

When no active notice is available, the system automatically switches to realtime clock display mode using NTP synchronized date and time information.

4.8 Driver IC Analysis

During implementation and testing, different HUB75E RGB LED matrix driver IC configurations were analyzed to evaluate display compatibility and rendering stability.

The DP3364S and DP5125F driver ICs were examined for display refresh performance, rendering stability, scrolling behavior, and compatibility with DMA-based rendering architecture.

The analysis showed variations in rendering behavior and synchronization characteristics between different driver IC configurations. Stable realtime display operation required proper GPIO synchronization and optimized DMA rendering support.

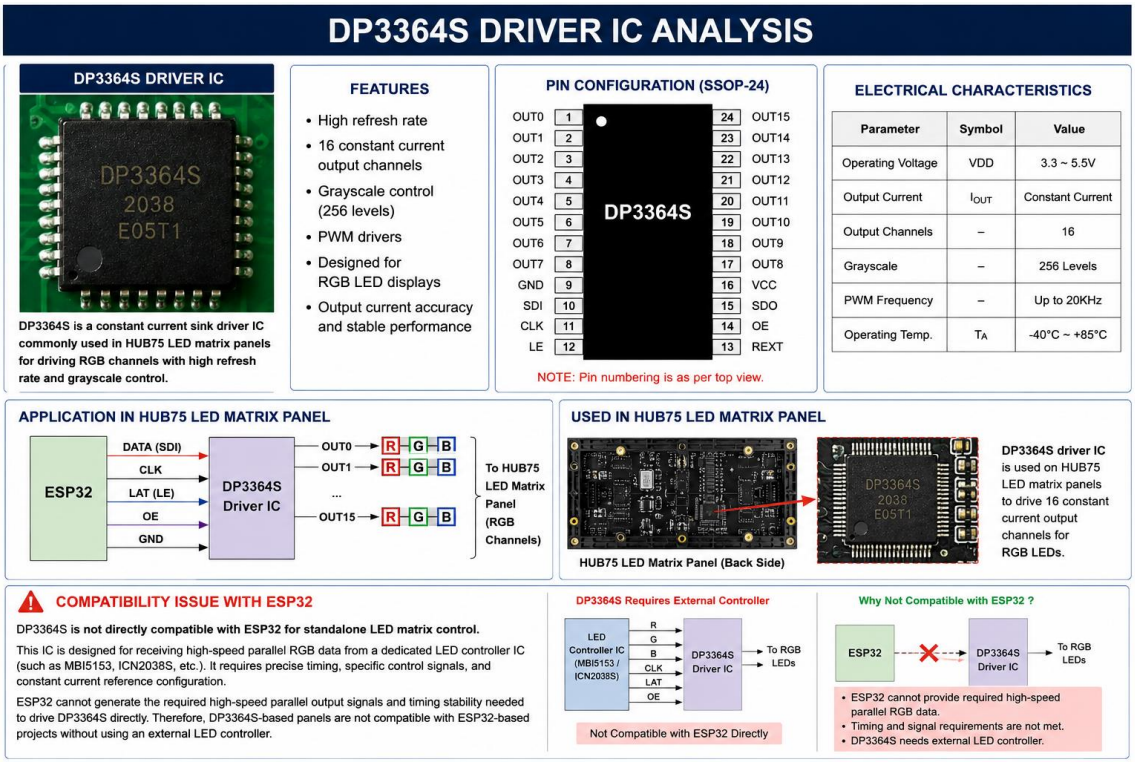


Figure 4.5 DP3364S Driver IC Analysis

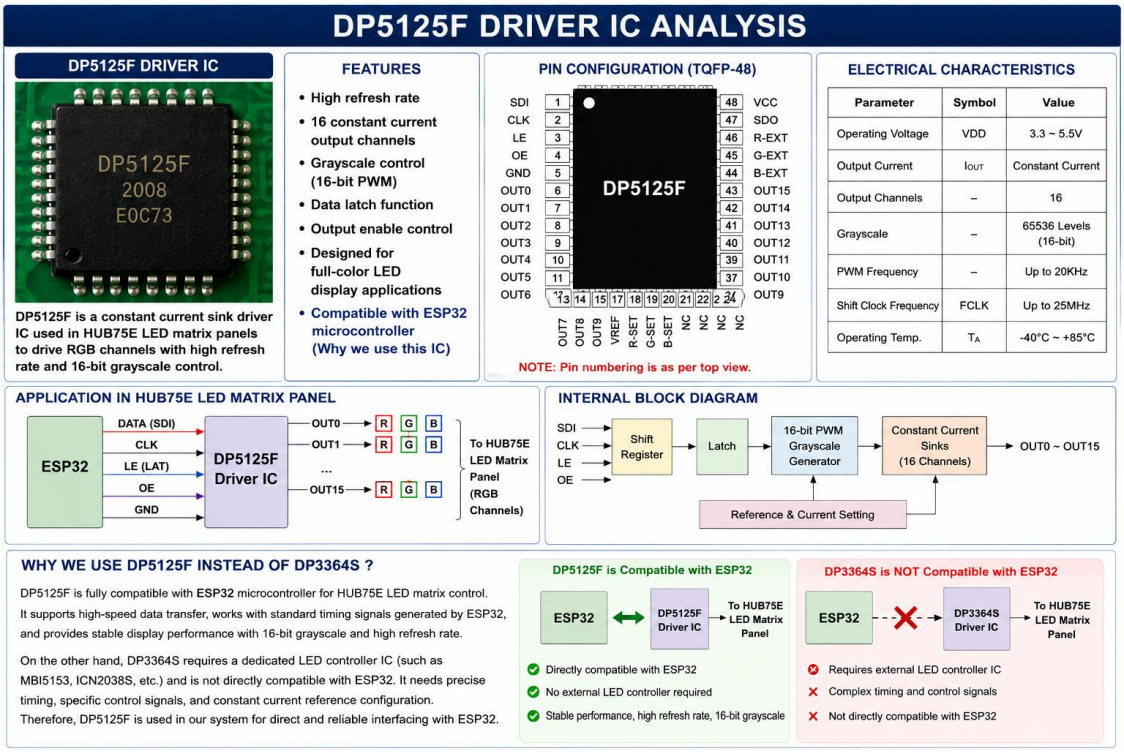


Figure 4.6 DP5125F Driver IC Analysis

4.9 Hardware Challenges and Implementation Issues

Several practical hardware challenges were encountered during development and implementation of the proposed system.

The major challenges included:

- Stable HUB75E display rendering
- GPIO synchronization
- Realtime cloud communication stability
- Memory management during DMA rendering
- SSL communication reliability
- Continuous display refresh stability

The ESP32 simultaneously handled WiFi communication, Firebase SSL communication, JSON processing, DMA rendering, and realtime display refresh operations. Proper GPIO mapping, optimized rendering configuration, and controlled Firebase synchronization intervals were required to maintain stable system operation.

Practical debugging and testing significantly improved realtime communication reliability and display rendering performance.

4.10 Summary

This chapter discussed the hardware design and implementation of the proposed “IoT-Enabled Digital Notice Board” system. ESP32 microcontroller architecture, HUB75E RGB LED matrix display, ESP32 and HUB75E interfacing, DMA-based rendering, Wi-Fi communication, NTP synchronization, and driver IC analysis were presented.

The hardware implementation successfully achieved stable realtime cloud communication, efficient LED matrix rendering, and reliable wireless notice synchronization suitable for practical digital notice board applications.

CHAPTER 5

SOFTWARE DESIGN AND IMPLEMENTATION

5.1 Introduction

The software implementation of the proposed "IoT-Enabled Digital Notice Board" system focuses on realtime cloud communication, secure user authentication, mobile application development, and embedded display synchronization. The software architecture was designed to provide modularity, scalability, efficient cloud communication, and responsive user interaction.

The complete software system consists of Flutter mobile application, Firebase cloud services, ESP32 firmware, and realtime synchronization logic. The Flutter framework was used for mobile application development, while Firebase services were integrated for authentication, cloud storage, and realtime communication.

The ESP32 firmware continuously synchronizes notice data and display settings from Firebase Realtime Database and updates the HUB75E RGB LED matrix display in realtime. Additional software modules were implemented for display configuration management, notice expiry handling, NTP synchronization, and command processing.

This chapter discusses mobile application development, Firebase cloud integration, authentication system, realtime database communication, state management, repository architecture, display configuration logic, notice expiry mechanism, and ESP32 cloud communication implementation.

5.2 Mobile Application Development

The mobile application of the proposed "IoT-Enabled Digital Notice Board" system was developed using the Flutter framework and Dart programming language. Flutter was selected due to its cross-platform capability, responsive UI rendering, fast development cycle, and efficient Firebase integration support.

The mobile application acts as the primary user interface of the system and enables authorized users to remotely create, manage, and publish notices through internet connectivity. The application also provides realtime monitoring of active notices and dynamic display configuration management.

The application supports multiple operational modules including:

- User authentication
- Notice publishing
- Realtime notice monitoring
- Display settings management
- Color customization
- Scroll speed control
- Brightness management
- Notice expiry configuration

Responsive UI layouts were implemented to maintain stable application behavior across different Android device resolutions and screen dimensions.

(Refer Appendix B for mobile Application Interface Screenshots)

5.3 Firebase Cloud Integration

Firebase cloud services were integrated to provide realtime communication, cloud storage, secure authentication, and centralized notice management.

A hybrid Firebase architecture was implemented using Firebase Authentication, Cloud Firestore, and Firebase Realtime Database. Each Firebase service performs a dedicated operational role within the system architecture.

Firebase Authentication manages secure login, registration, email verification, and password recovery functionality. Cloud Firestore is used for permanent notice storage and historical notice management, while Firebase Realtime Database provides low-latency synchronization between the Flutter application and ESP32 controller.

Whenever notices or display settings are updated through the mobile application, the changes are instantly synchronized with the ESP32 embedded controller through realtime database communication.

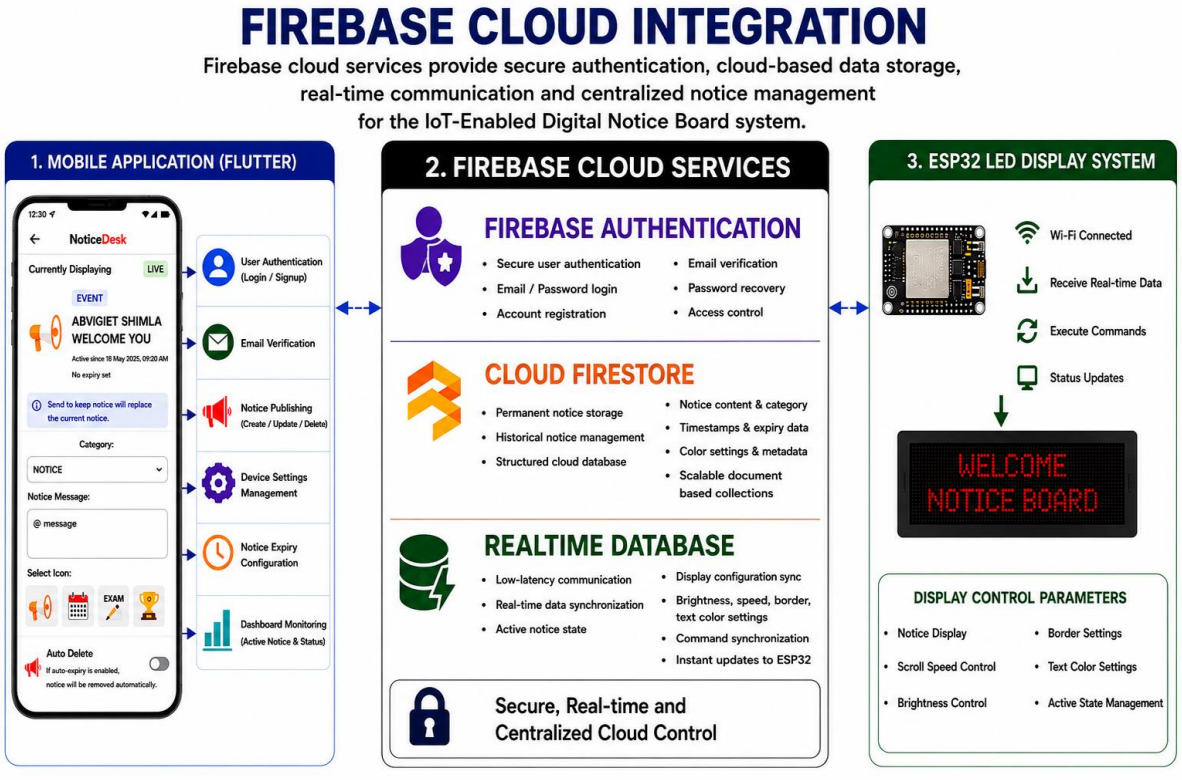


Figure 5.1 Firebase Cloud Integration Architecture

5.4 Authentication and User Access System

Secure user authentication was implemented using Firebase Authentication services. The authentication system manages user registration, secure login, email verification, password recovery, and access control operations.

Only verified users are allowed to access the notice management system and publish notices. Email verification functionality improves system security by preventing unauthorized access and invalid account usage.

The authentication workflow consists of:

- User registration
- Email verification
- Secure login
- Password reset functionality
- Session validation

Realtime authentication and state monitoring were also implemented within the Flutter application to maintain secure access control during application operation.

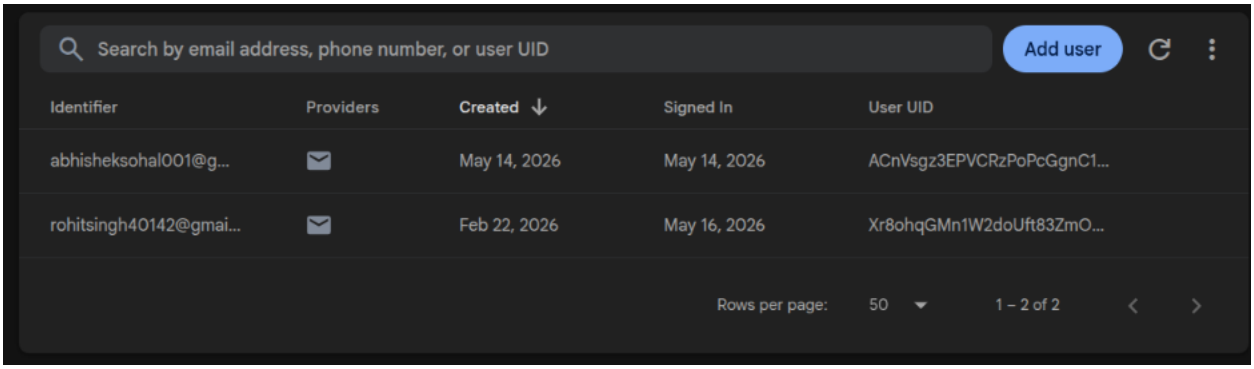


Figure 5.2 Firebase Authentication Panel for User Account Management

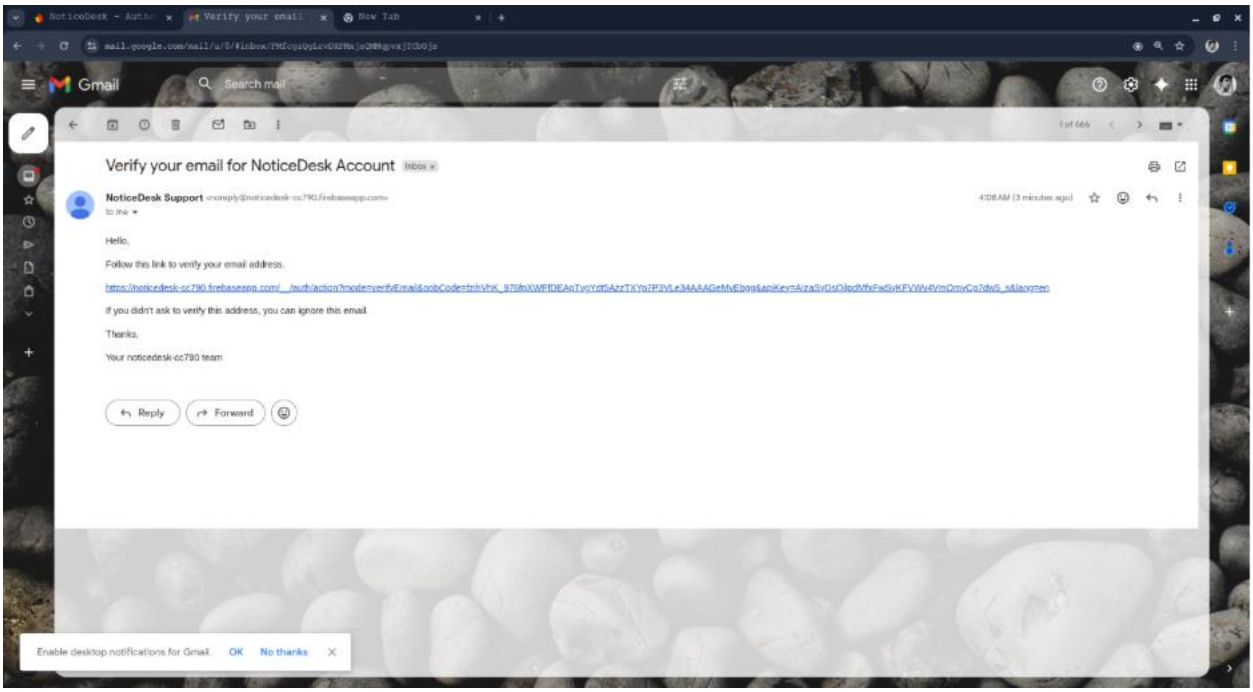


Figure 5.3 Authentication and Email Verification System

5.5 Realtime Database Communication

Firebase Realtime Database was implemented for low-latency communication between the Flutter mobile application and ESP32 embedded controller.

The ESP32 continuously monitors Firebase Realtime Database through WiFi communication and fetches updated notice data and display settings whenever changes occur.

Realtime Database stores:

- Active notice information
- Display settings
- Scroll speed configuration
- Brightness settings
- Border customization

- Text color configuration
- Command synchronization data

Realtime synchronization significantly reduced communication delay and enabled immediate display updates without requiring dedicated backend servers.

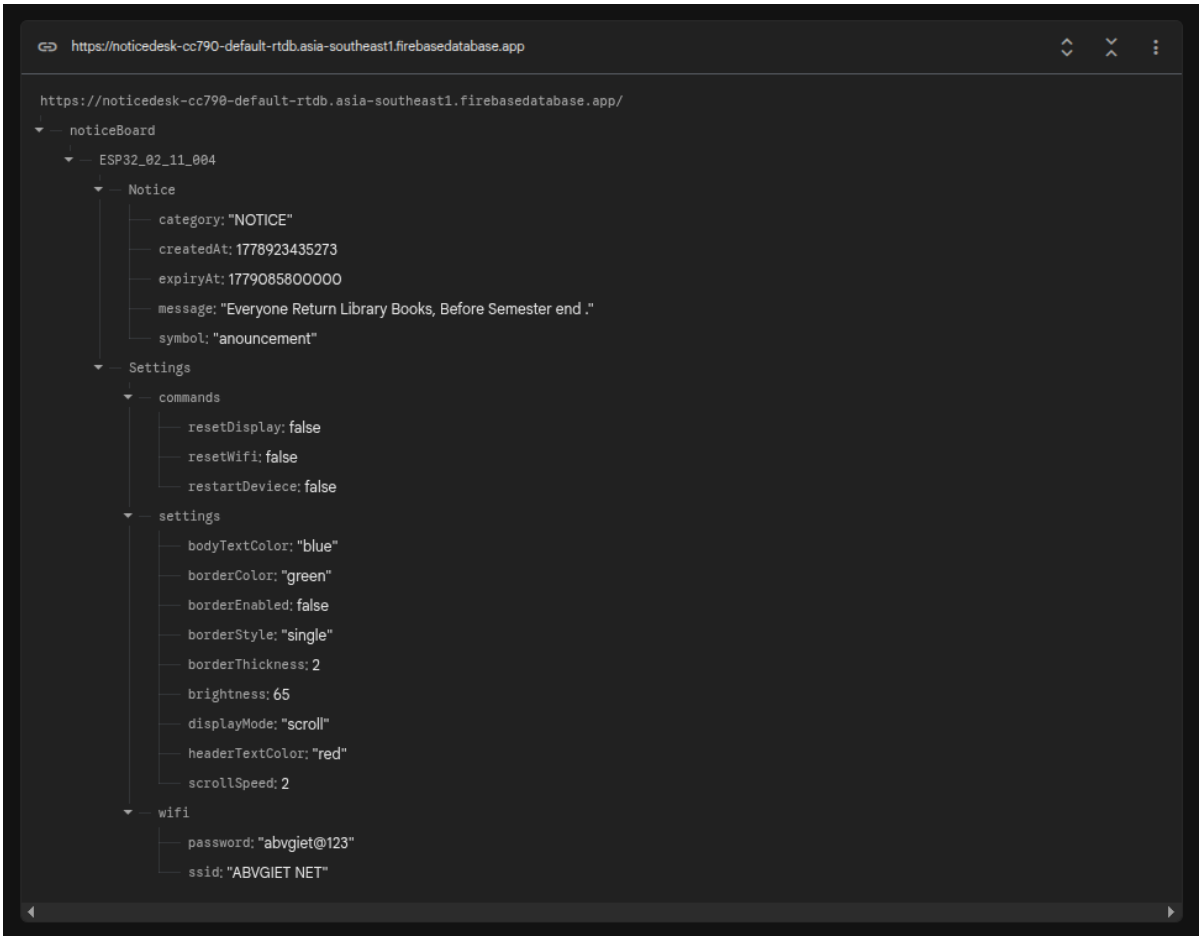


Figure 5.4 Firebase Realtime Database Structure

5.6 Bloc/Cubit State Management

Bloc and Cubit state management architecture was implemented within the Flutter application to improve modularity, maintainability, and responsive UI synchronization.

Cubit-based state management simplified handling of authentication states, notice synchronization states, loading operations, and realtime UI updates.

Separate Cubits were implemented for:

- Authentication management
- Notice management
- Settings management
- Realtime synchronization handling

Bloc/Cubit architecture improved code organization and reduced unnecessary UI rebuild operations during realtime cloud synchronization.

5.7 Repository Architecture and Dependency Injection

Repository architecture and dependency injection were implemented to improve software modularity, code organization, and maintainability.

Repository classes were used to separate Firebase communication logic and business logic from UI components. Separate repositories were implemented for authentication, notice management, realtime synchronization, and display configuration handling.

Dependency injection was implemented using GetIt service locator package. GetIt simplified centralized access to repositories, Firebase services, and shared application resources while reducing tight coupling between application modules.

The repository-based architecture improved scalability, debugging capability, and structure software development.

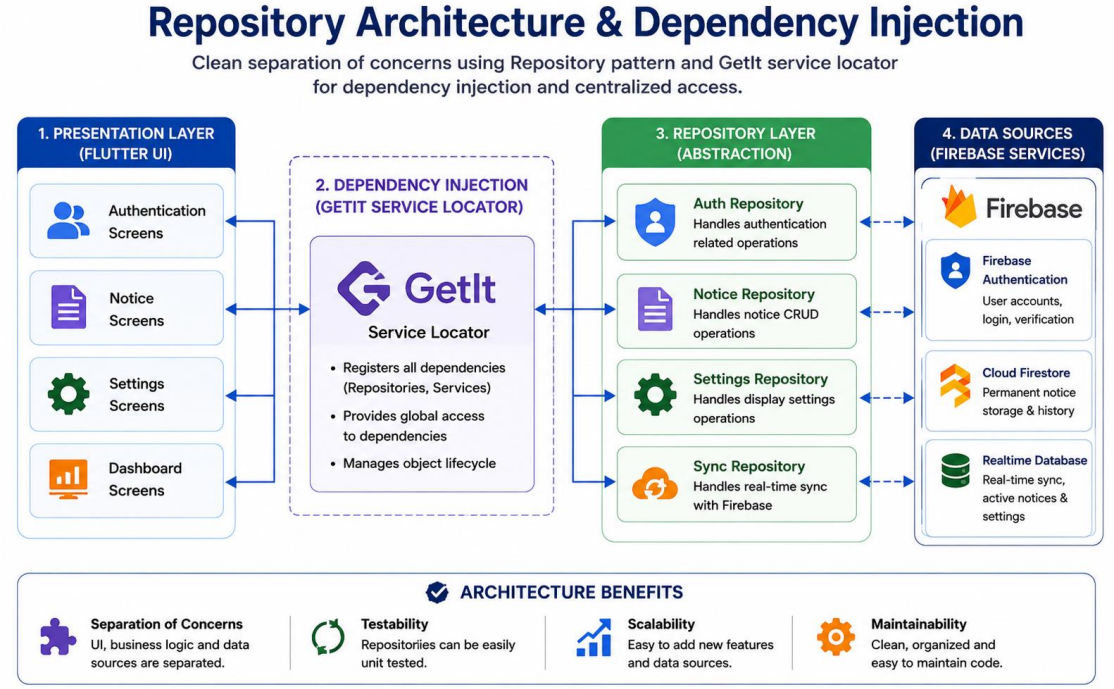


figure 5.5 Repository Architecture and Dependency Injection

5.8 Device Settings and Display Configuration

The system supports dynamic display customization and realtime configuration management through the Flutter mobile application.

Users can remotely configure multiple display settings including:

- Brightness level
- Scroll speed
- Display mode
- Border style
- Border color
- Header text color
- Body text color

Updated settings are synchronized through Firebase Realtime Database and automatically applied by the ESP32 controller without requiring system restart.

Realtime configuration management significantly improved the usability and operational flexibility of the display system.

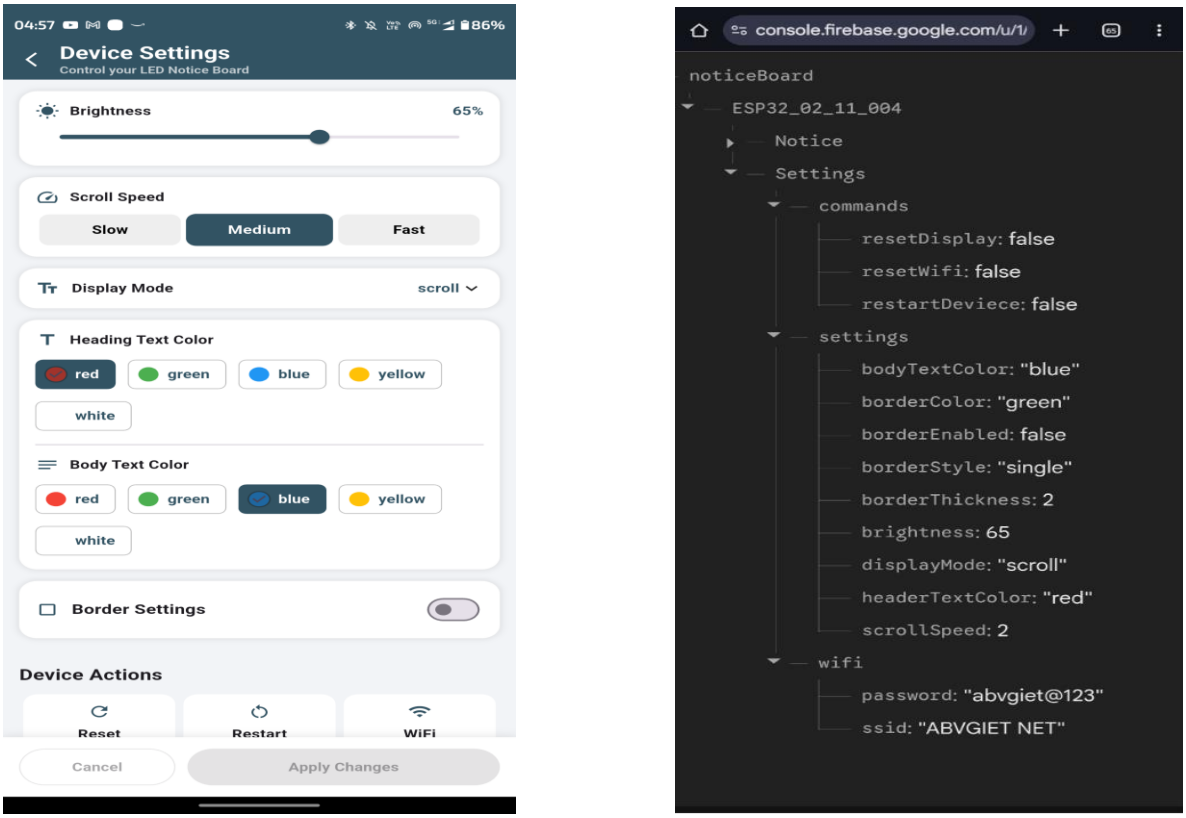


Figure 5.6 Display Configuration Management Interface

5.9 Notice Expiry and Clock Mode Logic

Automatic notice expiry handling was implemented using epoch timestamp comparison and NTP time synchronization.

Whenever a notice is published with expiry configuration enabled, the expiry timestamp is stored in Firebase Realtime Database. ESP32 continuously compares the current NTP synchronized time with the stored expiry timestamp.

When the expiry condition is reached:

- The notice is automatically removed
- Firebase notice data is cleared
- The system switches to clock mode operation

If no active notice is available, the HUB75E RGB LED matrix automatically displays realtime date and time information obtained through NTP synchronization.

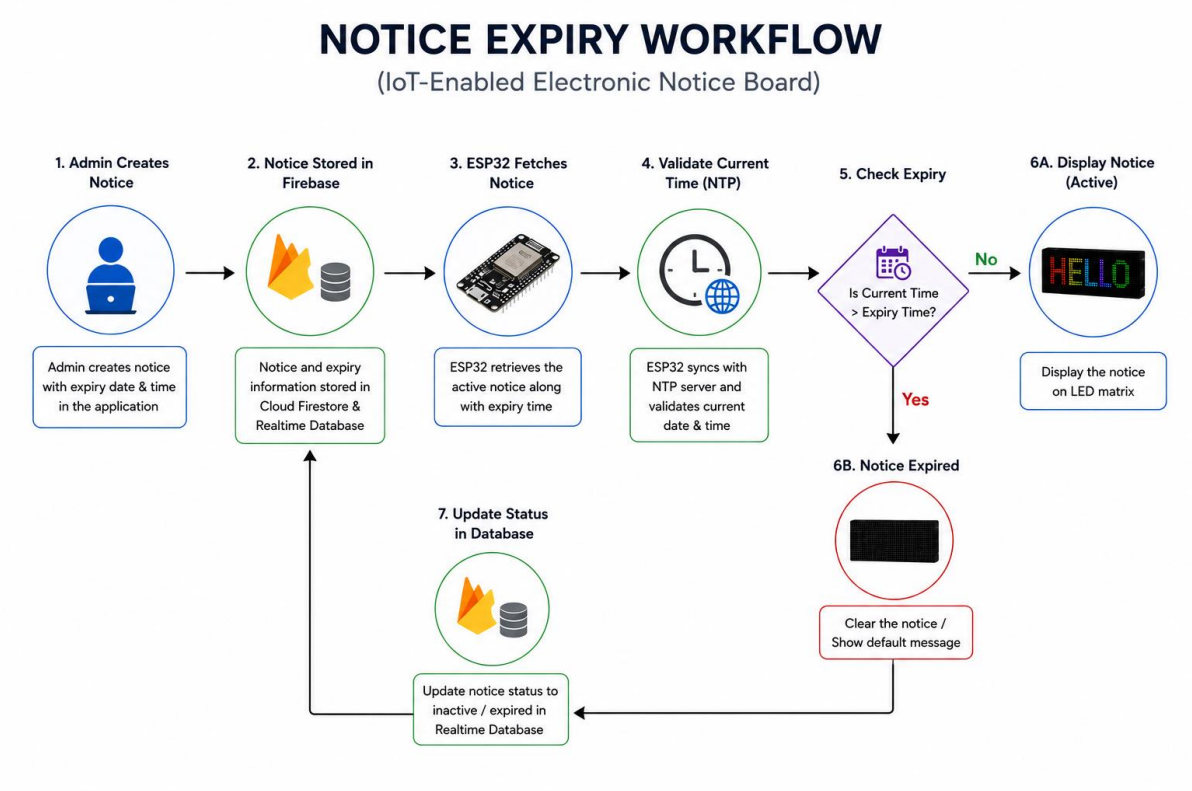


Figure 5.7 Notice Expiry and Clock Mode Workflow

5.10 ESP32 and Firebase Communication Logic

The ESP32 firmware continuously establishes communication with Firebase Realtime Database using WiFi connectivity and Firebase ESP Client Library.

The firmware performs the following operations continuously:

- WiFi connection management
- Firebase synchronization
- Notice fetching
- Display settings synchronization
- Command handling

- NTP synchronization
- Notice expiry validation
- LED matrix rendering

The ESP32 fetches updated cloud data at fixed synchronization intervals and updates the HUB75E RGB LED matrix display in realtime.

DMA-based rendering and optimized synchronization intervals improved stable display refresh performance during simultaneous cloud communication operations.

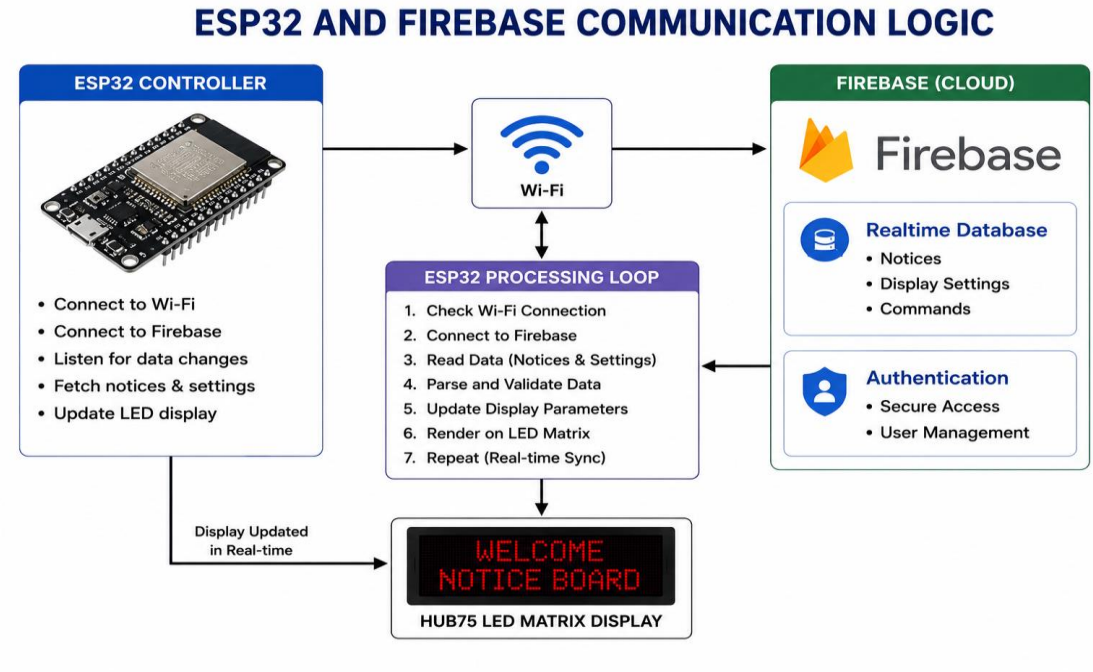


figure 5.8 ESP32 and Firebase Communication Logic

5.11 Software Challenges and Implementation Issues

Several practical software challenges were encountered during development and implementation of the proposed system.

The major challenges included:

- Stable Firebase SSL communication
- Realtime synchronization reliability
- Memory management during JSON processing
- Responsive UI handling
- Continuous realtime database communication
- Stable DMA rendering during cloud synchronization

Simultaneous handling of Firebase communication, realtime display rendering, and WiFi synchronization required optimized cloud polling intervals and efficient memory management strategies.

Practical debugging and testing significantly improved software stability, cloud synchronization performance, and realtime display responsiveness.

5.12 Summary

This chapter discussed the software design and implementation of the proposed "IoT-Enabled Digital Notice Board" system. Flutter mobile application development, Firebase cloud integration, authentication system, realtime database communication, Bloc/Cubit architecture, repository implementation, display configuration management, notice expiry handling, and ESP32 cloud communication logic were presented.

The software implementation successfully achieved secure cloud communication, realtime synchronization, modular application architecture, and responsive display management suitable for realtime digital notice board applications.

CHAPTER 6

TESTING AND RESULTS

6.1 Introduction

Testing and validation are essential processes in embedded IoT systems to ensure stable communication, reliable cloud synchronization, accurate display operation, and secure application functionality. The proposed "IoT-Enabled Digital Notice Board" system was tested under different operational conditions to evaluate realtime communication performance, display synchronization, authentication functionality, and overall system reliability.

The testing process included validation of Firebase cloud communication, user authentication, realtime synchronization, notice publishing, display configuration management, notice expiry handling, and LED matrix display operation.

Both hardware and software modules were tested individually and collectively to verify stable integration between Flutter mobile application, Firebase cloud services, ESP32 embedded controller, and HUB75E RGB LED matrix display system.

This chapter presents different testing procedures, experimental observations, console outputs, realtime synchronization results, and practical performance analysis of the proposed system.

6.2 Authentication Testing

Authentication testing was performed to verify secure user registration, login validation, email verification, password recovery, and access control functionality implemented using Firebase Authentication services.

The testing process included creation of new user accounts, verification of email delivery, login validation using verified accounts, invalid login testing, and password reset functionality.

The authentication system successfully restricted unauthorized users and allowed only verified users to access the notice management application. Firebase Authentication services maintained stable cloud-based user validation throughout the testing process.

Table 6.1 Authentication Testing Results

S. No.	Test Case	Result Status
1	User Registration	Successful
2	Email Verification	Successful
3	Secure Login	Successful
4	Invalid Login Handling	Successful
5	Password Recovery	Successful
6	Access Control Validation	Successful

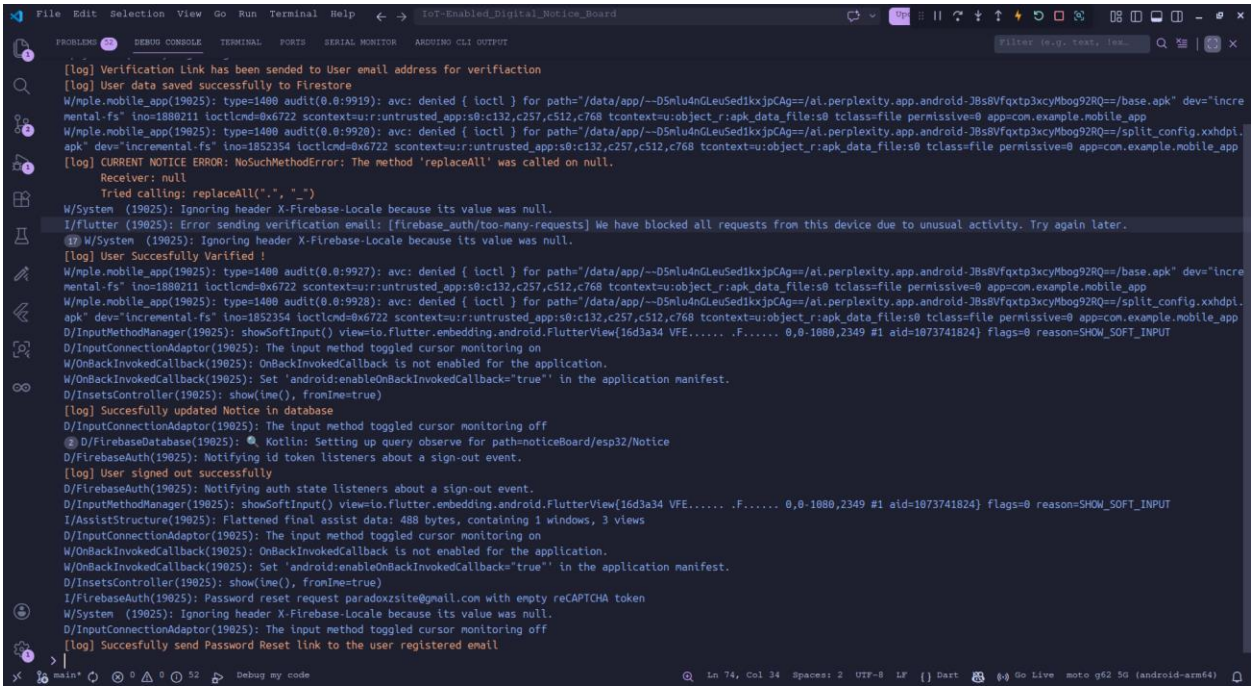


Figure 6.1 Authentication and User Access Testing Console Output

6.3 Realtime Synchronization Testing

Realtime synchronization testing was performed to verify communication stability between Flutter mobile application, Firebase Realtime Database, and ESP32 embedded controller.

The ESP32 continuously monitored Firebase Realtime Database and synchronized updated notices and display settings in realtime. Multiple notice publishing operations were performed to evaluate synchronization speed and display update reliability.

The testing confirmed stable realtime cloud communication and immediate synchronization between the mobile application and LED matrix display system.

Table 6.2 Realtime Synchronization Testing Results

S. No.	Test Case	Result Status
1	Notice Publishing	Successful
2	Realtime Display Update	Successful
3	Notice Replacement	Successful
4	Active Notice Synchronization	Successful
5	Display Settings Synchronization	Successful
6	Continuous Cloud Communication	Stable

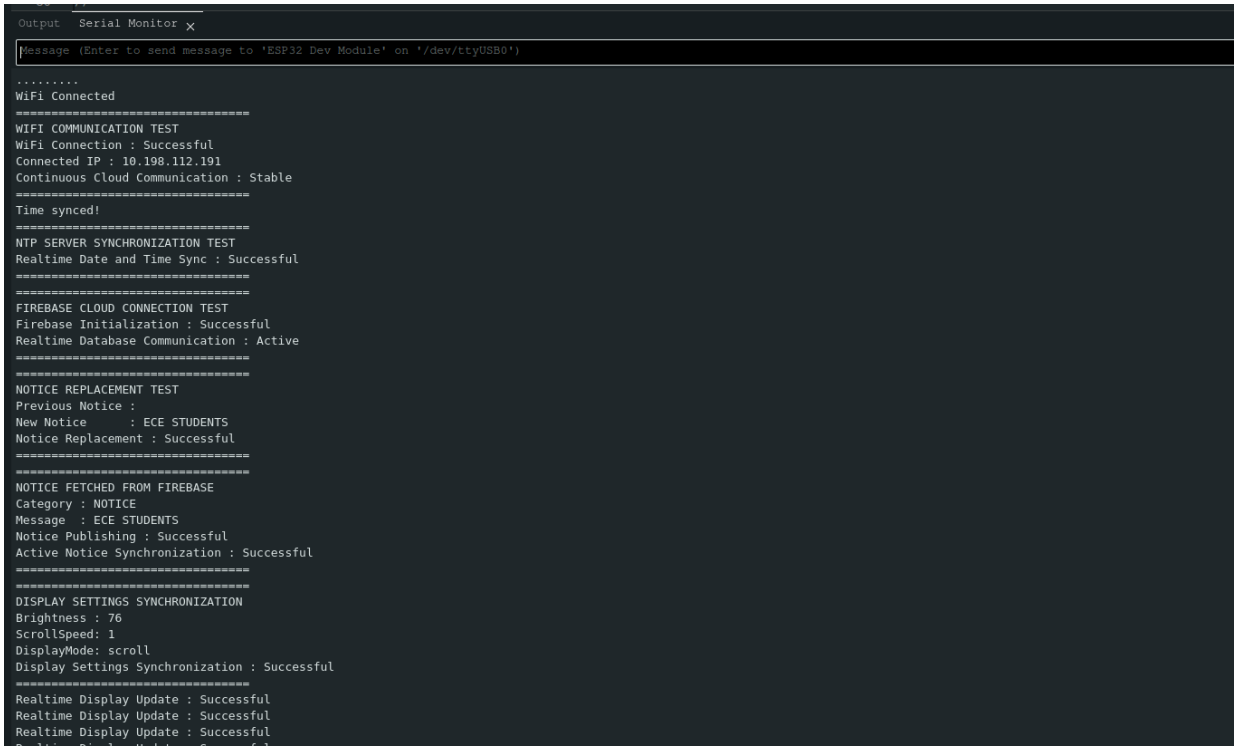


Figure 6.2 ESP32 Serial Monitor Realtime Logs

6.4 Display Configuration Testing

Display configuration testing was performed to verify realtime display customization functionality implemented within the mobile application and ESP32 display controller.

Different display parameters such as brightness level, scrolling speed, text color, border style, border color, and display mode were modified through the mobile application interface. ESP32 successfully synchronized updated settings from Firebase Realtime Database and applied the changes to the HUB75E RGB LED matrix display in realtime.

The display configuration system maintained stable synchronization without requiring device restart or manual refresh operations.

Table 6.3 Display Configuration Testing Results

S. No.	Configuration Parameter	Result Status
1	Brightness Control	Successful
2	Scroll Speed Adjustment	Successful
3	Border Customization	Successful
4	Text Color Configuration	Successful
5	Display Mode Switching	Successful
6	Realtime Settings Synchronization	Successful

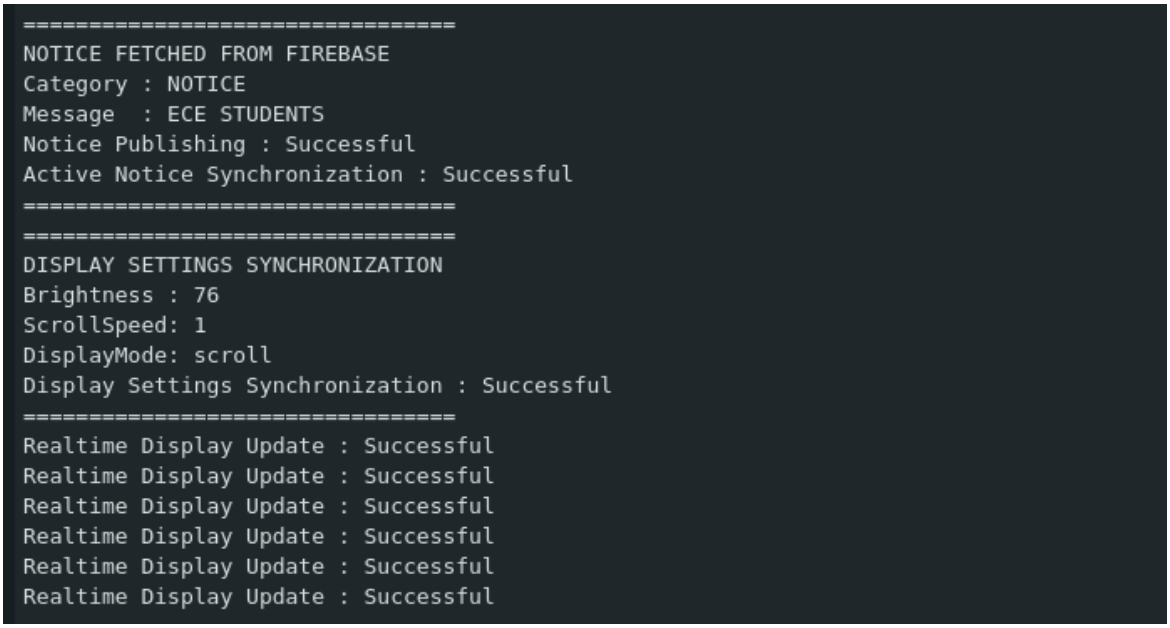


figure 6.3 Display Management Settings Testing

6.5 Notice Expiry and Clock Mode Testing

Notice expiry testing was performed to validate automatic notice removal functionality implemented using NTP time synchronization and epoch timestamp comparison.

Different notices were published with configured expiry durations. ESP32 continuously compared the current NTP synchronized time with the stored expiry timestamp. After reaching the expiry condition, the system automatically removed the active notice and switched to realtime clock display mode.

The testing confirmed accurate NTP synchronization, stable expiry handling, and reliable automatic clock mode activation.

Table 6.4 Notice Expiry and Clock Mode Testing Results

S. No.	Test Case	Result Status
1	NTP Time Synchronization	Successful
2	Notice Expiry Detection	Successful
3	Automatic Notice Removal	Successful
4	Clock Mode Activation	Successful
5	Realtime Date and Time Display	Successful

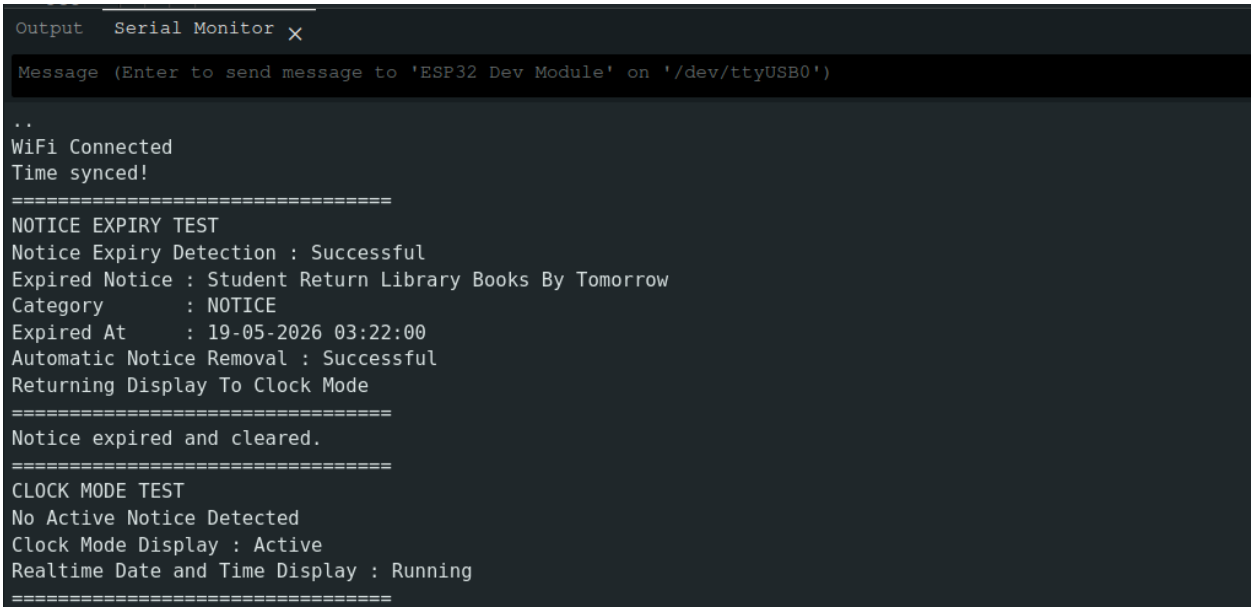


Figure 6.4 Notice Expiry and Clock Mode Testing Logs

6.6 Hardware Testing

Hardware testing was performed to evaluate stable operation of ESP32 microcontroller, HUB75E RGB LED matrix display, WiFi communication, DMA rendering, and GPIO synchronization.

The LED matrix display was tested under continuous realtime synchronization conditions to verify display refresh stability and scrolling behavior. WiFi communication stability and Firebase synchronization reliability were also monitored during long-duration operation.

The hardware implementation maintained stable realtime operation without major synchronization failure or display rendering instability.

Table 6.5 Hardware Testing Results

S. No.	Hardware Component	Result Status
1	ESP32 WiFi Communication	Stable
2	HUB75E Display Rendering	Stable
3	DMA-Based Refresh Operation	Stable
4	Firebase Communication	Stable
5	GPIO Synchronization	Successful
6	Continuous Display Operation	Stable

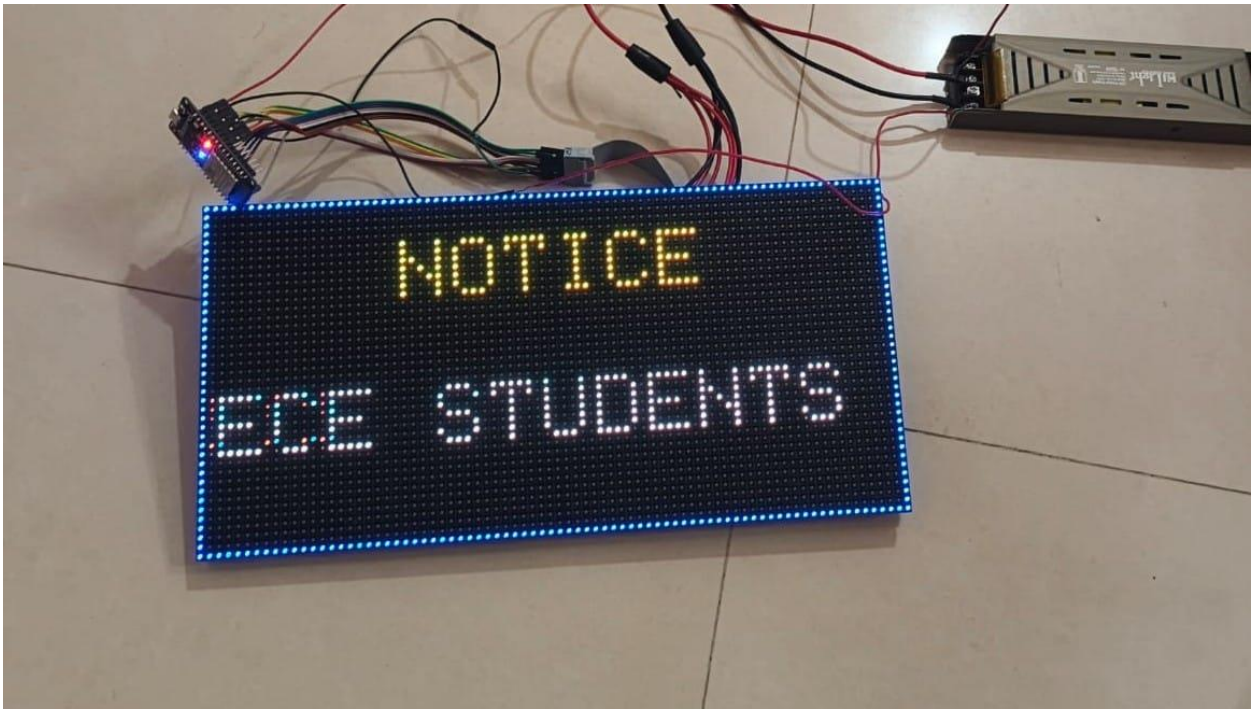


Figure 6.5 Hardware Testing Setup

6.7 Performance Analysis

Performance analysis was conducted to evaluate communication speed, synchronization delay, rendering stability, and operational reliability of the proposed system.

The implementation demonstrated efficient realtime synchronization between Firebase cloud services and ESP32 embedded controller. Notice updates were reflected on the LED matrix display within short synchronization intervals.

DMA-based rendering improved display refresh stability and reduced processor overhead during simultaneous cloud communication and display rendering operations.

The mobile application also maintained responsive UI interaction and stable cloud synchronization during continuous testing operations.

Table 6.6 System Performance Analysis

Parameter	Observation
Cloud Synchronization	Stable
Display Refresh Performance	Smooth
Notice Update Delay	Low
WiFi Communication Stability	Stable
Realtime Database Communication	Reliable
Application Responsiveness	Good

6.8 Challenges During Testing

Several practical challenges were encountered during testing and validation of the proposed system.

The major challenges included:

- Firebase SSL communication instability
- Realtime synchronization delay during unstable internet connectivity
- HUB75E rendering compatibility issues with different driver ICs
- Memory management during JSON processing
- Continuous WiFi communication stability
- GPIO synchronization during DMA rendering

These challenges were minimized through optimized Firebase synchronization intervals, improved memory handling, stable DMA rendering configuration, and continuous debugging of ESP32 firmware.

6.9 Results and Discussion

The proposed “IoT-Enabled Digital Notice Board” system successfully achieved realtime cloud communication, secure authentication, remote notice management, and stable LED matrix display synchronization.

The Flutter mobile application provided responsive user interaction and centralized notice management functionality, while Firebase cloud services maintained stable realtime synchronization between the application and ESP32 embedded controller.

The ESP32 successfully synchronized notices, display settings, and command operations in realtime and continuously updated the HUB75E RGB LED matrix display without major communication failure.

The implementation demonstrated efficient integration of IoT technology, embedded communication systems, cloud services, and mobile application development suitable for realtime digital notice management applications.

6.10 Summary

This chapter presented testing procedures, experimental observations, synchronization validation, hardware testing, and performance analysis of the proposed “IoT-Enabled Digital Notice Board” system.

The testing results confirmed stable realtime cloud communication, reliable Firebase synchronization, secure authentication, responsive mobile application behavior, and efficient LED matrix display operation. The proposed system successfully achieved practical realtime digital notice management using IoT and cloud communication technologies.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The proposed “IoT-Enabled Digital Notice Board” system successfully achieved realtime digital notice management using IoT technology, Firebase cloud services, ESP32 embedded controller, Flutter mobile application framework, and HUB75E RGB LED matrix display technology.

The system successfully integrated cloud communication, secure authentication, realtime synchronization, embedded display control, and remote notice management into a centralized digital communication platform. Firebase cloud services provided stable realtime communication and secure cloud-based data management, while ESP32 continuously synchronized notices and display settings through WiFi communication.

The Flutter mobile application enabled authorized users to remotely publish notices, configure display settings, and monitor active notice status through an internet-connected interface. Realtime synchronization between Firebase and ESP32 significantly improved communication efficiency and eliminated the limitations of traditional manually updated notice boards.

Additional features such as automatic notice expiry handling, NTP-based clock synchronization, scrolling text display, brightness control, color customization, and realtime display configuration further improved usability and operational flexibility of the proposed system.

DMA-based rendering architecture improved HUB75E LED matrix refresh stability and enabled smooth scrolling display performance during continuous realtime cloud synchronization operations.

The implementation demonstrated that low-cost IoT hardware and cloud communication technologies can be effectively integrated to develop scalable, paperless, and realtime digital

communication systems suitable for educational institutions, offices, and public information display applications.

The proposed system successfully fulfilled all major project objectives related to realtime cloud synchronization, secure authentication, remote notice management, embedded display control, and wireless communication.

7.2 Future Scope

The proposed system can be further improved and expanded with additional features and advanced communication capabilities in future implementations.

Possible future enhancements include:

- Multi-display centralized management system
- Voice announcement integration
- Multimedia content support
- Web-based administration dashboard
- AI-based automated notice scheduling
- Touchscreen display integration
- Mobile notification integration
- Offline synchronization support
- Multi-user role management
- Advanced analytics and monitoring system
- Larger LED matrix display support
- OTA firmware update functionality
- MQTT-based communication architecture
- Smart campus integration

The modular architecture of the proposed system allows efficient scalability and future integration of advanced IoT communication technologies and intelligent automation features.

REFERENCES

1. Espressif Systems, *ESP32 Series Datasheet*, 2024. https://www.espressif.com/en/products/socs/esp32
2. Google Firebase Documentation, *Firebase Realtime Database and Authentication Documentation*. https://firebase.google.com/docs
3. Flutter Documentation, *Flutter Application Development Framework*. https://docs.flutter.dev
4. Arduino Documentation, *ESP32 Programming using Arduino IDE*. https://docs.arduino.cc
5. Mobizt, *Firebase ESP Client Library for ESP32*. https://github.com/mobizt/Firebase-ESP-Client
6. Raj Kamal, *Internet of Things: Architecture and Design Principles*, McGraw Hill Education. https://www.mheducation.co.in
7. Vijay Madisetti and Arshdeep Bahga, *Internet of Things – A Hands-On Approach*, Universities Press. https://www.universitiespress.com
8. HUB75 RGB LED Matrix Documentation, *Display Interface and Communication Standards*. https://learn.adafruit.com/32x16-32x32-rgb-led-matrix
9. Flutter Bloc Documentation, *Bloc and Cubit State Management*. https://bloclibrary.dev
10. NTP Pool Project Documentation, *Network Time Protocol Synchronization Services*. https://www.ntppool.org/en/

APPENDIX A

ESP32 Firmware Code Snippets

This appendix contains important ESP32 firmware code snippets used for WiFi communication, Firebase cloud synchronization, realtime database communication, NTP synchronization, notice expiry handling, and HUB75E RGB LED matrix display control.

The firmware was developed using Arduino IDE and Firebase ESP Client Library for realtime communication between ESP32 and Firebase cloud services.

The appendix includes:

- WiFi connection logic
- Firebase initialization code
- Firebase Realtime Database communication
- NTP synchronization code
- Notice expiry handling logic
- HUB75E display rendering functions
- Realtime synchronization logic

A.1 WiFi Connection Logic

The following code establishes WiFi communication between the ESP32 controller and wireless network for internet-based Firebase cloud synchronization.

```
// ==========================
// WIFI CONNECTION
// ==========================

const char* ssid = "wifi";
const char* password = "wifi123456";

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

  delay(500);
  Serial.print(".");
}

Serial.println("\nWiFi Connected");
```

A.2 Firebase Initialization Code

The following code initializes Firebase cloud services and establishes communication between ESP32 and Firebase Realtime Database.

```
// ==========================
// FIREBASE INITIALIZATION
// ==========================

#define API_KEY "YOUR_API_KEY"

#define DATABASE_URL "YOUR_DATABASE_URL"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

config.api_key = API_KEY;
config.database_url = DATABASE_URL;

Firebase.signUp(&config, &auth, "", "");

Firebase.begin(&config, &auth);

Firebase.reconnectWiFi(true);
```

A.3 Firebase Realtime Database Communication

The following code continuously fetches notice data from Firebase Realtime Database and synchronizes it with ESP32.

```
// ==========================
// NOTICE FETCH
// ==========================

if (Firebase.RTDB.getJSON(
    &fbdo,
    "/noticeBoard/ESP32_02_11_004/Notice")) {

  FirebaseJson &json = fbdo.jsonObject();

  FirebaseJsonData res;

  if (json.get(res, "message"))
      message = res.stringValue;

  if (json.get(res, "category"))
      category = res.stringValue;
}
```

A.4 NTP Time Synchronization Code

The following code synchronizes ESP32 with Network Time Protocol (NTP) server for accurate realtime date and time management.

```
// ==========================
// NTP TIME SYNCHRONIZATION
// ==========================

configTime(19800, 0, "pool.ntp.org");

struct tm timeinfo;

while (!getLocalTime(&timeinfo)) {

  Serial.println("Waiting for NTP time...");

  delay(1000);
}

Serial.println("Time synced!");
```

A.5 Notice Expiry Handling Logic

The following code automatically removes expired notices based on realtime NTP synchronized timestamps.

```
// ==========================
// NOTICE EXPIRY CHECK
// ==========================

if (expiryAt > 0)
{
  struct tm timeinfo;

  if (getLocalTime(&timeinfo))
  {
    time_t now;

    time(&now);

    unsigned long long currentMillis =
      (unsigned long long) now * 1000ULL;

    if (currentMillis >= expiryAt)
    {
      clearCurrentNotice();

      Serial.println("Notice expired and removed");
    }
  }
}
```

A.6 HUB75E Display Rendering Functions

The following code renders scrolling notices on the HUB75E RGB LED matrix display using DMA-based rendering architecture.

```
// ==========================
// DISPLAY RENDERING
// ==========================

dma_display->clearScreen();

dma_display->setBrightness8(brightness);

dma_display->setTextColor(
  getColor(bodyTextColor)
);

dma_display->setCursor(scrollX, 22);

dma_display->print(message);

scrollX -= scrollSpeed;

int len = message.length() * 6;

if (scrollX < -(len + 20))
    scrollX = PANEL_RES_X;
```

A.7 Realtime Synchronization Logic

The following code continuously synchronizes Firebase cloud data with ESP32 at fixed intervals for realtime display updates.

```
// ==========================
// REALTIME SYNCHRONIZATION
// ==========================

static unsigned long lastFetch = 0;

if (millis() - lastFetch > 3000) {

  lastFetch = millis();

  Firebase.RTDB.getJSON(
    &fbdo,
    "/noticeBoard/ESP32_02_11_004/Notice"
  );

  Firebase.RTDB.getJSON(
    &fbdo,
    "/noticeBoard/ESP32_02_11_004/Settings/settings"
  );

  Firebase.RTDB.getJSON(
    &fbdo,
    "/noticeBoard/ESP32_02_11_004/Settings/commands"
  );
}
```

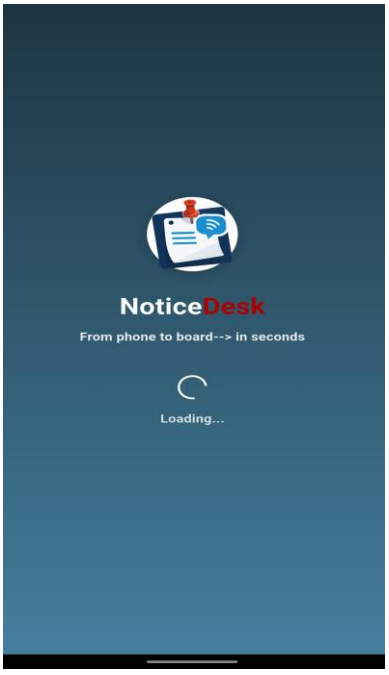
APPENDIX B
Mobile Application Screens

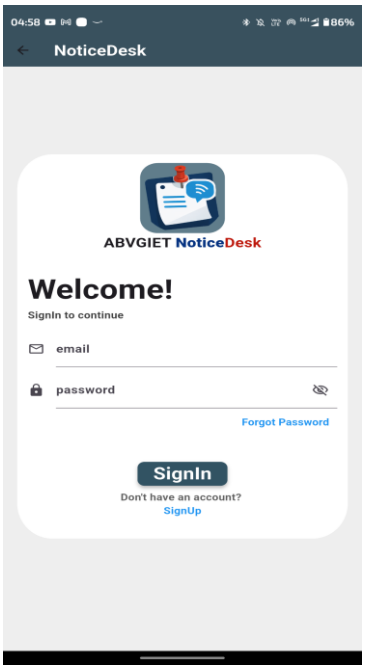
This appendix contains additional screenshots of the Flutter mobile application developed for the proposed system.

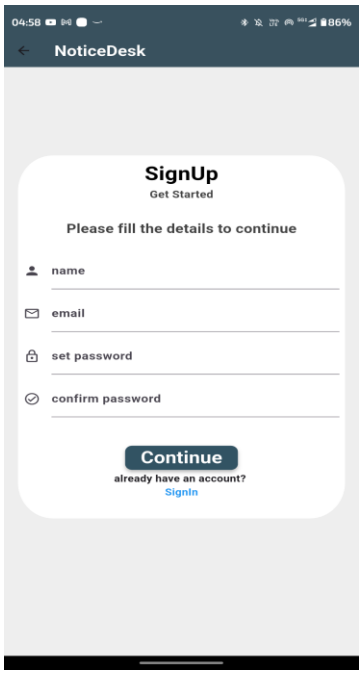
The screenshots demonstrate different application modules and user interface components including authentication screens, dashboard interface, notice publishing interface, display settings management, realtime monitoring screens, and notice expiry configuration interface.

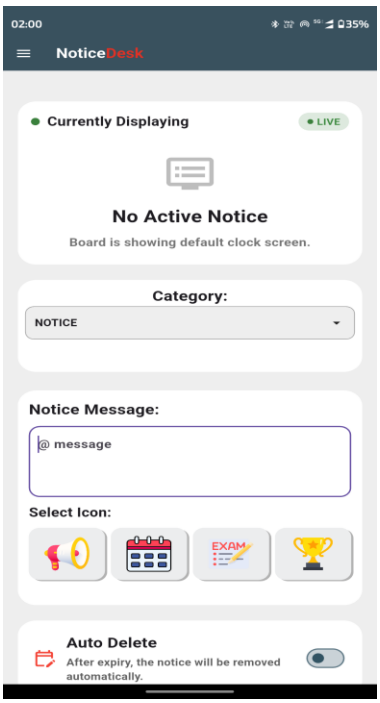
The appendix includes:

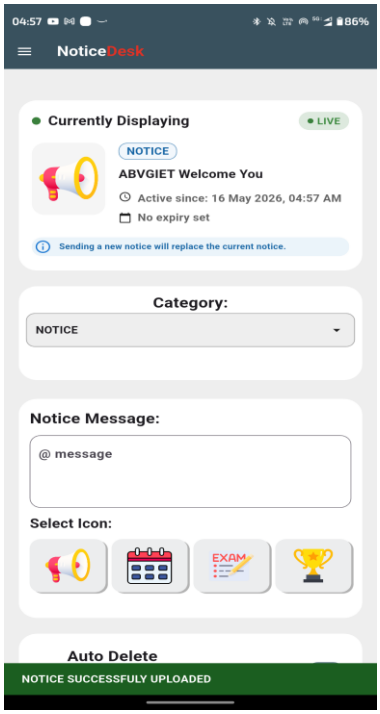
- Splash screen
- Login screen
- Signup screen
- Email verification screen
- Forgot password screen
- Dashboard interface
- Notice publishing interface
- Active notice monitoring screen
- Display settings interface
- Color customization interface
- Brightness control interface
- Scroll speed management interface

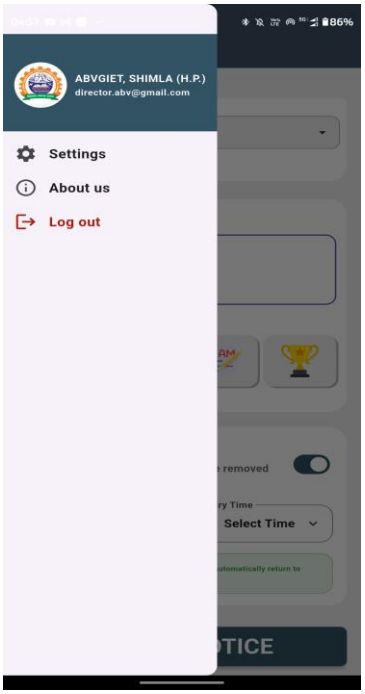

NoticeDesk
From phone to board--> in seconds
Loading...

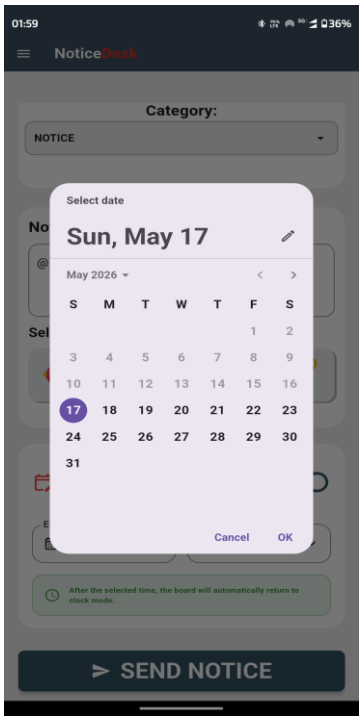

NoticeDesk
ABVGIET NoticeDesk
Welcome!
SignIn to continue
email
password
Forgot Password
SignIn
Don't have an account?
SignUp

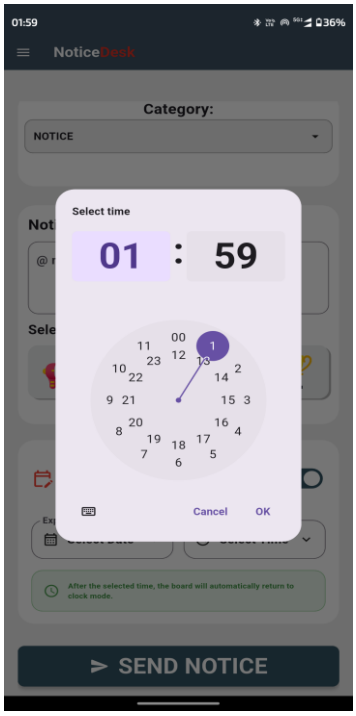

NoticeDesk
SignUp
Get Started
Please fill the details to continue
name
email
set password
confirm password
Continue
already have an account?
SignIn

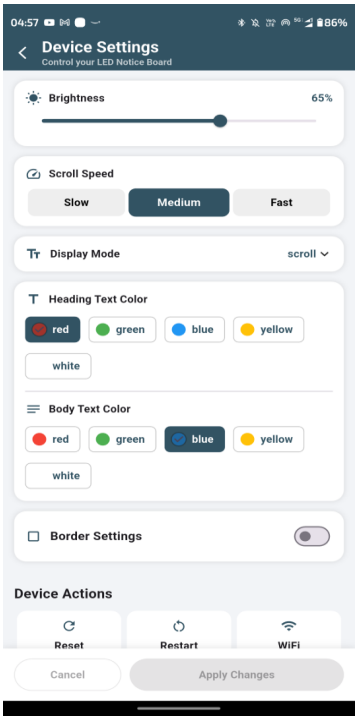

NoticeDesk
Currently Displaying
LIVE
No Active Notice
Board is showing default clock screen.
Category:
NOTICE
Notice Message:
@ message
Select Icon:
EXAM
Auto Delete
After expiry, the notice will be removed automatically.

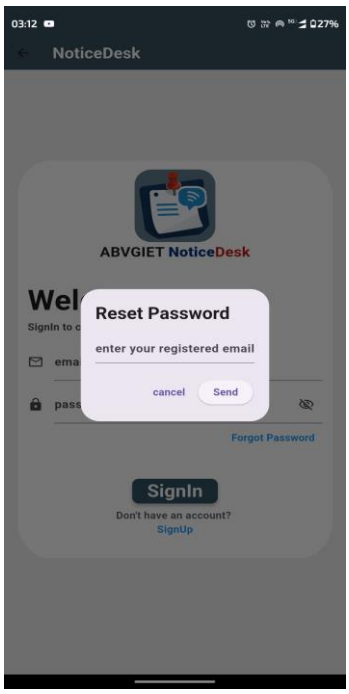

NoticeDesk
Currently Displaying
LIVE
NOTICE
ABVGIET Welcome You
Active since: 16 May 2026, 04:57 AM
No expiry set
Sending a new notice will replace the current notice.
Category:
NOTICE
Notice Message:
@ message
Select Icon:
EXAM
Auto Delete
NOTICE SUCCESSFULY UPLOADED

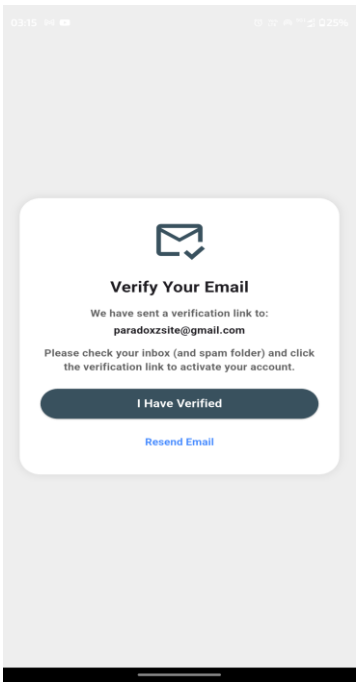

ABVGIET, SHIMLA (H.P.)
director.abv@gmail.com
Settings
About us
Log out
Select Time

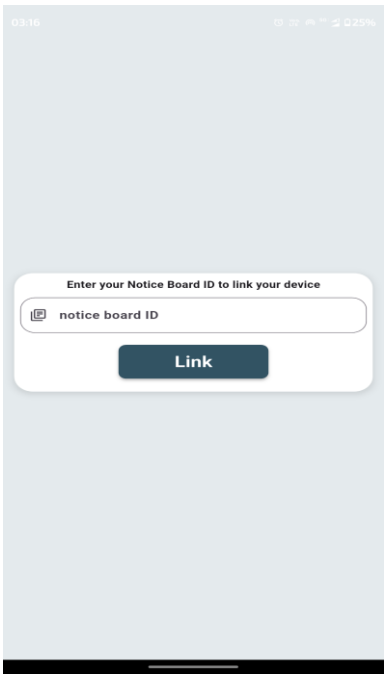

NoticeDesk
Category:
NOTICE
Select date
Sun, May 17
May 2026
Cancel
OK
SEND NOTICE


NoticeDesk
Category:
NOTICE
Select time
01 : 59
Cancel
OK
SEND NOTICE


Device Settings
Control your LED Notice Board
Brightness
65%
Scroll Speed
Slow
Medium
Fast
Display Mode
scroll
Heading Text Color
red
green
blue
yellow
white
Body Text Color
red
green
blue
yellow
white
Border Settings
Device Actions
Reset
Restart
WiFi
Cancel
Apply Changes


NoticeDesk
ABVGIET NoticeDesk
Reset Password
enter your registered email
cancel
Send
Forgot Password
SignIn
Don't have an account?
SignUp


Verify Your Email
We have sent a verification link to:
paradoxzsite@gmail.com
Please check your inbox (and spam folder) and click the verification link to activate your account.
I Have Verified
Resend Email


Enter your Notice Board ID to link your device
notice board ID
Link