SENIOR DESIGN II

SCAN: Smart Check-in & Analytics Network

University of Central Florida
Department of Electrical and Computer Engineering

Senior Design 2 Final Report

GROUP 35

Cory Brynds
Computer Engineering

DeAndre Hendrix
Computer Engineering

Jonah Sprandel
Electrical Engineering

REVIEWERS
Dr. Mike Boroczak, ECE
Dr. Sonali Das, ECE
Dr. Piotr Kulik, ECE

ADVISOR
Dr. Lei Wei

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY....1
2. PROJECT DESCRIPTION....2
2.1 Project Background and Motivation....2
2.2 Project Function....2
2.3 Related Projects and Technologies....3
2.3 Project Objectives and Goals....6
2.3.1 Hardware Objectives....7
2.3.2 Software Objectives....8
2.4 Requirement Specifications....11
3. TECHNOLOGY RESEARCH....14
3.1 Processors....14
3.1.1 Microcontroller (MCU)....14
3.1.2 Field-Programmable Gate Array (FPGA)....15
3.1.3 System-on-Chip (SoC)....16
3.1.4 Processor Part Comparison and Selection....18
3.2 Proximity Sensors....20
3.2.1 Ultrasonic Sensing Technologies....20
3.2.2 Infrared Sensing Technologies....21
3.2.3 Capacitive Sensing Technologies....21
3.2.4 Proximity Sensor Selection....22
3.3 User Authentication Methods....23
3.3.1 Password Authentication....24
3.3.2 Fingerprint Authentication....24
3.3.3 Hardware Authentication....25
3.3.4 Keypad Component Selection....26
3.3.6 Smart Card Component Selection....27
3.4 Long-Range Communication Technologies....28
3.4.1 WiFi....28
3.4.2 Long-Range Radio (LoRa)....29
3.4.3 LTE for Machines (LTE-M)....29
3.4.4 Long-Range Communication Technology Selection....30
3.5 Short-Range Wireless Communication Technologies....31
3.5.1 Radio Frequency Identification....31
3.5.2 Near-Field Communication....32
3.5.3 Infrared Communication....32
3.5.5 NFC Reader Component Selection....34
3.6 Display Technologies....35

3.6.1 Liquid Crystal Display (LCD)..... 36
3.6.2 Thin-Film Transistor (TFT) Display..... 36
3.6.3 Organic Light-Emitting Diode (OLED) Display..... 37
3.6.4 TFT Component Selection..... 38
3.7 Batteries..... 40
3.7.1 Lead Acid..... 40
3.7.2 Lithium-Ion (Including LiPo and LiFePO4)..... 41
3.7.3 Nickel-Metal Hydride (NiMH)..... 42
3.7.4 Component Selection for Lithium-Ion Battery..... 43
3.8 Database Technologies..... 44
3.8.1 Relational Databases..... 44
3.8.2 NoSQL Databases..... 45
3.8.3 Cloud Databases..... 46
3.8.4 Database Selection..... 47
3.9 Frontend Frameworks..... 49
3.9.1 Angular..... 50
3.9.2 ReactJS..... 50
3.9.3 Vue..... 51
4. STANDARDS AND DESIGN CONSTRAINTS..... 53
4.1 Design Standards..... 53
4.1.1 Wireless Communication Standards..... 53
4.1.1.1 Near-Field Communication Standard..... 53
4.1.1.2 WiFi Standard..... 56
4.1.2 Wired Communication Standards..... 58
4.1.2.1 Serial-Peripheral Interface (SPI) Standard..... 58
4.1.2.2 Inter-Integrated Circuit Standard..... 59
4.1.3 Programming Standards..... 60
4.1.4 Power Standards..... 61
4.1.4.1 USB Power Delivery Standard..... 61
4.2 Design Constraints..... 62
4.2.1 Economic Constraints..... 62
4.2.2 Time Constraints..... 63
4.2.3 Sustainability Constraints..... 64
4.2.4 Environmental Constraints..... 65
4.2.5 Social and Political Constraints..... 66
4.2.6 Ethical Constraints..... 66
4.2.7 Health and Safety Constraints..... 67
4.2.8 Manufacturability Constraints..... 67

5. USE OF LARGE LANGUAGE MODELS IN REPORT WRITING 69
5.1 Statement on Use of LLMs for Report Writing 69
5.2 Large Language Models 69
5.2.1 ChatGPT 69
5.2.2 Perplexity 70
5.3 Strengths of LLMs 70
5.4 Limitations of LLMs 71
5.5 LLM Case Study 71
6. HARDWARE DESIGN 75
6.1.1 Power Supply Design 77
6.1.1.1 USB Power Delivery Sink Controller 77
6.1.1.2 Power Management Controller and Buck Converter 78
6.1.1.1 Senior Design II Updates 80
6.1.2 Automatic Boot Mode for ESP32 80
6.1.3 NFC Transceiver Schematic Design 82
6.1.3.1 Senior Design II Updates 83
6.1.4 Display Design 83
6.1.5 Keypad and IO Bus Expander 84
6.1.6 Other Connections 86
6.1.7 Senior Design II Major Schematic Revisions 86
7. SOFTWARE DESIGN 87
7.1 Software Development Framework and Tools 88
7.1.1 Visual Studio Code 88
7.1.2 GitHub 88
7.1.3 Docker Development Environment 89
7.1.4 ESP-IDF Framework 89
7.2 Kiosk Software 90
7.2.1 Kiosk User Mode 90
7.2.2 Kiosk Administrator Mode 91
7.2.3 Proximity Sensor Software 92
7.2.4 Card Interface Software 92
7.2.4.1 Senior Design II Updates to Card Interface Software 94
7.2.5 Kiosk Display Interface Software 95
7.2.5.1 Senior Design II Updates to Display Interface Software 96
7.3 Database Design 97
7.4 SCAN Web Application 99
7.4.1 Web Application Key Features 99
7.4.2 Technical Implementation 102

7.5 Over-the-Air Update Implementation....102
8. SYSTEM FABRICATION....105
8.1 PCB Design Process....105
8.2 PCB Manufacturers....107
8.3 Enclosure Fabrication....108
9. SYSTEM TESTING AND INTEGRATION....111
9.1 Testing & Evaluation....111
9.1.1 ESP32-C6 Testing....111
9.1.2 Proximity Sensor Testing....111
9.1.3 NFC Transceiver Testing....112
9.1.3.1 Senior Design II Update....113
9.1.4 Display Testing....114
9.1.4.1 Senior Design II Update....114
9.1.5 Keypad Testing....114
9.1.7 WiFi Interface Testing....115
9.1.8 Database Software Testing....116
9.1.9 Web Interface Testing....117
9.1.10 PCB Testing....119
9.1.11 Requirements Testing....119
9.2 System Integration....121
10. ADMINISTRATIVE CONTENT....124
10.1 Project Budget and Financing....124
10.2 Bill of Materials....124
10.2.1 Senior Design II Updated Budget....127
10.3 Project Milestones....128
10.4 Work Distribution....129
11. CONCLUSION....131
APPENDIX A - REFERENCES....132
APPENDIX B - COPYRIGHT PERMISSIONS....139
APPENDIX C - LLM PROMPTS AND RESPONSES....144
1.0 Perplexity Prompts and Responses....144
1.1 Microcontroller Selection....144
1.2 Table Comparison of Microcontrollers....145
1.3 Extended Table Comparison of Microcontrollers....145
APPENDIX D - REFERENCE SCHEMATICS....147
1.0 PN532 Breakout Schematic (Adafruit Industries)....147
2.0 PCF8574 Application Circuit (From Texas Instruments Datasheet)....148
3.0 CYPD3177 Datasheet Reference Schematic [72]....149

4.0 BQ25798 Datasheet Reference Schematic [73].....150
5.0 LM62460-Q1 Datasheet Reference Schematic [74].....151
6.0 ESP32-C6-DevKitC-1 Schematic [75].....152
APPENDIX E - SCAN FULL SCHEMATIC.....153

LIST OF FIGURES

Figure 2.1 Rendering of SCAN Kiosk Prototype..... 3
Figure 2.2 Eventbrite Handheld Check-In Scanner..... 4
Figure 2.3 Envoy Visitors Kiosk Dashboard..... 4
Figure 2.4 Envoy Visitors Kiosk Dashboard..... 5
Figure 2.5 SCAN System Diagram.....8
Figure 2.6 Embedded Software Flowchart..... 9
Figure 2.7 Frontend and Backend Software Flowchart..... 11
Figure 2.8 House of Quality Diagram.....13
Figure 3.1 LCD Layer Composition (Obtained from Britannica)..... 36
Figure 3.2 TFT Display Composition (Obtained from RS DesignSpark)..... 37
Figure 3.3 OLED Display Composition (Obtained from OLED Devices).....37
Figure 4.1 NFC Specification Architecture (Obtained from NFC Forum).....55
Figure 4.2 SPI Bus Diagram..... 58
Figure 4.3 I2C Bus Diagram (Obtained from embedded systems lab manual)..... 59
Figure 4.4 I2C Read Command (Obtained from embedded systems lab manual)..... 60
Figure 4.5 I2C Write Command (Obtained from embedded systems lab manual).....60
Figure 6.1 SCAN Schematic Block Diagram..... 75
Figure 6.2 SCAN System Schematic.....76
Figure 6.3 Power Delivery Block Diagram..... 77
Figure 6.4 USB Power Delivery Schematic..... 77
Figure 6.5 Power Management Schematic..... 79
Figure 6.6 Logic Power Schematic.....80
Figure 6.7 ESP32 USB-to-UART Bridge..... 81
Figure 6.8 ESP32 Auto Downloader..... 82
Figure 6.9 NFC Transceiver Schematic Design (Referenced from Adafruit)..... 82
Figure 6.10 TFT Display Schematic.....84
Figure 6.11 Shared SPI Bus Interface.....84
Figure 6.12 PCF8574 default I2C address configuration..... 85
Figure 6.13 IO Bus Expanders Schematic..... 85
Figure 6.14 Shared I2C Bus Interface.....86
Figure 7.1 SCAN Use Case Diagram..... 87
Figure 7.2 Software Flowchart for Kiosk Application..... 90
Figure 7.3 Software Flowchart for Administrator Mode.....91
Figure 7.5 NDEF Library Classes Diagram.....93
Figure 7.6 Card Reader Software Flowchart..... 94
Figure 7.7 Display Addressable Region and Frames Library.....95
Figure 7.8 Display Software Flowchart (Kiosk User Mode).....96

Figure 7.9 Database Fields....97
Figure 7.10 Database Class Diagram....98
Figure 7.11 Analytics Dashboard Page....100
Figure 7.12 Activity Log Page....100
Figure 7.13 User Management Page Wireframe....101
Figure 7.14 Kiosk Management Page....101
Figure 7.15 Sequential Diagram of OTA Update from HTTP Server....103
Figure 8.1 PCB Layout (Top Layer and GND1)....106
Figure 8.2 PCB 3D Model....106
Figure 8.3 Wireframe Rendering of Kiosk Enclosure....109
Figure 8.4 Proposed PCB Mounting Scheme....110
Figure 9.1 Breadboard Testing Setup for PN532 Development Board....112
Figure 9.2 Terminal Output of ReadTag Example Program....113
Figure 9.3 Terminal Output of WriteTag Example Program....113
Figure 9.4 Display Graphics Test....114
Figure 9.5 Breadboard Testing Setup for Keypad Testing with IO Expander....115
Figure 9.6 Wi-Fi Connection Test....116
Figure 9.7 Database Interfacing Test - Terminal Output....116
Figure 9.8 Database Interfacing Testing - Firebase Web-Viewer....117
Figure 9.10 System Integration Timeline....123

LIST OF TABLES

Table 2.1 Project Goals..... 6
Table 2.2 SCAN Requirement Specifications..... 12
Table 3.1 Comparison of Processor Technologies..... 17
Table 3.2 Comparison of SoCs..... 19
Table 3.3 Comparison of Proximity Sensing Technology..... 22
Table 3.4 Comparison of PIR Sensors..... 23
Table 3.5 Comparison of Authentication Methods..... 26
Table 3.6 Comparison of Keypads..... 27
Table 3.7 Comparison of Smart Cards..... 28
Table 3.8 Comparison of Long-Range Communication Technologies..... 30
Table 3.9 Comparison of Close-Range Communication Technologies..... 33
Table 3.10 Comparison of NFC Reader ICs..... 35
Table 3.11 Comparison of Display Technologies..... 38
Table 3.12 Comparison of TFT Displays..... 40
Table 3.13 Comparison of Battery Technologies..... 42
Table 3.14 Comparison of Lithium-Ion Batteries..... 44
Table 3.15 Comparison of Database Technologies..... 46
Table 3.16 Comparison of Cloud Databases..... 49
Table 3.17 Comparison of Frontend Frameworks..... 51
Table 4.1 Summary of Standard NFC Tag Types..... 54
Table 4.2 NDEF Record Format..... 56
Table 4.3 SPI Modes of Operation..... 59
Table 5.1 MCU Comparison by Perplexity..... 73
Table 5.2 Expanded MCU Comparison by Perplexity..... 74
Table 6.1 Host Interface Selection Bits (Taken from PN532 Datasheet)..... 83
Table 8.1 Comparison of PCB Vendors..... 108
Table 9.1 Web Interface Test Cases..... 118
Table 9.2 SCAN Requirement Specifications and Test Plan..... 119
Table 10.1 Per-Kiosk Budget Breakdown..... 124
Table 10.2 Project Bill of Materials..... 125
Table 10.4 Senior Design 1 Milestones..... 128
Table 10.5 Senior Design 2 Project Milestones..... 129
Table 10.6 Work Distribution..... 130

1. EXECUTIVE SUMMARY

SCAN is a low-cost, IoT-driven kiosk system designed to provide real-time analytics based on check-in and check-out information. By integrating NFC-based authentication, real-time data processing, and a user-friendly interface, SCAN provides users with a near-effortless check-in experience and provides facilities with actionable insights for improving operations. SCAN is able to track attendance and provide detailed analytics on occupancy trends, peak usage hours, and visit durations. These insights are accessible to administrator and facility managers via SCAN's web application, which can provide them with the necessary statistics to allocate resources efficiently, optimize staffing, and improve user experiences.

This research document serves as a guide to the design and implementation of the SCAN system. Organized into key chapters, the document outlines:

1. **Project Description**: Overviews the motivations behind the SCAN project, its core functionality, and the specifications to which SCAN has been evaluated on.
2. **Technology Research**: Compares numerous devices, communication protocols, and software tools to find the best technologies for the SCAN project.
3. **Standards and Design Constraints**: Explores the technology standards and design constraints which guided the development of the SCAN project.
4. **Use of LLMs in Report Writing**: Explores case studies of using LLMs such as ChatGPT to aid in technical report writing, and discusses their limitations.
5. **Hardware Design**: Details the full system schematic for SCAN's hardware
6. **Software Design**: Explores the software frameworks used to develop SCAN's kiosk application, database, and user interface.
7. **System Fabrication**: Explains PCB design best practices and the fabrication procedure for the SCAN enclosure.
8. **System Testing and Integration**: Covers testing of all individual components of SCAN's hardware and software and the plans for integrating these components into one functional design.
9. **Administrative Content**: Includes the project budget, bill of materials, project milestones, and distribution of work between SCAN's members.

The document's purpose extends beyond the technical implementation—it demonstrates how the SCAN team, Senior Design Group 35, can translate abstract goals into a functional, impactful product while adhering to engineering best practices. It highlights design decisions, challenges encountered, and trade-offs made during development.

2. PROJECT DESCRIPTION

This chapter outlines the motivations behind the project and provides an overview of the SCAN kiosk's function and subsystems. In Section 2.1, we provide background information on the motivations behind the SCAN kiosk, and in sections 2.2-2.4 we describe the high-level functionality, objectives, and requirement specifications for the project.

2.1 Project Background and Motivation

At a large public university like UCF, data collection is not just important—it is essential. Graduation rates, enrollment statistics, and faculty-to-student ratios are carefully tracked each year and reported to institutions for accreditation and ranking. Another measurement often overlooked is engagement: How many students attend major university events? How long are they staying? What are their activity patterns week-to-week throughout the academic year? The answers to these questions directly influence key decisions on resource allocation, staffing, and campus planning. Without access to accurate, accessible engagement data, the university risks missing insights that could impact its future.

UCF students, faculty, and staff already use their UCF ID cards—embedded with a combination of magnetic stripe and RFID technology—for many daily functions, whether it's scheduling appointments at Student Health Services, accessing restricted buildings, or checking into locations like the UCF Recreation and Wellness Center (RWC). Due to their widespread use, these cards present a prime and overlooked opportunity to streamline engagement data collection, uniquely identifying each student and capturing valuable insights in real time.

The UCF RWC is a prime case example. With an occupancy tracking system in place, the RWC could transform and adjust its daily operations. Staff could predict peak hours precisely, manage staffing levels throughout the week, and even gather detailed demographic data on the types of students utilizing the facility. By tracking this engagement data, not only is the efficiency of the RWC's operations enhanced, but the experiences of individual attendees are improved by ensuring that resources are allocated exactly when and where they're needed most.

2.2 Project Function

We propose SCAN: a Smart Check-in and Analytics Network capable of tracking such user data. Using the UCF Recreation and Wellness Center as our example, we have designed a low-cost scalable network of kiosks equipped with an NFC reader and interactive display to log check-ins and check-outs of RWC attendees. Multiple kiosks can be used in synchronicity to provide more than one access point to the facility. Encoded NFC tags emulate UCFIDs to avoid any security concerns involving the direct use of actual IDs. For authentication and analytics, user data is stored in a secure database on a cloud server external to the kiosk, and the kiosk queries the database to ensure that the user has permission to access the area. In addition, a web application displays analytics based on the check-in data, which can be made available to staff and students to

enhance the user experience. Figure 2.1 shows a rendering of the SCAN kiosk prototype, where the key features of the kiosk are displayed and labelled.

Figure 2.1 Rendering of SCAN Kiosk Prototype

2.3 Related Projects and Technologies

To better understand the landscape of smart kiosk technologies and inform the design of the SCAN system, we analyze existing software and kiosk-based products currently in the market. These solutions range from simple check-in interfaces to advanced data-driven systems that include analytics and personalized user experiences.

Eventbrite Check-In: Eventbrite offers a smart check-in application compatible with IOS and Android devices and is designed to manage ticketing and entry at events. It utilizes a combination of QR codes and NFC-enabled ticketing, allowing users to check in with mobile devices or pre-printed tickets. The software integrates directly with the Eventbrite platform, providing real-time event occupancy, attendance data, and demographic analysis. [1] Its seamless use of NFC technology and integration with a large analytics platform makes the Eventbrite kiosk a strong reference point for the SCAN system. However, Eventbrite is targeted as a one-size-fits-all solution for primarily one-time events, rather than recurring check-ins such as those at the RWC.

Figure 2.2 Eventbrite Handheld Check-In Scanner

Envoy Visitors Kiosk: Another check-in solution, the Envoy Visitors kiosk, provides check-in services for offices and similar environments. It features an interactive touchscreen, visitor badge printing capabilities, and analytics for employee and visitor management. The configurable software of Envoy enables customized workflows for different types of visitors and regular users. Envoy's system is designed with regularity in mind, making it adaptable to specific check-in scenarios like the RWC. However, its extensive set of features and customizable software come at a cost: the Envoy Visitors software has a monthly subscription plan with pricing up to $330 per month [2]. The SCAN team aimed to incorporate the customizable software element for tailored analytics but with a much more affordable pricing plan.

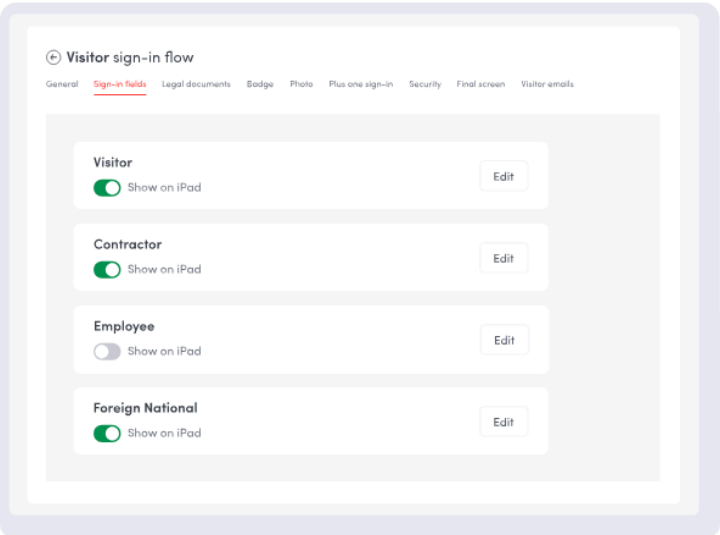


Figure 2.3 Envoy Visitors Kiosk Dashboard

PerfectGym Kiosk: The Portland-based company PerfectGym provides check-in hardware and software specifically designed for gyms and other recreation facilities. They offer products such as a self-service front desk kiosk, which can be used to automate check-in and check-out procedures, reduce front desk wait queues, and store all analytics data (e.g. monthly revenue, daily occupancy) in one centralized system. It includes an integrated scanner, printer, camera, and payment terminal to provide an all-in-one solution and eliminate the need for physical employees in the sign-up and check-in process. The kiosk accepts RFID and barcode-based cards to facilitate the check-in process [3]. While this system has many features, its feature set is beyond the scope of what is useful in a setting such as the RWC, where many of the features such as payment, account creation, and class booking are handled by pre-existing software external to the kiosk. The SCAN system adapts many of the check-in features and user interface elements from the PerfectGym kiosk while focusing only on the relevant aspects.

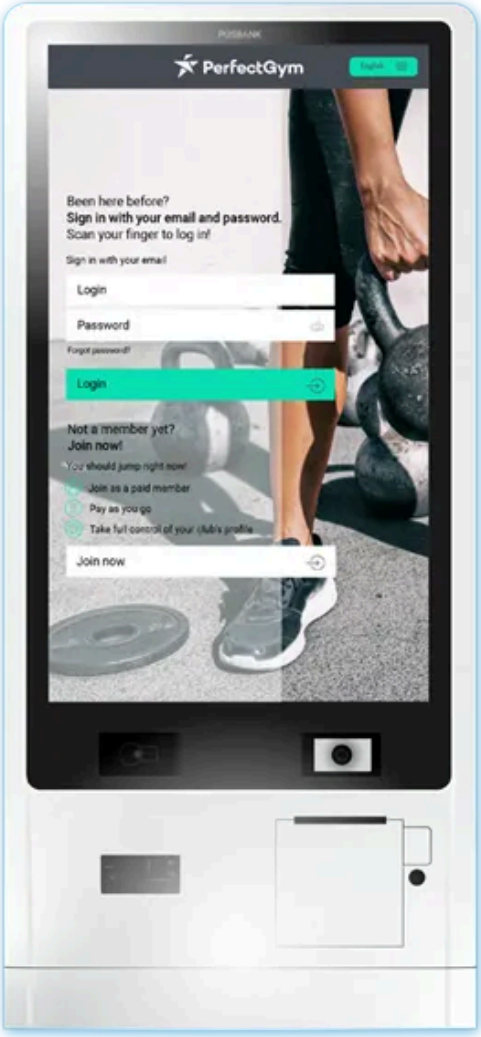


Figure 2.4 Envoy Visitors Kiosk Dashboard

Each check-in solution provides a unique and adaptable platform for processing user check-ins and analyzing the resulting data. The SCAN system aimed to take the best aspects of each solution and produce a scalable, low-cost alternative, ready to be deployed at the RWC or a similar environment to process check-in data reliably.

2.3 Project Objectives and Goals

This project aimed to provide a low-cost, scalable internet-of-things (IoT) alternative to existing check-in and authentication kiosks, opting for NFC-based authentication to utilize existing technology within UCF's ecosystem (the UCFID). Throughout the report, we use the UCF Recreation and Wellness Center as a use case, as it is a prime demonstration of the technology's use and is relatable to many students. However, the technology can be adapted for various applications at UCF and beyond. The goals for the project, shown in Table 2.1, are divided into three primary categories. Basic goals encompass the baseline functionality of our project. Our advanced goals increase the usability and reliability of the SCAN system. Finally, our stretch goals are nice-to-have features that further increase the utility and security of the system but are not necessary to its functionality.

Table 2.1 Project Goals

Basic Goals
SCAN can be used to identify, authenticate, and check in users.
SCAN can collect and track user data through a user database.
SCAN can analyze user data and provide useful statistics.
A web application accompanying SCAN displays the collected data.
Advanced Goals
SCAN can operate in administrator mode, where it can write user IDs to cards.
SCAN can operate as a network of kiosks, providing synchronous check-in and analytics from multiple locations.
SCAN's web application is mobile-friendly.
Stretch Goals
SCAN is capable of being charged by solar power.
SCAN can receive over-the-air updates for firmware patches and diverse use-cases.
The SCAN web app implements lazy loading for more efficient data fetching.

The following hardware and software objectives for the project outline the specific feature sets enabling our basic, advanced, and stretch goals.

2.3.1 Hardware Objectives

The primary function of the SCAN kiosk is to verify a user's identity and either grant or deny the user access to the UCF Recreation and Wellness Center. Shown in **Figure 2.5**, a wireless-capable microcontroller is used at the heart of the kiosk to process check-ins and communication with the external database of user information.

In a typical interaction, the user approaches the kiosk, and a proximity sensor detects his or her presence and brings the kiosk out of low-power mode. The display on the kiosk then prompts the user to check in using their ID card. As an added form of redundancy, the smart kiosk also supports a keypad to enter the user's ID directly. For ease of use, the user's ID is encoded in a standard near-field communication (NFC) card, and the kiosk's NFC transceiver transmits the ID to the database for authentication. Upon successful authentication, the screen displays a message to indicate that the user's ID was accepted, in addition to LEDs and an audible indicator. Following an interaction with the user, the kiosk remains active for a short duration to seamlessly authenticate the next user during high-traffic intervals. After this, it returns to a light sleep mode and remains on standby until the next user approaches.

The SCAN team wanted to make the hardware for this project as cost-effective, power-efficient, and accessible as possible. For these reasons, all of the necessary hardware for the project is packaged inside the kiosk, which sits on a stand at the height of a typical user. Additionally, the kiosk has the option to be powered by either a standard wall outlet or an internal battery, enabling portability. Consideration for power consumption was taken when selecting core components such as the microcontroller, kiosk display, and card reader. For increased accessibility, SCAN can operate as a network of multiple kiosks that can be used to process check-ins in parallel. This required synchronization between multiple kiosk clients and a single server, which manages the user information database. Multiple kiosks enable the RWC and other organizations to eliminate queues, increase check-in throughput, and support multiple regulated entry points into the facility.

Stretch Goal: Solar Charging: In applications outside of our RWC use case, kiosks are often used for outdoor events such as sports games and concerts, where sunlight is readily available. Access to power outlets can be limited in these situations, so extending the battery runtime is important. In addition to reducing power consumption, introducing solar charging capabilities further enhances the utility of the SCAN system, increasing its portability and ease of use in outdoor environments. Furthermore, introducing solar charging enables the SCAN system to be a more environmentally friendly solution.

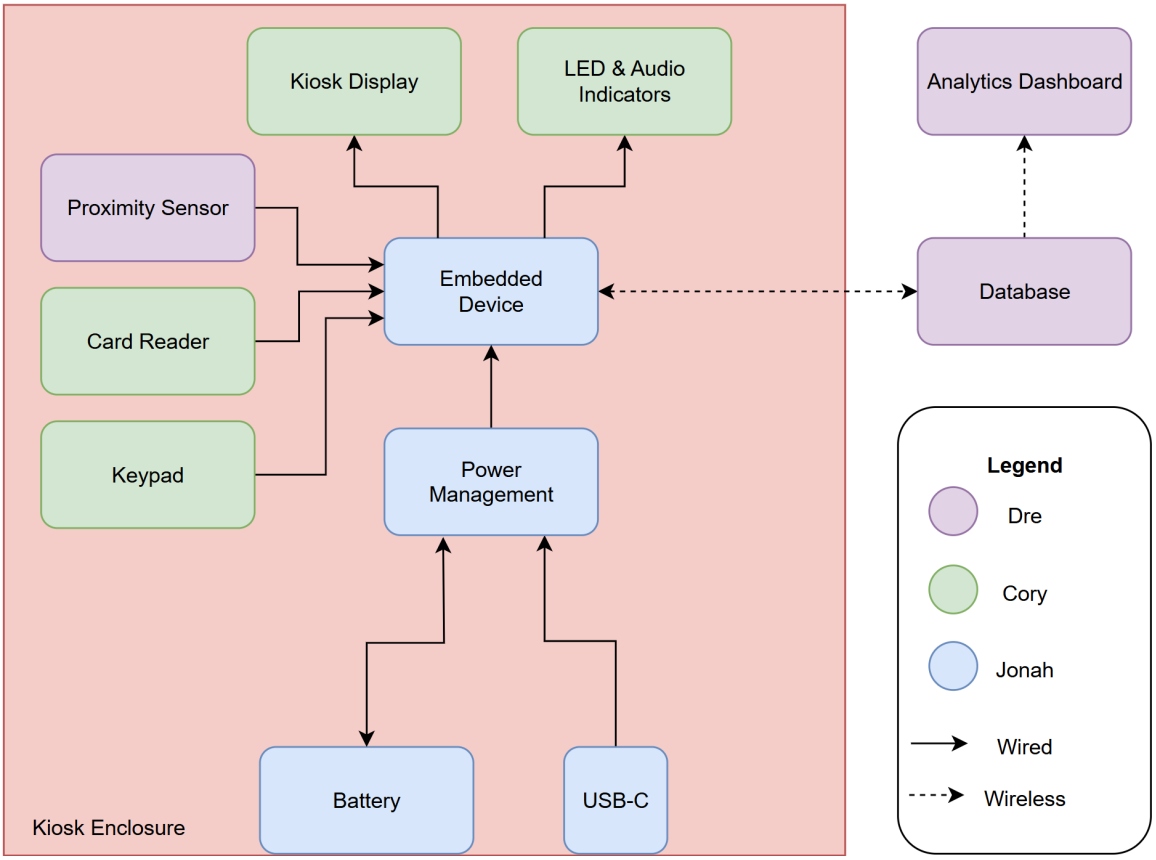


Figure 2.5 SCAN System Diagram

2.3.2 Software Objectives

SCAN's software can be split into four primary applications and objectives:

1. The SCAN GUI guides users through the check-in and check-out process.
2. The SCAN system's embedded application interfaces the GUI, LEDs, card reader, and proximity sensor to a microcontroller for processing and later transmission to a database over Wi-Fi.
3. SCAN utilizes a database to store centralized data between kiosks and web application instances.
4. The web application interfaces with the database to display occupancy analytics.

Figure 2.6 depicts the Kiosk GUI and embedded kiosk flowcharts, while Figure 2.7 shows the frontend and backend software flowcharts.

Kiosk GUI Objectives: SCAN's GUI is responsible for providing an intuitive, responsive, and efficient interface for interacting with SCAN's technologies. This interface needed to be responsive and user-friendly, displaying visual feedback to the user throughout each step of the NFC scanning process. It also needed to update in real time, showing messages such as "Access Granted," "Authentication Failed," or "Please Retry" immediately after a response has been returned from the backend server. Sprites and other graphics for the display were stored in the microcontroller's flash memory. By providing

visual feedback and a low-latency interface, our design satisfies the goal of having an easy-to-use and responsive GUI.

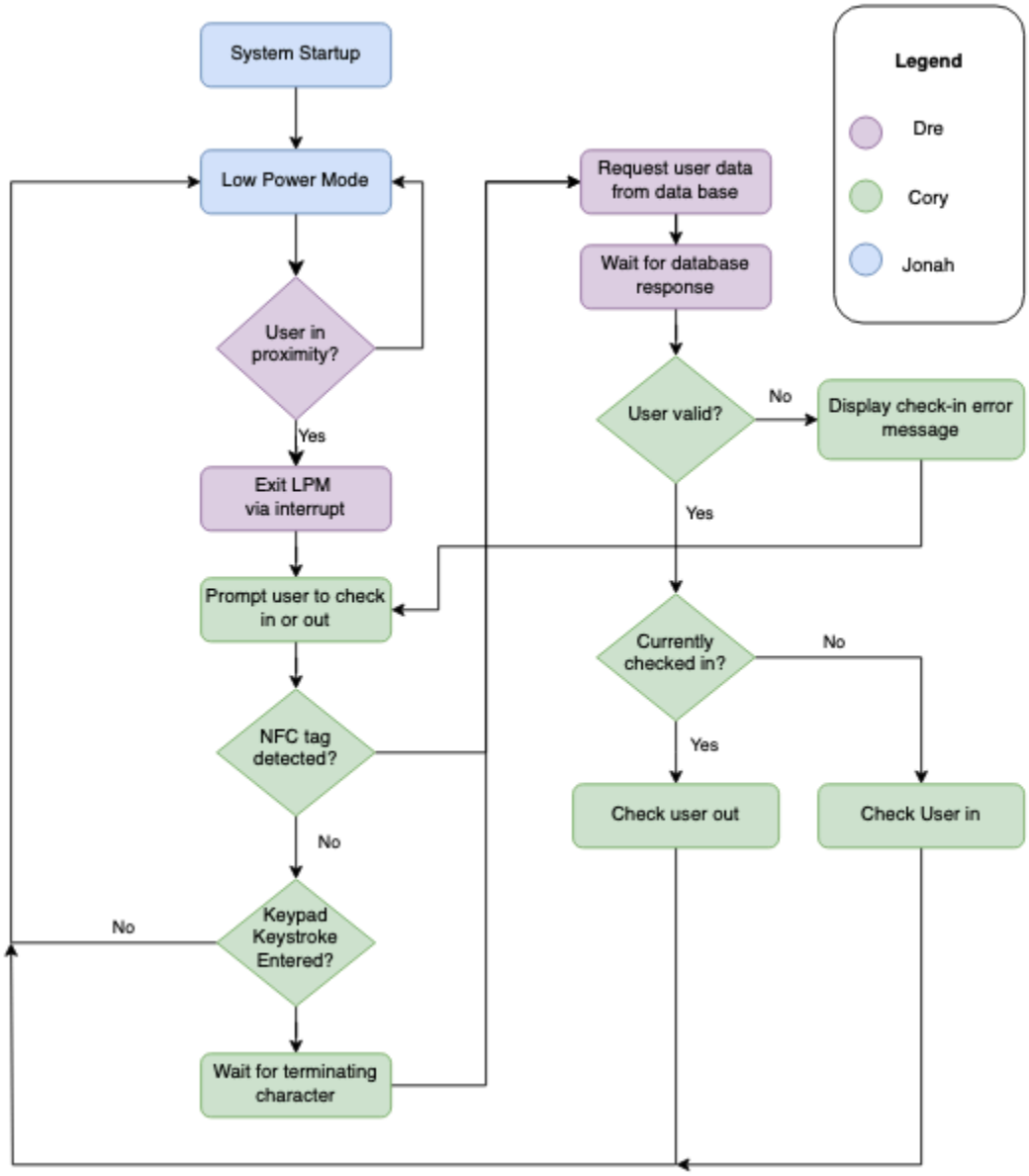


Figure 2.6 Embedded Software Flowchart

Embedded Kiosk Application: SCAN's embedded application's primary goal was to interface all of the hardware and software components using a central processor/microcontroller. To meet this goal, our first objective was for the embedded application to handle the NFC scanning process, ensuring each user's NFC tag is correctly identified and sent to the database for authentication and processing. Along with interfacing with the NFC scanner, our embedded application needed to be capable of interfacing with status LEDs and audio signaling devices. These devices serve as feedback indicators for successful completion of events like check-in, check-out, and

multifactor authentication. The MCU also needed to be capable of interfacing with the LCD and displaying messages and user data throughout each step of the check-in process. The interface needed to be quick to ensure the system was as responsive as possible. Finally, the embedded application needed to be optimized for power efficiency, especially in instances where it was running solely from battery power, such as outdoor events. This was aided by a low-power consumption motion sensor for detecting user presence, waking the system from low-power mode.

Database: A database facilitates communication between SCAN's kiosk and the web application. The backend database securely stores user profiles and gym information, including a user's name, ID, logs of check-in and check-out events and other necessary user information. Using this information, client-side edge nodes like the kiosk and web application are available to access user data for various purposes. The kiosk can use the database to access and update user check-in information and logs during kiosk interaction events. On the other hand, the front-end web application communicates with the database to request check-in/check-out logs and user data to display analytics to admin users and update user profiles.

Frontend Web Application: The web application displays up-to-date user statistics, allowing owners and building managers to monitor real-time occupancy and analyze user data. The following list describes the user data that the web application displays:

- Number of attendees currently checked in
- History of past attendance for a maximum interval of time
- Histogram of peak attendance hours for a given day and hour during the week
- The average duration of an attendee's stay

Stretch Goal: Over-the-Air Updates: SCAN enhances its ease of use and versatility by implementing over-the-air (OTA) updates, providing an efficient method for end-users to receive new updates and deploy specific functionality for certain use-cases. Over-the-air updates are a method of wirelessly updating the kiosk's firmware via a network. Although SCAN also implements reprogramming over USB-C for more efficient software development and testing, over-the-air updates allow end-users to quickly load new firmware onto their kiosk without needing any software development tools, cables, or additional hardware. Updates can be triggered by various events, such as an external I2C command, pre-configured keypad combination, or through the web app. These event-based triggers provide high flexibility for event administrators and technicians. In addition to this, OTA updates can be time-based for automated maintenance. The kiosk could periodically check for new firmware releases hosted on a GitHub repository or private server, ensuring the kiosk benefits from the latest firmware updates at all times. By combining event-driven and automated OTA update methods, this stretch goal reduces manual maintenance and allows SCAN to adapt to a wide range of applications, offering a user-friendly solution that aligns with the project's objectives and goals.

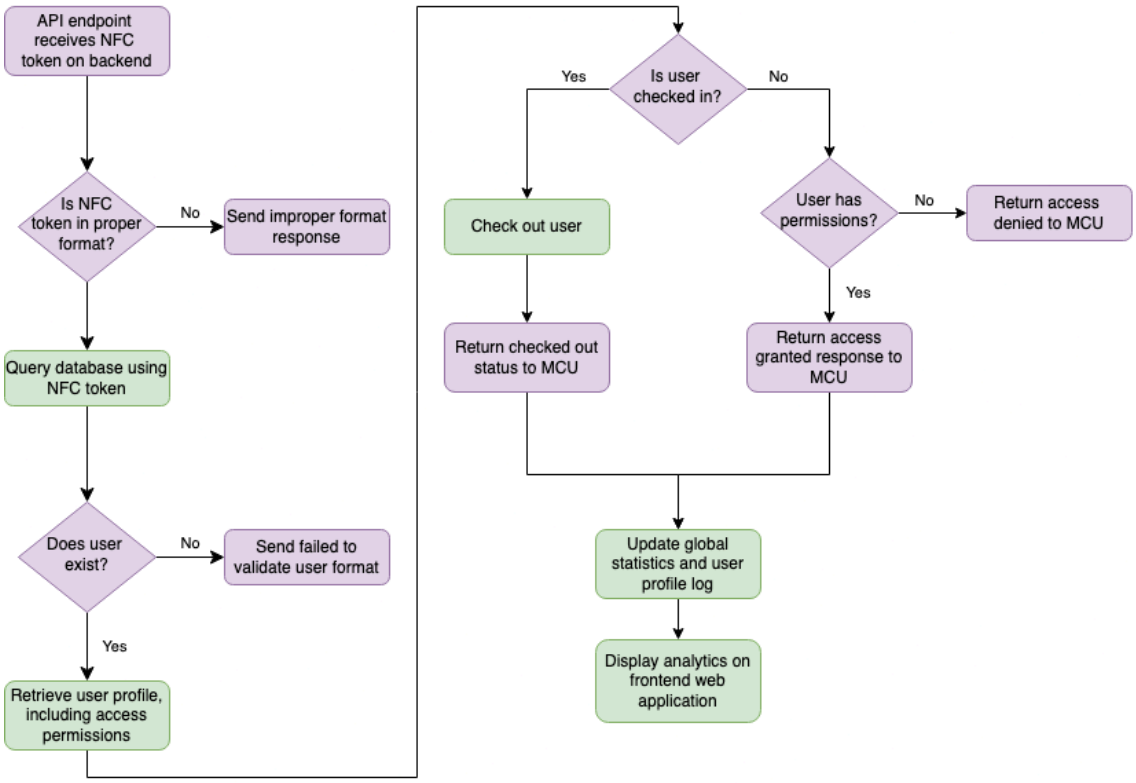


Figure 2.7 Frontend and Backend Software Flowchart

Stretch Goal: Lazy Loading: As a stretch goal, SCAN integrates lazy loading into its web application. Lazy loading is a feature in which applications retrieve and process data dynamically rather than preloading all information when the application starts. By implementing lazy loading, we can enhance the application's performance, reducing initial load times and conserving resources. This approach is particularly beneficial for large datasets or pages with extensive user interactions, as it minimizes unnecessary data transfers and optimizes user experience. To implement this feature, SCAN preloads only a small amount of data on each page. Subsequent blocks are requested from the database only when user input events like scrolling or results page cycling occur.

2.4 Requirement Specifications

In the previous sections, the goals, objectives, and motivations behind the SCAN project were described. To transition SCAN from a set of goals into a physical project, we defined a set of requirement specifications. These requirements were used to assess the functionality of the project and demonstrate a successful design. This set of requirement specifications is summarized in **Table 2.2**, and the requirements highlighted in yellow were used for the project demonstration to our peers and committee in Senior Design II.

Table 2.2 SCAN Requirement Specifications

Requirement	**Specification**
Maximum Enclosure Size	The maximum size of the enclosure and stand for each SCAN system is **1'x1'x4'**
Minimum number of authentication methods	SCAN supports **two methods** of authentication: NFC tags and manual entry of a user ID through a keypad.
Minimum Battery Runtime	SCAN has a minimum operating time of **2 hours** on battery power under typical operating conditions.
Minimum Charging Power	SCAN has a minimum supported charging power of **10W**.
Time to Authenticate	SCAN authenticates the credentials of a user ID in **less than 10 seconds**.
Authentication Accuracy	SCAN accurately authenticates users with an error rate of **5% or less.**
Number of Check-in Statistics	SCAN displays **at least three forms of analytics** based on check-in data, including occupancy, peak hours throughout the week, and the average duration of a user's stay.
Minimum Proximity Sensing Range	SCAN switches to active mode once a user is within **1.5 meters** of the system's range.
Analytics Refresh Rate	The analysis of SCAN's collected data is updated on the web page once **every minute**.

In addition to the Requirement Specifications table, the House of Quality diagram in **Figure 2.8** provides a thorough approach to relating customer requirements and engineering requirements. By identifying the correlation between each customer requirement and engineering requirement, the diagram helps to assess trade-offs and ensure alignment between the project's technical specifications and customer expectations. In this case, the target "customer" would be facilities such as the UCF Recreation and Wellness Center that require a check-in procedure and real-time analytics. In addition to these comparisons, the "roof" of the diagram compares each engineering requirement against each other, showing potential strengths and conflicts. All of these comparisons were continuously evaluated and used to guide the design process.

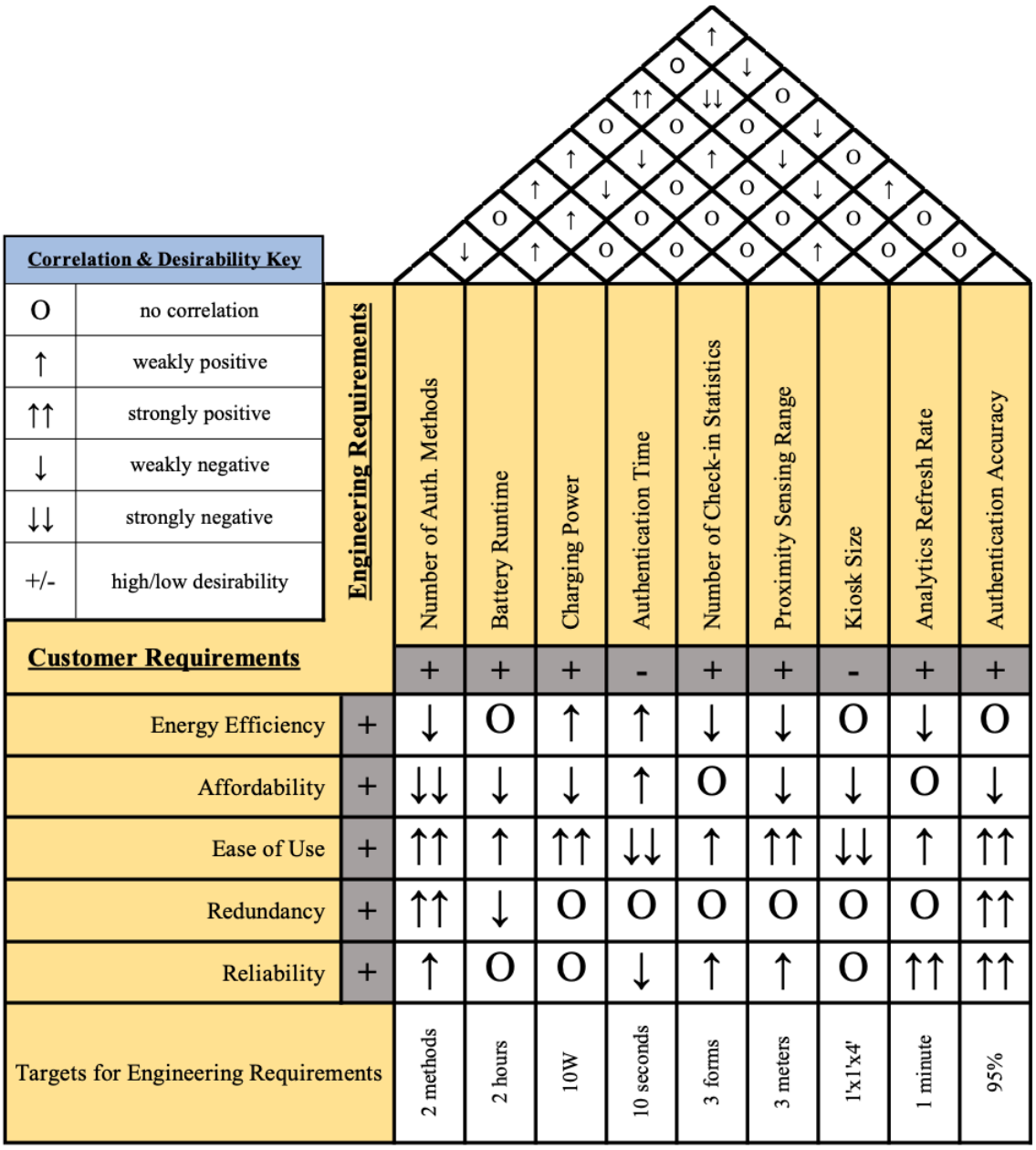


Figure 2.8 House of Quality Diagram

3. TECHNOLOGY RESEARCH

In this chapter, we investigate the technologies that informed the design and development of the SCAN system. This chapter conducts an in-depth study of existing hardware, software, and wireless communication protocols. Each section evaluates the strengths, limitations, and costs associated with various components to make informed decisions according to SCAN's requirements for scalability, reliability, and cost-effectiveness. This analysis provided us with a detailed foundation for the SCAN kiosk's design choices, ultimately guiding the selection of the best components and technologies for a low-cost, high-performance kiosk solution.

3.1 Processors

Processors are fundamental components in electronics, responsible for managing tasks, executing instructions every clock cycle, and doing complex mathematics. They are the "brains" behind electronic devices. Processors include microcontrollers (MCU), field programmable gate arrays (FPGA), system-on-chips (SoC), digital signal processors (DSP), and central processing units (CPU), each offering different advantages based on power draw, performance, versatility, and more. For example, some processors are suited for complex operating systems like Windows or Linux, while others excel in real-time embedded applications. Processors are necessary in everything from real-time control systems to high-performance computers. SCAN required a processor to handle the computing and transfer of data from user check-in, database fetches, proximity sensing, and analytics calculations. Since SCAN uses a cloud-based database, the selected processor also needed to handle wireless communication. This section compares the advantages of some processor technologies in order to make an informed decision for the processor to be used in SCAN.

3.1.1 Microcontroller (MCU)

A microcontroller (MCU) is an integrated circuit (IC) designed typically for embedded applications requiring general processing, sensor input, data handling, external device communication, and control systems. They are known for low power consumption, real-time operation, and cost-effectiveness, making them ideal for applications like home appliances, Internet of Things (IoT) devices, automotive control systems, various robotics projects, and more. Microcontrollers are geared towards running specific tasks repeatedly and are often used when results are needed within a certain amount of time (real-time operations). Although they are more limited in processing power than other technologies, they integrate many peripherals, including random-access memory (RAM) and general-purpose input/output (GPIO) pins, which can be programmed by software to do whatever is needed. Despite their comprehensive features, microcontrollers are relatively simple to implement due to their ability to be programmed in C or C++. Along with supportive development environments, robust embedded applications can be developed quite fast. Common MCUs include the STM32 series by STMicroelectronics and the MSP430 family by Texas Instruments.

MCUs typically incorporate built-in support for communication protocols like I^2C, SPI, CAN bus, and UART, which allow them to interface with sensors and other external embedded components. For example, I^2C is particularly effective for connecting multiple devices in a master-slave interaction on the same two-wire bus. The SPI (serial peripheral interface) protocol, on the other hand, excels at high-speed device communication. Support for CAN bus is also considered an important factor in selecting a microcontroller for automotive applications because most vehicles utilize multiple CAN buses to communicate between electronic control units (ECUs). Integrated support for low-level communication protocols, often with extensive libraries and example code, allows fast prototyping and design of embedded electronics.

Furthermore, MCUs often include peripherals like analog-to-digital converters (ADCs), pulse-width modulation (PWM), and timers. [4] This greatly simplifies the process of interfacing with external sensors and hardware. The ADC allows for the conversion of analog sensor signals from devices like temperature sensors or photodetectors, while PWM signal generators are necessary for motors, LEDs, and power converters. Combined with control algorithms programmed on the microcontroller, the MCU becomes a powerful device for controlling and managing complex embedded systems. These features are especially important for SCAN because of our need to integrate peripherals including NFC scanners and proximity sensors.

In terms of memory, MCUs integrate on-chip nonvolatile (usually flash memory) and volatile memory (RAM). The inclusion of flash memory means on-chip code and data storage, along with RAM for smooth instruction execution. This design facilitates quick responses to real-time events, which makes a microcontroller ideal for SCAN because of its ability to quickly respond to user check-ins.

Lastly, one of the exceptional features of microcontrollers is their low power consumption. Normally in the range of milliwatts or even microwatts [62], this characteristic makes them ideal for battery-operated applications, such as a portable kiosk. Along with their overall low power consumption, many support sleep modes, enabled in software, that reduce energy usage by disabling clocks and unused peripherals.

3.1.2 Field-Programmable Gate Array (FPGA)

Field-programmable gate arrays (FPGAs) are highly customizable processors that allow engineers to configure hardware functionality after manufacturing for their own application. Unlike microcontrollers, which have predefined hardware architectures (ARM, RISC-V, etc.), FPGAs consist of many reprogrammable logic blocks that can perform almost any digital function. The configuration can be reprogrammed to adjust to different applications or changing needs at any time. This makes them ideal for rapid prototyping and applications requiring high performance, parallel processing, and unique hardware functionality. They are even capable of emulating other integrated circuits.

While FPGAs do not have built-in support for communication protocols, they can be configured to handle almost all communication protocols, like UART, I^2C, ethernet, and more. For instance, integrating ethernet communication allows an FPGA to connect to the

internet for data transmission, making it suitable for high bandwidth networking applications. Compared to microcontrollers, the time to configure these protocols is longer, but the hardware flexibility of FPGAs allows them to handle higher bandwidth data if needed, depending on the specific configuration and design. The flexible capability of FPGAs is particularly useful in scenarios requiring large data throughput, such as video processing or complex signal processing. By utilizing parallel processing capabilities, FPGAs can execute multiple tasks simultaneously, moving lots of data at the same time and significantly enhancing system performance.

However, two disadvantages of FPGAs are their higher power consumption and higher cost. [5] Unlike MCUs, which can operate on only a few milliamps, FPGAs are considered power-hungry, typically using anywhere from one hundred milliamps up to tens of amps. This power usage varies significantly based on the tasks being performed. For example, high-resolution video processing is guaranteed to use more power than simple logic functions. Consequently, careful power management is essential when working with FPGAs. Since SCAN primarily runs off of battery power, this puts FPGAs at a strong disadvantage. Furthermore, FPGAs come at a higher cost upfront than MCUs. In our application, because cost is a consideration, this puts FPGAs at a disadvantage.

As opposed to MCUs, FPGAs do not normally have flash memory built-in. Instead, they include static RAM that can be mapped into distributed RAMs or block RAMs. [6] This flexibility allows FPGAs to manage larger datasets and perform more complex functions more efficiently than MCUs. They can also connect to external memory, allowing them to handle high-performance tasks that would be beyond what MCUs are capable of.

Lastly, programming FPGAs involves using hardware description languages (HDLs) like Verilog or VHDL, which is more complex than programming in C or C++. This presents a steeper learning curve for developers because it requires a solid understanding of digital logic and circuit design concepts. However, the ability to reconfigure and optimize hardware for specific tasks can result in significant performance gains and lead to lower long-term costs as processing needs change, making the choice to use an FPGA worthwhile in many circumstances.

3.1.3 System-on-Chip (SoC)

A system-on-chip integrates multiple components, including a CPU, memory, GPIO, and sometimes wireless communication modules like Wi-Fi and Bluetooth. [7] Being on a single chip, SoCs make versatile and powerful solutions compared to MCUs, with applications from smartphones to IoT devices and other embedded systems. The integration of multiple functions allows for reduced size, cost, and often power consumption, making them ideal for applications requiring a combination of requirements (ex. wireless communication, signal processing, and sensor interfacing).

SoCs typically support a wide variety of communication interfaces, including UART, I^2C, USB, Bluetooth, Wi-Fi, and more. These interfaces enable the SoC to handle a multitude of communication tasks and interact with multiple devices. Some examples of external

devices SoCs often interface with include cameras, Wi-Fi networks, display modules, and USB devices. Compared to microcontrollers, while they often share many of the same communication interfaces, SoCs typically refer to the integration of multiple devices not normally included on a microcontroller. For example, a microcontroller does not include a Wi-Fi antenna, unlike many SoCs.

Finally, SoCs are still relatively power efficient. Compared to FPGAs, they usually consume less power, operating in the range of milliamps to several amps. Though they are not necessarily known to be as power efficient as microcontrollers, many still feature sleep modes and other ways to pare down on power consuming functions. Battery-powered devices like smart sensors and smartphones rely on this efficiency for an extended battery life. Furthermore, SoCs can be brought up relatively fast due to their supported libraries for high-level programming, unlike FPGAs.

Table 3.1 Comparison of Processor Technologies

Feature	**Microcontrollers**	**FPGAs**	**SoCs**
Power Consumption (active)	0.05–10 mW	50–500 mW	10–500 mW
Clock Speed	1–200 MHz	50 MHz–500 MHz	80–240 MHz
Cost	$0.10–$10+	$10–$2000	$1–$10+
Typical Implementation Time	Days to Weeks	Weeks to Months	Days to Weeks
Implementation Complexity	Low (10–100 lines of code)	High (100-1000+ lines)	Low to Moderate (100–500 lines of code)
Wireless Capabilities	Limited	No	Yes (Bluetooth, WiFi, etc.)

After comparing the three technologies in Table 3.1, a system-on-chip (SoC) would be the best processor technology for our kiosk application due to its integration of multiple components, including a CPU, memory, GPIO, and wireless communication modules such as Wi-Fi and Bluetooth, all on a single chip. This consolidation minimizes the need for additional peripherals and simplifies the overall design, reducing the size, cost, and complexity. SoCs are versatile and powerful, supporting multiple communication interfaces like UART, I2C, and USB, which are essential for connecting to sensors, databases, and the NFC scanner in the kiosk. Additionally, the built-in Wi-Fi and Bluetooth functionality enables seamless wireless communication for database interactions and user check-ins. SoCs also offer both strong performance and power efficiency, making them ideal for battery-powered devices like SCAN. Lastly, the

availability of high-level programming libraries speeds up development, allowing for a more efficient and streamlined implementation compared to FPGAs.

3.1.4 Processor Part Comparison and Selection

Texas Instruments CC3235SF: The TI CC3235SF is a high-performance, dual-band Wi-Fi SoC designed for Internet of Things (IoT) applications. It features an ARM Cortex-M4 processor and supports 802.11 a/b/g/n Wi-Fi, along with security features like secure boot, encryption, and certificate-based authentication. [8] This makes it ideal for applications such as industrial automation, smart homes, and even medical devices. The integrated Wi-Fi supports station, access point, and Wi-Fi Direct modes, giving users flexibility in wireless communication options. As discussed earlier, SoCs feature both a processor and wireless communication modules, allowing the system to offload Wi-Fi tasks and provide more resources for other application tasks. Another significant feature of the CC3235SF is its support for low-power operation. With low-power deep sleep modes, it is well suited for battery-operated devices such as our battery-powered kiosk. In terms of memory, the chip includes 256 KB of RAM and 1 MB of on-chip flash memory. This makes it capable of storing relatively complex programs and ensuring fast processing times. With its versatile GPIO pins, ADCs, and other communication protocols (UART, SPI, I2C), the CC3235SF is suitable for applications requiring sensor integration and real-time control systems.

Espressif ESP32-C6-WROOM-1-N8: The Espressif ESP32-C6 is a low-cost, RISC-V SoC with integrated 2.4 GHz Wi-Fi and Bluetooth LE, making it a versatile choice for IoT applications. It is optimized for low-power operations and ideal for applications where energy efficiency is a priority, such as wearable devices, smart home appliances, and battery-powered sensors. Its integrated Bluetooth LE support also allows for easy pairing with smartphones and other Bluetooth devices, enhancing its usability in applications like fitness tracking systems. Similar to the TI CC3235SF, the ESP32-C6 offers secure boot and flash encryption, making it suitable for IoT solutions where data security is critical. [9] The chip also includes a number of GPIOs, ADCs, UART, SPI, I2C, and PWM, allowing it to interface with various external peripherals, such as sensors and actuators. Additionally, its low-power modes make it ideal for applications that require intermittent operation, such as the SCAN kiosk. The chip is popular with hobbyists due to its affordability, wireless communication features, and easy-to-implement libraries and example code.

Infineon CYW55912: The Infineon CYW55912 is a high-performance SoC designed for audio and IoT applications, with integrated dual-band Wi-Fi (2.4 GHz and 5 GHz) and Bluetooth 5.0. This chip is ideal for wireless audio streaming devices, including smart speakers and headphones. It features a more powerful ARM Cortex-M33, offering higher processing performance over the Texas Instruments and Espressif SoCs. [10] One of the features of the CYW55912 is its wireless coexistence capability, which allows it to handle both Wi-Fi and Bluetooth traffic at 2.4 GHz without interference. While this is a key requirement in systems utilizing both Wi-Fi and Bluetooth, since the kiosk only uses Wi-Fi, this is not necessarily beneficial to SCAN. Similar to the other SoCs, the chip also

offers a range of digital interfaces, including I2S, SPI, I2C, and UART, enabling seamless interactions with external devices. Unlike the ESP32, the CYW55912 is geared towards high-end consumer electronics, automotive infotainment systems, and wireless display applications.

Table 3.2 Comparison of SoCs

Feature	**Texas Instruments CC3235SF**	**Espressif ESP32-C6-WROOM-1-N8**	**Infineon CYW55912**
Processor Core	ARM Cortex-M4	RISC-V	ARM Cortex-M33
Clock Speed	80 MHz	160 MHz	192 MHz
Wi-Fi Support	802.11 a/b/g/n: 2.4 GHz and 5 GHz	802.11 b/g/n/ax: 2.4 GHz	802.11ax: 2.4 GHz and 5 GHz
Wi-Fi Data Rate	16 Mbps	150 Mbps	143 Mbps
RAM	256 KB SRAM	512 KB SRAM	768 KB SRAM
Flash Storage	1 MB	8 MB	None
Relevant Peripheral Interfaces	1x I2C 1x SPI 2x UART 4x ADC 27x GPIO	1x I2C 2x SPI 3x UART 1x ADC (7ch) 30x GPIO	2x I2C 2x SPI 4x UART 7x ADC 47x GPIO
IC Cost	$4.99	$3.13 (link)	$5.72 (link)
Development Board Cost	$65.99 (link)	$9.00 (link)	Not readily available

After comparing these three processors in Table 3.2, the ESP32-C6 is by far the best choice for our application due to its low cost, numerous features, strong maker community support, and fast programming capabilities. While the Infineon processor is a great choice for applications requiring complex signal processing and wireless capabilities, its higher performance comes at a higher cost and longer programming time, therefore making it not an ideal choice. Additionally, while the CC3235SF is a good choice overall, its high development board cost immediately eliminates this option. Together, all of these factors make the ESP32-C6-WROOM-1-N8 the best choice for the SCAN kiosk, where budget constraints and efficient programming are essential.

3.2 Proximity Sensors

For creating an efficient and low-power design, a triggering mechanism is necessary to transition our kiosk from low power mode to active status. This function is achieved using a proximity sensor. There are plenty of sensors employing various types of proximity sensing technologies, including infrared, ultrasonic, capacitive, LIDAR, radar-based, and hall effect-based sensors. Each of these sensing technologies presents unique features that make them well-suited for particular applications. In this section, we'll explore and compare three key proximity-sensing technologies, focusing on their distinctive characteristics in regard to how they meet the requirements of our project.

3.2.1 Ultrasonic Sensing Technologies

One of the proximity sensing technologies we compared for this project was ultrasonic sensing. An ultrasonic sensor is a sensor that measures the distance between the sensor and nearby objects using ultrasonic waves. [11] Ultrasonic pulses are emitted by a transducer and reflect off nearby objects, soon returning to the original emitter. Using the measured time difference between the emission and detection of the ultrasonic pulse, a simple calculation can be done to determine the distance traveled by the wave. This form of sensing is commonly used in proximity and distance measuring applications in various industries, including automotive, industrial automation, healthcare, consumer electronics, etc. For example, we see ultrasonic sensors commonly used in the automotive industry for applications like parking assistance, blind spot detection, and automated driving systems.

Interfacing with sensors employing ultrasonic sensing technology for distance measuring is fairly standard. Many ultrasonic sensors interface over typical communication protocols like I2C, SPI, or UART, and in many cases, simply via routing a single analog signal to a microcontroller's GPIO pin. These active sensors require a typical 3.3 V to 5 V input supply and are considered low power. It is important to consider IP ratings and temperature range when determining the ultrasonic sensor that best meets a project's needs, as each spec corresponds to the environmental conditions the sensor is most suited for.

Ultrasonic sensors offer strong technical advantages, making them well-suited for our application. They have a typical detection range between .1 and 10 meters, a response time within 50 ms, power consumption between 72 and 336 mW, small size, low cost, and mm range accuracy. [12]

Compared to other proximity sensing technologies, ultrasonics excel in their adversity to tough environmental conditions. They are able to detect objects in low light and dust/gas-filled environments with ease. Their operation is also unaffected by transparent surfaces, which is not the case with light-based sensing technologies.

3.2.2 Infrared Sensing Technologies

Infrared sensing technology is very common in proximity sensing. It is often used in sensors for security, home automation, and occupancy detection systems because of its low power consumption and aptitude for detecting body heat. Passive infrared sensors (PIRs) detect motion by measuring changes in infrared radiation emitted by objects in the sensor's detected field. Infrared sensing relies on pyroelectric transducers, which are responsible for converting changes in electromagnetic radiation to measurable electrical outputs. [13] The only power input required for implementing a sensor using this technology is for amplifier and analog to digital converter circuitry.

Interfacing with infrared-based sensors is comparable to interfacing with many other sensors. These sensors typically come with built-in ADCs and communication peripheral interfaces for data transmission via UART, SPI, or I2C. Simpler sensors implementing this technology may also output a single analog or digital signal corresponding to the detection of an object in the sensor's range.

Infrared-based sensors like PIRs boast a myriad of strong technical aspects that make them a good consideration for this project. A big factor is power consumption. PIRs consume only 1-3 mW (5 mA) of power in standby mode and 50-100 mW of power in active mode. PIRs also fit this project's range detection requirements with a range of 5-12 meters. They have a low response time of 0.1 to 1 second and are easy to interface with. [14]

3.2.3 Capacitive Sensing Technologies

Capacitive sensors operate by detecting changes in the capacitance between the sensor and its environment caused by humans or objects entering the sensor's detection field. When objects enter this field, they alter the dielectric constant, therefore changing the capacitance, which is then measured by the sensor. [16] Capacitive sensors have a wide range of use cases, from sensing touchscreen interaction to level sensing in industrial applications.

In terms of technical aspects, capacitive sensors have many qualities that make them a worthy consideration for this project. For one, capacitive sensors are quite power efficient, consuming only 50 uW to a few mW, meaning they are well suited for battery-powered systems. They also boast a low response time, typically within 10 ms, which makes them fast enough for touchscreen interfaces and certainly fast enough for proximity detection in our use case. With fine-tuning, the accuracy of these sensors can reach the centimeter range, which certainly meets the requirements of this project.

One notable aspect of capacitive sensors is that they are really only meant for short-range proximity sensing, in the range of a few millimeters to ten centimeters. Depending on our implementation, this may struggle to meet our needs.

Typical capacitive sensors can be simply interfaced using a microcontroller's GPIO pins, but more advanced sensors may offer UART, SPI, or I2C communication interfaces.

There are also many popular libraries for common processors like the ESP32 and STM32 that help with interfacing these sensors. The only real difficulties seem to come from the setup phase, where adjusting the sensor for environmental conditions to avoid false positives may introduce some complexity.

Table 3.3 Comparison of Proximity Sensing Technology

Feature	**Ultrasonic Sensor**	**Passive Infrared Sensor**	**Capacitive Sensor**
Detection Range	0.1 to 10 m	5 to 12 m	< 10 cm
Measurement Medium	Sound	Infrared radiation	Electric field (capacitance)
Response Time	5 to 50 ms	.1 to 1 second	< 10 ms
Power Consumption	72 mW to 336 mW	Standby: 1 to 3 mW Active: 50 to 100 mW	50 uW to 5 mW
Cost	$1 - $3	$1 - $5	$0.02 to $2
Ease of Integration	Easy	Easy	Moderate (requires careful tuning)
Environmental Sensitivity	Affected by temperature, humidity, and noise	Affected by heat sources	Affected by humidity, temperature, and nearby metals

Considering the factors discussed above and summarized in Table 3.3, we ultimately chose passive infrared sensing as our ideal sensing technology. Passive infrared sensing is the best option for meeting our range, power, and response time requirements. These types of sensors are also very commonly used in hobbyist projects, meaning there are ample numbers of example projects to use as a basis.

3.2.4 Proximity Sensor Selection

Knowing our desired proximity-sensing technology is passive infrared sensing, we chose to evaluate three of the most popular sensors on the market for hobbyist projects as these tend to have the most resources and are easiest to interface with. Those were the HC-SR501, AM312, and EKMB1107112, the important characteristics of which are detailed in Table 3.4.

Table 3.4 Comparison of PIR Sensors

Feature	HC-SR501	AM312	EKMB1107112
Operating Voltage	5 - 12 V	2.7 - 12 V	3 - 5 V
Detection Range	6m	3 - 5 m	.0.1 - 6 m
Power Consumption	65mA	0.1mA	3mW
Delay Time	0.3 - 5 min	2 seconds	1 - 25s
Size	32*24 mm	10*8 mm	Larger
Price	$10	$2	$8.70
Datasheet Link	[17]	[18]	[19]

Based on the information provided in the table, we ultimately selected to use the AM312 sensor as our choice passive infrared sensor. Some of the driving factors for this decision were this sensor's extremely low power consumption, response time, size, and price. It is also a very commonly used sensor with a development module and a very simplistic 3-pin interface.

3.3 User Authentication Methods

Authentication refers to the process of verifying that a user is who they claim to be. User authentication, therefore, is central to the design of the SCAN system. The challenge of user authentication must meet two key criteria. On one hand, the authentication process must be sufficiently challenging to prevent unauthorized users from impersonating an authorized user or gaining entry to a restricted location. On the other hand, it must be easy and straightforward enough for the authorized user to gain access. Therefore, the authentication method chosen for the SCAN system must meet a desired trade-off of security and ease of use. [15] Authentication methods fall into 3 categories:

1. Something that a user knows (e.g. PIN/password, answer to a security question)
2. Something that a user has (e.g. token, credit card, mobile device)
3. Something that a user is (e.g. biometric data: face ID, fingerprint, retinal scan)

To make the SCAN kiosk as accessible as possible, the kiosk supports authentication methods from two of the three categories listed above. This serves as an added redundancy in the event that one of the supported authentication methods fails due to a hardware or software issue, and it also provides the user with an option on how they would wish to be authenticated. The following section describes three different authentication methods from each of these categories: password authentication (something a user knows), hardware token-based authentication (something a user has), and fingerprint authentication (something a user is).

3.3.1 Password Authentication

One of the simplest and most common authentication methods is password authentication, where a user is prompted to enter a known password associated with their user ID to gain entry into a physical or digital location. This corresponds to the "something a user knows" category of authentication. Anytime a user is prompted to make an account online, they are required to input a password to protect their accounting information. When a user attempts to log in, the system will compare the entered password and username against the credentials stored in a database, authenticating the user if the entered credentials match. The most basic guidelines for creating a strong password include:

- Minimum of 8 characters
- At least one uppercase letter
- At least one number
- At least one special character

The strength of the password, or the amount of time it would take an unauthorized user to crack, is directly related to the complexity of the characters and symbols used. The longer and more varied the chosen password is, the stronger it will be against attackers. Assuming that a hacker is attempting to brute force guess a password with the minimum requirement of 8 characters listed above, with no prior information about the contents of the password, there would be approximately 6.1 quadrillion possible passwords.

Assuming an attacker can guess 1 billion passwords per second, it would take roughly 71 days to break in the worst case. In reality, this is an optimistic estimate. For one, a truly random password is strong from a security standpoint but hard to remember from a user standpoint. Users often default to using the same password across multiple services or using easy-to-guess phrases in their passwords, degrading the security. There are other methods of obtaining passwords beyond brute force attacks, such as phishing and data breaches. The security of passwords can be further enhanced by using a one-way cryptographic hash to store passwords inside of the database, such that even if an attacker gained access to the database, they would not be able to recover the passwords. Salting, a related process, can be used prior to hashing, which adds a random string of characters to a user's password to increase the complexity of brute-force attacks. [20]

In summary, passwords are an extremely common and straightforward approach to authentication users. They are affordable to implement and do not require any additional hardware and can be made quite secure against brute force attacks if randomization, hashing, and salting are used. However, security concerns can arise if weak or easily identifiable passwords are used.

3.3.2 Fingerprint Authentication

Fingerprint authentication utilizes the third category of authenticating data, "something that a user is." It uses a user's uniquely identifying fingerprint in order to verify their identity. Numerous technologies such as laptops, smartphones, and tablets have adopted

fingerprint scanners as a method of authentication. In studies conducted by the National Institute of Technologies and Standards (NIST), it was found that the best fingerprint recognition systems produced false negatives 1.9% of the time for single-finger identification, which drops to a false negative of 0.27% if two fingers are used. Most modern fingerprint scanners also repeatedly scan a fingerprint from different angles to obtain a more accurate sample, further reducing the false negative rate [21].

For fingerprint recognition to function, an optical signal is directed at the fingerprint from multiple angles to detect how the light is reflected by the ridges and valleys in the print. The collective depth map from these samples is used to construct the unique fingerprint signature of a user. After the initial capture period, a user's smartphone or other fingerprint recognition-compatible device will compare future input fingerprints to the baseline fingerprint for authentication. The data for each baseline fingerprint is stored in an encrypted numeric format, ensuring that the only way for an attack to gain access to the biometric data would be to break the underlying encryption.

Fingerprint authentication has many advantages over password-based authentication, including ease of use for authentication, convenience, difficulty to forge, and enhanced security. However, false negatives are possible, as mentioned previously, and NIST does not consider fingerprint recognition or other forms of biometric authentication to be reaching its standards for accuracy. To be a truly accurate standard, there can only be 1 false negative for every 100,000 scans, or a 0.00001% false negative rate. Furthermore, integrating fingerprint scanner technology into embedded applications can be quite expensive, with fingerprint scanners ranging in cost from $40 to $400, and the most accurate fingerprint scanners will tend towards the higher end of this range [22].

3.3.3 Hardware Authentication

Hardware authentication is a method of authenticating a user's identity with a physical device such as a tag or token to grant access to a physical or digital resource. It represents the "something you have" category. There are various types of hardware authentication devices. The two primary categories are hardware security tokens and smart cards. Hardware security tokens produce one-time passwords (OTPs) or cryptographic keys for authentication, and they can come in the form of USB tokens, Bluetooth tokens, or near-field communication (NFC) tokens. All of these tokens can be used to transfer the authentication password or key to a reader device for authentication.

Another type of hardware authentication is the smart card, which is a physical card with active circuitry that securely stores authentication data. These can come in the form of contact smart cards, which must be inserted into a reader; contactless smart cards, which employ RFID or NFC technology; and hybrid smart cards, which support both methods. Smart cards are frequently used to carry out financial transactions, and their tamper-resistant hardware provides high levels of security [23].

Hardware authentication tokens and smart cards offer a couple key advantages over other authentication methods such as passwords and biometric authentication. They provide a

high level of security as authentication is linked to a physical object, which is further enhanced if tamper-resistant hardware such as a smart card is used. However, like fingerprint readers, they require specialized hardware in order to read the information on hardware tokens or smart cards, limiting the accessibility of this authentication method in certain situations.

Table 3.5 Comparison of Authentication Methods

Feature	**Password**	**Fingerprint**	**Hardware**
Security	Vulnerable to attacks such as phishing and brute force	High security with unique physical traits; false negatives are possible	Very high security; resistant to phishing and remote attacks; requires physical possession
Ease of Use	Moderate; requires users to remember password phrase	High; users simply need to place a finger on the reader	High; users tap card, quickly complete the authentication process
Cost to Implement	Low; primarily software costs, or the cost of a physical keypad	High; requires biometric sensors which can be expensive to install	Moderate; depends on the hardware authentication method used (e.g. NFC, RFID, etc.)

Table 3.5 summarizes the three authentication methods based on their security, ease of use, and implementation costs. To provide the best blend of high security and ease of use, we selected hardware authentication as the primary authentication method for the SCAN system. Hardware tokens provide a high level of protection against phishing and remote attacks, as authentication is tied to the physical possession of the token. This ensures that only users with authorized access can gain entry. Additionally, hardware authentication offers a simple user experience, allowing quick and easy authentication with a simple tap of the token. As a secondary method, the SCAN system supports password authentication via a keypad located on the kiosk, which is a cost-effective and reliable backup in the event that the hardware token is unavailable or the token reader malfunctions.

3.3.4 Keypad Component Selection

To enable password-based authentication, the scan system must allow the user to manually enter his or her password via a physical keypad. Such a keypad must be responsive, provide enough key combinations to provide adequate security, and contribute to the aesthetic design of the SCAN system. Three potential keypad technologies were selected, each with roughly the same technical capabilities. The

primary difference between each keypad is the user experience, physical design, and cost. Table 3.6 compares the physical quality and capabilities of each keypad.

Table 3.6 Comparison of Keypads

Feature	Option 1	Option 2	Option 3
Name	Teyleten 4x4 Matrix Array	Oiyagai 4x4 Matrix Keypad	DIYmalls Matrix Membrane Keypad
Dimensions	68x20mm	43x38mm	65x63mm
Number of keys	16	16	16
Number of Pins	8	8	8
Tactility	Not tactile	Very tactile	Tactile
Cost	$1.60	$2.20	$7.50

With each keypad possessing similar technical capabilities, the component selection comes down to which keypad supported the aesthetic design of the SCAN kiosk while still providing responsive functionality and security. Option 1 provides a 4x4 membrane keypad with a flat profile. It is aesthetic with labeled buttons, but it is not very tactile and would require the mechanical construction of buttons or switches to increase tactility. Option 2 is very tactile, using physical push buttons as keys. However, the keys are not labeled and would once again require mechanical design efforts to integrate the keypad into the kiosk. In contrast, Option 3, the DIYmalls Matrix Membrane Keypad, is both aesthetic and tactile, with labeled buttons that can be physically pressed. It required the least effort to integrate into the SCAN kiosk and provides a cohesive look for the final design. While it is more expensive than other options, the labor costs saved in mechanical design justify the selection.

3.3.6 Smart Card Component Selection

When selecting a tag to represent user ID cards, the selected card must be low-cost (as it is being distributed to all users of the facility where the SCAN kiosk is deployed), able to store enough data to authenticate users, and passively powered. Three cards have been identified that fit this description.

The first option, the NTAG215 series of smart NFC cards, is compatible with a wide range of NFC devices and is often used in hobbyist applications like Amiibo cloning or data storage. As seen in Table 3.7, they offer around half a kilobyte of memory and are compatible with most mobile devices, but their security features are limited, and they are often used for low-security applications. Similarly, the NTAG213 sticker tag is widely used for basic low-memory, low-security tasks such as URL linking or WiFi password sharing.

Table 3.7 Comparison of Smart Cards

Feature	Option 1	Option 2	Option 3
Name	NFC Card	Mifare Classic 1K RFID Smart Card	NFC Sticker
IC	NTAG215	NXP Mifare Classic	NTAG213
Standards	ISO/IEC 14443A and 18000-3	ISO/IEC 14443A	ISO/IEC 14443A and 18000-3
Capacity	540 Bytes	1044 Bytes	144 Bytes
Write Endurance	100,000 Writes	100,000 Writes	100,000 Writes
Size	54*85.5*0.8mm	54*85.5*0.8mm	2.54cm (round)
Price	$10/20 cards	$8/10 cards	$15/50 tags

Meanwhile, the Mifare Classic 1K card has double the capacity of the NTAG215 and stronger security features. Its memory can be partitioned into sectors, where each sector has a different key, enhancing security across multiple applications. The card is compatible with the widely-used ISO 144443A standard and is able to be used with many popular RFID transceivers. **An important note**: Unlike the NTAG215 and NTAG213, which are both directly based on the NFC forum specifications, Mifare cards are a proprietary RFID technology produced by NXP Semiconductors, which has some distinct differences compared to NFC. However, Mifare smart cards are fully compatible with NFC readers and the NDEF format through a mapping model. NXP provides technical documentation on this mapping process and designates the Mifare card as an “NFC Type Mifare Tag”. [65] Furthermore, numerous NFC libraries for popular embedded systems frameworks such as Arduino and ESP32 are compatible with the Mifare series cards. These reasons make the Mifare Classic 1K card the ideal candidate for SCAN’s hardware authentication system.

3.4 Long-Range Communication Technologies

The SCAN project consists of both an embedded system and a remote database that need to be connected by means other than a direct wired connection to facilitate transmission of check-in data. This presents the need for wireless communication technology. There are a multitude of wireless communication technologies available today, each with its own strengths and weaknesses. Below we explore a few of those technologies: Wi-Fi, LoRa (long-range radio), and LTE-M (LTE for Machines).

3.4.1 WiFi

WiFi is one of the most renowned wireless communication protocols, used in various applications from connecting our personal home devices to the internet to aiding in small

hobbyist projects. It is extremely well suited for high bandwidth transmission in local areas, communicating via 2.4 GHz and 5 GHz frequency bands. WiFis wide adoption, even in fairly cheap hobbyist microcontrollers like the ESP32, makes it a strong candidate for interconnecting our kiosk to a backend server.

WiFi offers up to 100 meters of range indoors and 300 meters outdoors, making it well-suited for our intended environment, the UCF RWC. Depending on the specific standard implemented, WiFi can support bandwidths up to a few hundred Mbps. Typical microcontrollers for projects of this scale, like the ESP32, support 802.11b/g/n standard, offering speeds up to 150 Mbps. In terms of power consumption, WiFi is known to consume relatively more power in comparison to other wireless communication standards, which is a notable consideration if the kiosk is being operated off of battery power. Finally, WiFi offers strong latency metrics, meaning communication between the kiosk and database won't suffer from extensive delays. [24]

3.4.2 Long-Range Radio (LoRa)

Long-range radio or LoRa, is a low-power, wide-area network technology designed for very long-range, low-bandwidth communication. It is typically used in projects where long-range, low bandwidth and power-efficient communication are priorities, like in smart agriculture and smart city applications.

LoRa offers ranges of up to 15 km in rural areas and 2-5 km in urban areas. This may be applicable in the case that a kiosk is placed far away from the backend server access point. LoRa is also renowned for its ultra-low power consumption. LoRa communication has the option to remain in low-power mode where operation is effectively halted until the next outgoing transmission. This is definitely a strong consideration point for remote applications. One of LoRa's restrictions is its bandwidth. Another is that it can only support up to around 50 kbps speeds, meaning rapid or large-sized data transmission may not be well suited for this technology. This transmission rate doesn't seem to be an issue in the context of SCAN, as very low amounts of data is transmitted between the kiosk and database. Another downside to LoRa is its relatively high latency. High range and lower power consumption comes with the tradeoff of high latency. This may be a restrictive factor when it comes to SCAN, limiting response time, which would not help us meet our requirements.

Implementation of LoRa communication requires purchasing LoRa transmitter and receiver modules and an antenna. These are then interfaced to the MCU and power system, as with all other components of this project. On the software side, of course, relevant libraries for interfacing the receiver and transmitter modules are required. [25]

3.4.3 LTE for Machines (LTE-M)

LTE for Machines, or LTE-M, is a cellular network technology that is efficient in long-range communication for remote devices targeted towards IOT devices. It is often used in remote tracking and monitoring systems. LTE-M, similar to LoRa, is especially

useful in remote locations where access to WiFi networks is scarce. LTE-M operates as a part of the 4G LTE family and enables connection from edge node devices to other devices via cellular towers.

LTE-M has strong technical aspects, including its long-range and low power consumption, but it suffers from flaws similar to LoRa. LTE-M has a range that spans potentially multiple kilometers depending on the density of cellular towers in a given area. This makes LTE-M an excellent option for projects where a stable network connection can't be easily established, whether because of mobility or lack thereof. LTE-M offers a substantially larger bandwidth compared to LoRa at 1Mbps but still does not compare to that of WiFi. Similar to LoRa, LTE-M offers low-power modes that contribute to battery life extension in remote environments. Latency is not on par with WiFi, but it is still much lower than with LoRa. Ultimately LTE-M's features make it a considerable choice for long-range communication technologies.

Integration of LTE-M into the SCAN system is more complex than either WiFi or Lora. Hardware-wise, this would require an LTM module, antenna, and SIM card. Regarding software, compatible software libraries would be needed alongside a cellular plan for usage. The initial setup involves setting up a SIM card and data plan, which adds a lot of complexity to the initial complexity. Still, afterward, there isn't much more complexity than something like LoRa due to the prebuilt infrastructure for cellular networks like cell towers. [26]

3.4.4 Long-Range Communication Technology Selection

Table 3.8 Comparison of Long-Range Communication Technologies

Feature	**WiFi**	**LoRa**	**LTE-M**
Range	100 meters indoors, 300 outdoors	15km rural, 5km urban	10-15 km urban
Data Rate	Up to 1 Gbps	Up to 50 Kbps	Up to 375 Kbps
Power Consumption	100-200 mW	10-50 mW	50-150 mW
Latency	1-10 ms	200-1000 ms	10-50 ms
Cost	$5-20/module	$2-10/module	$10-50/module
Complexity	Low	Moderate	Moderate
Bandwidth	Up to 160 MHz	125-500 kHz	1.4 MHz

Comparing these three different long-range wireless transmission technologies in Table 3.8, we ultimately settled on WiFi as our primary wireless communication technology for

this project because of its high data rate and bandwidth, low power consumption, and, most importantly, ease of implementation. Many microcontrollers now come with built-in WiFi capabilities and easy-to-use libraries and examples that we plan to take advantage of. Although WiFi doesn't offer nearly as large a range as the other two technologies, 100 meters of indoor range was more than enough for this project.

3.5 Short-Range Wireless Communication Technologies

To support the contactless hardware authentication of a user, the SCAN system utilizes a short-range wireless communication interface capable of transferring users' uniquely identifying information to the kiosk. It is necessary to use a technology that is easy to use, supports quick set-up times, and consumes low power. Three communication technologies have been identified—Radio Frequency Identification (RFID), Near-Field Communication (NFC), and Infrared (IR) Communication—and the following section describes the trade-offs of each technology.

3.5.1 Radio Frequency Identification

Radio Frequency Identification, or RFID, is a wireless technology that enables data exchange between two devices using radio waves. It can be used to identify and track objects or individuals and operates between an RFID tag and an RFID reader. There are three different forms of RFID tags:

1. *Passive RFID* tags are not internally powered and rely on the RFID reader to transmit the necessary power for the tag data to be read.
2. *Active RFID* tags contain an internal battery and are typically more reliable and able to transmit for longer ranges.
3. *Semi-passive RFID* tags contain an internal battery but rely on the signal provided by the reader to enable transmission.

The RFID reader is a transceiver used to retrieve data on an RFID tag using radio waves, which can then be processed by an external system. RFID readers can be stationary, in the case of a kiosk, or mobile, in the case of a handheld scanner or a phone. The amount of data that a single RFID tag can store ranges from a single serial number to numerous pages of data. Under the umbrella of RFID technology, there are four different RFID classifications, corresponding to different frequency ranges. *Low-frequency RFID* operates in the 30-300 KHz frequency range and operates over short distances (up to 10 cm). It is typically used for anti-theft systems, building access, and microchipping. *High-frequency RFID*, which has a slightly longer range of up to 1 meter, operates in the 3-30 MHz frequency range and is used for contactless payment and smart cards. *Ultra-high-frequency RFID* can transmit even longer ranges, up to 12 meters or more and boasts even faster data rates at a frequency band of 300 MHz to 3 GHz. It is used in toll collection (such as E-ZPass) and supply chain management tracking. Finally, *microwave RFID*, operating in the 2.45 GHz frequency band, can extend to over 10 meters and provides extremely high-speed data transmission. It is more expensive to implement and is used in high-end toll collection systems. [27]

3.5.2 Near-Field Communication

Near-field communication, or NFC, is a subset of RFID communication optimized for ultra-short-range communication (less than four centimeters) and supporting a low latency setup, connection, and transfer. Like RFID, only the reader needs to be actively powered to exchange data, enabling an active reader and passive sender configuration. This is enabled by the transmission of power at a maximum rate of 1W to the passive device. It operates in the 13.56 MHz frequency band and has a transfer rate between 46 kbit/s up to 1.7Mbp/s. During setup, NFC devices must establish the connection and power requirements of the target device. If the reading device targets a powerless device, it first must power up the remote device. After reading the data, it must be able to disconnect, and all of the following steps occur within the duration of a single tap. NFC is designed to be inherently user-friendly, where the use case is centered around a simple tap of a smartphone or NFC-compatible tag.

NFC-enabled devices have four distinct modes of operation. In card emulation mode, the NFC device operates as a contactless card able to communicate with a contactless reader device (e.g. the emulation of banking or public transit cards). In reader/writer mode, the NFC device acts as a contactless reader in communication with tags or other contactless devices. It allows the device to read tags that include web addresses or contact information. In peer-to-peer mode, two active NFC devices are tapped to exchange data. In wireless charging mode, the active NFC device can be used for the dedicated transfer of a maximum of 1W of power over an NFC connection. Wireless charging mode can charge small devices such as a stylus, headset, or smartwatch. [43]

Primarily used in Reader/Writer mode, NFC tags are contactless, passive cards capable of storing data in the NFC Data Exchange Format (NDEF), a special format for a data payload defined by an NFC Forum Specification. This specification will be explored in more depth in Chapter 4: Standards and Design Constraints.

3.5.3 Infrared Communication

Infrared communication is a wireless technology that utilizes infrared light to transmit data across short distances. It requires a direct line of sight between the transmitting and receiving devices and operates in the spectrum of light between 750 nm to 1 mm wavelengths. In IR communication, the IR transmitter will rapidly turn on and off at a typical rate of 38 KHz. [51] The intervals and pulse distance of IR communication depend on the standard being implemented. One such standard for IR communication is overseen and managed by the Infrared Data Association (IrDA). IrDA specifies a range from one to three meters, uses line-of-sight communication, and has data transmission rates from 9600 bits per second up to 16 megabits per second. IrDa also supports a low bit error rate and secure transfer of data.

Consumer industries including electronics, automobiles, medical devices, and household appliances use IrDa for its speed, security, and cost efficiency in transferring data. Remote control systems commonly use IR communication, as it provides a simple and cost-effective way of communicating and transferring data between televisions, air

conditioning units, and audio systems. It can also be used for motion detection in security systems and for capturing heat signatures in cameras that use thermal imaging. Prior to the introduction of Bluetooth and WiFi, it was also commonly used in devices for short-range data transfer.

While IR communication provides many advantages as a short-range wireless communication protocol, it also comes with several trade-offs. Notably, it requires a line of sight for the IR signal to pass between transmitter and receiver, making IR communication less reliable in the event of obstructed environments or situations. Furthermore, it has a lower data rate than many other communication protocols such as WiFi and Bluetooth.

Table 3.9 Comparison of Close-Range Communication Technologies

Feature	**RFID**	**NFC**	**IR Communication**
Set-Up Time	< 0.1ms	< 0.1ms	~0.5s
Range	Up to 3m	Up to 10cm	Up to 5m
Usability	Item-centric, Easy	Human-centric, Easy, tap to connect	Data-centric, Easy
Connectivity	Sender/receiver	Tag/reader	Line of sight
Power	One device can be passive, very low power consumption with passive tags	One device can be passive, very low power consumption	Both devices must be active, least energy efficient
Use cases	Item tracking	Pay, get access, share, initiate service	Control and exchange data

For the SCAN system, the primary factors used to select a short-range communication interface include set-up time, range, usability, connectivity, and power consumption. Table 3.9 compares the different communication interfaces against these criteria. Under usability, NFC is defined to be human-centric, as it is designed to be as easy for users to use as possible. RFID is item-centric for its ability to track and locate items. Meanwhile, IR communication is data-centric, as they are designed with data exchange in mind. Using these evaluation criteria, we select NFC for the SCAN system. NFC was designed with ease of use in mind with its effortless tap-to-transfer feature. Furthermore, it is more power efficient than IR communication for transferring small packets of data over ultra-short distances. Today, it is a widely used and well-defined technology standard used in smartphones, payment platforms, and identification. It is ideal for transferring authentication data from a passive tag, or ID, to a stationary kiosk with a short set-up time, all taking place within a single tap.

3.5.5 NFC Reader Component Selection

To support hardware token authentication via the SCAN system, two NFC technologies are required: a tag and a reader. The NFC reader must follow the NFC standard, respond quickly to tags within proximity, and interface with the microcontroller inside the kiosk.

PN532: The PN532 by NXP is a versatile NFC transceiver that supports multiple communication modes, including reader/writer, card emulation, and peer-to-peer modes. It is compliant with standards like ISO/IEC 14443A/B and ISO/IEC 18092, which enable communication with a wide range of NFC and RFID tags. The PN532 supports data transfer rates of up to 424 kbit/s and is capable of handling Mifare Classic and FeliCa smart cards. Its low-power modes make it suitable for battery-powered applications. Furthermore, the PN532 offers various host interfaces, including SPI, I2C, and UART, allowing it to integrate easily with different systems. [30]

ST25R3911B: The ST25R3911B by STMicroelectronics is a high-performance NFC reader IC designed for applications such as access control, payment, and ticketing. It supports multiple NFC standards, including ISO/IEC 14443A/B, ISO 15693, and FeliCa. One of its standout features is the automatic antenna tuning system, which optimizes performance for applications with directly driven antennas. To support applications requiring high data rates, the ST25R3911B has a very high bit rate (VHBR) mode, where it can receive and write data at rates up to 6.7 mbit/s. Additionally, it offers advanced power management capabilities through low-power card detection modes using capacitive sensors. [31]

CLRC663: The CLRC663, also by NXP, is a multi-protocol NFC frontend IC that supports several NFC and RFID standards, such as ISO/IEC 14443A/B, ISO/IEC 15693, and FeliCa. It can communicate at high speeds (up to 848 kbit/s) and features a secure access module (SAM) interface for high-security applications. The CLRC663 is optimized for reading and writing to a variety of Mifare products, making it suitable for industrial, gaming, and access control applications. Its design includes flexible power-saving modes, which make it a good option for low-power systems. It also offers a wide operating temperature range, making it adaptable to various environments. [32]

Ultimately, the PN532 was chosen for the SCAN kiosk due to its compatibility with the desired NFC standards (such as ISO/IEC 14443A/B and Mifare), its ease of integration using various host interfaces, and its support for reader/writer mode. Its support of Mifare Classic cards is a major contributing factor, as we are using Mifare cards as our smart card of choice. The PN532's ability to operate efficiently in low-power modes, combined with its proven reliability in handling secure transactions made it the optimal choice for a low-cost, reliable, and secure system like the SCAN kiosk. Additionally, its wide availability and strong documentation contribute to the decision. Numerous low-cost development boards with integrated PN532 ICs are available, ensuring smooth development and deployment for the project.

Table 3.10 Comparison of NFC Reader ICs

Feature	Option 1	Option 2	Option 3
Model	PN532	ST25R3911B	CLRC663
Supported Standards	ISO/IEC 14443 Type A and B, Mifare, FeliCa, ISO/IEC 18092	ISO 18092, ISO 14443 Type A and B, ISO 15693	ISO/IEC 14443 A and B, ISO/IEC 18092, ISO/IEC15693, ISO/IEC 18000-3 mode 3, Felica, Mifare, NFCIP-1, EMVCo
Communication Modes	Reader/Writer, Peer-to-Peer, Card Emulation	Reader/Writer Peer-to-Peer	Reader/Writer Peer-to-Peer
Max Data Rate (kbit/s)	424	6800	848
Host Interfaces	I2C, SPI, UART	SPI	I2C, SPI, UART
Max Power Output (mW)	820	1400	1125
Supported Range (cm)	7	10	12
Operating Temperature	-25 to 85 °C	-40 to 125 °C	-25 to 85 °C
Cost per IC	$7.14	$6.48	$7.53
Datasheet	[67]	[68]	[69]

3.6 Display Technologies

To provide a responsive user interface as users complete the check-in process, the SCAN system requires a form of display technology with low power consumption, responsiveness, affordability, programmability, and image quality. In the following section, we compare three different display technologies: liquid crystal displays (LCDs), thin-film transistor (TFT) displays, and organic light-emitting diode (OLED) displays.

3.6.1 Liquid Crystal Display (LCD)

Liquid Crystal Displays, or LCDs, are a common display technology used to display graphics by applying a voltage to a liquid crystal layer. Illustrated in Figure 3.1, the core components of an LCD are an LED backlight to provide illumination, two glass plates with a liquid crystal layer in between, and a polarizing filter on either side of the crystal layer. When a voltage is applied to the liquid crystal layer, the crystal's structural properties are altered. Without an applied voltage, the molecules in the liquid crystal are twisted, allowing the backlight to shine through to the glass screen. When a voltage is applied to the electrodes, the molecules are untwisted, which prevents the light from reaching the screen. Each pixel in an LCD contains red, green, and blue subpixels. By controlling the voltage applied to each subpixel, each pixel can produce a wide array of colors. [33] The most common type of LCD, the passive-matrix LCD, is commonly used in watches, calculators, clocks, and other embedded devices for its low cost and power consumption compared to other display technologies. [34]

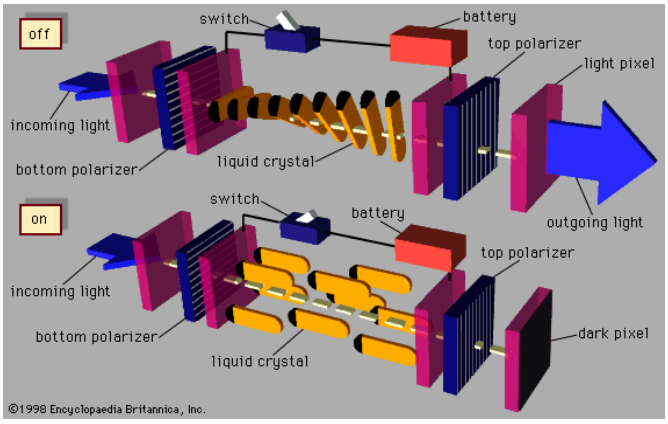


Figure 3.1 LCD Layer Composition (Obtained from Britannica)

3.6.2 Thin-Film Transistor (TFT) Display

The thin-film transistor display is a special type of LCD that uses an active-matrix configuration. An active matrix display means that each pixel is individually controlled, making it more responsive and of higher image quality compared to a passive-matrix LCD. Similar to LCDs, TFT displays require backlight, glass, and polarizing layers, as seen in **Figure 3.2**. The display composition differs in the inclusion of an array of thin-film transistors, which are used to individually address each pixel and maintain the pixel state between refreshes to the display, leading to smoother images. TFT displays enable much higher resolutions and can be used to address thousands or even millions of pixels. TFT displays are the most popular type of color display for devices requiring high-quality visuals, and they are frequently used in computer monitors and televisions. [35]

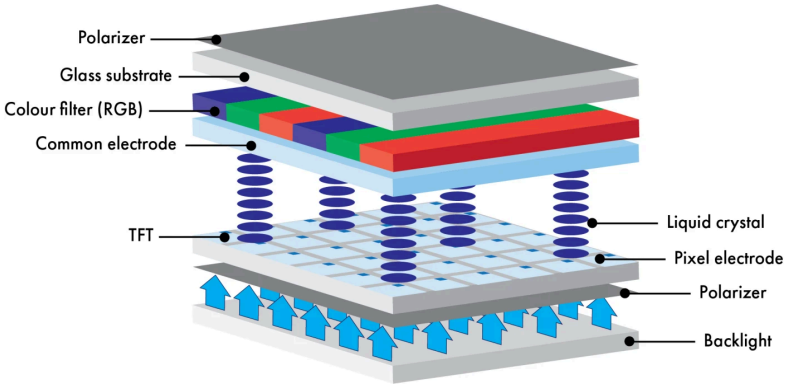


Figure 3.2 TFT Display Composition (Obtained from RS DesignSpark)

3.6.3 Organic Light-Emitting Diode (OLED) Display

Organic light-emitting diode, or OLED, displays are a display technology that, unlike LCDs, do not require a backlight to display pixels. Instead, the organic LED materials inside of an OLED display emit light directly when an electric current passes through. Compared to LCDs, OLED displays are more energy efficient, less complicated to produce, and thinner. They produce a wide range of colors with a sharp contrast since OLED displays don't use a backlight. The primary technology enabling OLED displays is the OLED emitter, which is a carbon-based (organic) material that emits light when an electric current is applied. This emissive OLED layer is sandwiched between a cathode and anode layer. Figure 3.3 provides a more in-depth view of the composition of an OLED display.

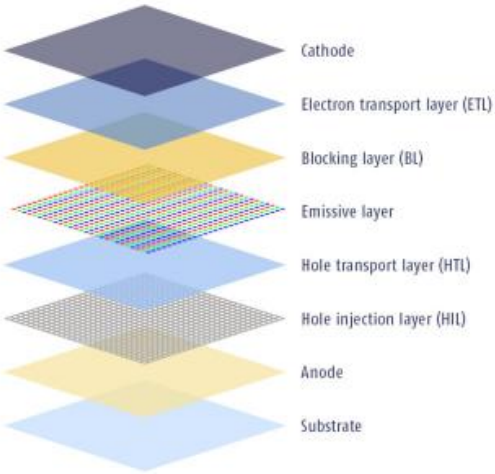


Figure 3.3 OLED Display Composition (Obtained from OLED Devices)

There are two primary types of OLED displays:

1. Passive-Matrix OLED (PMOLED): These displays are simpler, cheaper, and easier to manufacture. PMOLED displays are typically quite small in size and resolution, and they are primarily used with embedded systems, wearables, and other small devices to provide a graphic display.

2. Active-Matrix OLED (AMOLED): These displays use an active-matrix array of thin-film transistors; as a result, they can scale in size and resolutions much better than PMOLED displays. AMOLED displays are used in laptops, TVs, and smartphones that require higher image quality. [36]

Table 3.11 Comparison of Display Technologies

Feature	**LCD**	**TFT LCD**	**OLED**
Pixel control	Passive matrix	Active matrix	Active matrix
Image quality	Good color	High resolution	Sharp contrast
Response time	Slower (~10ms)	Fast	Extremely fast (~1us)
Power consumption	Low	Low, but higher than standard LCD	Low for dark content but higher for bright
Backlight	Required	Required	Not required
Thickness	Thin	Thin	Thinnest
Durability	Durable	Durable	Sensitive to moisture
Cost	Lowest cost	Low cost	Higher cost

After comparing the different display technologies and summarizing the results in Table 3.11, the thin-film transistor (TFT) display emerged as the most favorable display choice for the SCAN kiosk. While OLED displays provide superior contrast and energy efficiency in certain scenarios, their higher cost and sensitivity to environmental conditions such as moisture make them less suitable for a low-cost, durable kiosk system. In contrast, TFT displays hold a clear advantage over LCDs in terms of image quality, responsiveness, and degree of pixel control due to their active matrix configuration. This allowed for smoother visuals and faster UI interactions. With a balance of affordability, durability, and performance, the TFT display was the optimal choice for the SCAN system.

3.6.4 TFT Component Selection

When selecting a TFT display for the SCAN kiosk, responsiveness and user-friendliness are the most important factors. As in the design philosophy behind NFC, the goal of the SCAN kiosk is to make check-ins as effortless as possible and as simple as a single tap, followed by feedback from the display. Users should be able to clearly understand and receive feedback at all stages of the check-in progress, but they do not need to navigate through complex menus or interact directly with the display through a touchscreen. This

section compares three different TFT displays to find the display most suited for the SCAN system.

HiLetgo 3.5" TFT LCD Display: This display from HiLetgo is a 3.5”, 480x320 resolution display designed to interface with the Arduino Mega and similar microcontrollers. Of the displays considered, this display has the largest form factor and the highest resolution. The size of the entire display module is 96.6mm by 60.3mm. With 36 total pins, it is designed to plug directly into the Arduino Mega 2560 as a shield, but it can be interfaced with other microcontrollers with the proper connections. This display can be driven by either a 3.3V or 5V supply and has onboard level conversion between the two voltage levels. The display has a 16-bit parallel data bus and supports 4-pin SPI as a communication interface. Retailing at $18.50 on Amazon, the HiLetgo display provides a sharp image, high resolution, and quick refresh rate.

Teyleten 1.28” TFT LCD Display: This round TFT display offers a resolution of 240 by 240 pixels and a size of 1.28”. Like the previous display, it can be interfaced with a microcontroller via 4-pin SPI, and it can be driven by either 3.3V or 5V. It is designed to be interfaced with Arduino, ESP32, and similar microcontrollers. This display has a much simpler pinout than other displays with just seven pins. Customers of this display commented on its brightness, compactness, and image quality. On Amazon, a pack of three Teyleten displays can be purchased for $18.99, or roughly $6.33 per display, making it the most cost-effective display option considered in this section.

ELEGOO 2.8” TFT Touch Screen: The final TFT display considered in this section has a screen size of 2.8” (43.2*57.6mm) and a display resolution of 240*320 pixels. It features an SD card slot for displaying full images on the screen and a resistive touch screen for an interactive display experience. It uses an 8-bit parallel data bus and an SPI interface to communicate with a microcontroller or other driving device. Like the other displays, it is compatible with 3.3V and 5V MCUs and accepts a voltage input ranging from 3.3-5.5V. The ELEGOO display is easily programmable, supporting Arduino libraries for graphics rendering. At $15.99 on Amazon, this display is designed for ease of use, programmability, and creating interactive user experiences.

After considering each option and summarizing the results in Table 3.12, Option 2—the Teyleten 1.28" TFT display—was selected for the SCAN system due to its compact size, simplicity, and cost-effectiveness. To fulfill the kiosk’s goal of providing quick and effortless user interactions without an overly complex interface, the smaller, round display offers the perfect amount of screen real estate to provide clear feedback during the check-in process. Its high brightness and image quality, combined with a straightforward pinout and compatibility with MCUs like the Arduino UNO and ESP32, make it an effective and user-friendly choice. Additionally, with a low cost per display, it fits well within our project’s budget, allowing us to scale up the SCAN system by adding additional kiosks to the network at a lower cost.

Table 3.12 Comparison of TFT Displays

Feature	Option 1	Option 2	Option 3
Brand	HiLetgo	Teyleten	ELEGOO
Screen Size	73.4*49mm	31.2mm diameter	43.2*57.6mm
Resolution	480x320	240x240	320x240
Driver chip	ILI9486/ILI9488	GC9A01	ILI9341
Color	Full RGB	Full RGB	Full RGB
Operating voltage	5.V/3.3V	5V/3.3V	5V/3.3V
Cost	$18.50	$6.33	$15.99
Additional features	N/A	Round display	Touchscreen

3.7 Batteries

Batteries are an essential component of nearly every application, from consumer electronics to industrial equipment, electric vehicles, and energy storage systems. They are central to many modern innovations and technologies. The main, obvious benefit is portability. Instead of being tied to the grid (physically and electrically), batteries allow electronics to be used in any place and at any time. Different battery types offer various trade-offs in terms of energy density, lifespan, cost, and operability, making the battery type choice one that requires evaluation of the system it is being used for. Analyzing SCAN's needs and comparing battery technologies is necessary for optimizing performance and reliability. This section compares various rechargeable battery types, focusing on lead-acid, lithium-ion, lithium-iron-phosphate, and nickel metal hydride. Their suitabilities for the kiosk are compared based on their technical details and different use cases. Additionally, since one of our goals is ease of use, single-use batteries, such as Alkaline batteries, were not considered. Replacing batteries after every use would increase the system's complexity and make it more difficult for users to maintain.

3.7.1 Lead Acid

Lead Acid batteries are one of the oldest and most widely used rechargeable battery technologies. Despite their age, they remain popular due to their low cost and ability to deliver reliable performance. These batteries are especially valued in applications where weight and size are less critical, as they are relatively heavy. They have a low energy density, in the range of 30-50 Wh/kg, and a low specific energy, between 20 and 50 Wh/L, compared to more modern battery technologies like lithium-ion. [56] Their ruggedness and ability to operate in numerous conditions make them ideal for use in

large-scale energy storage systems, renewable energy systems, and uninterruptible power supplies (UPS) for emergency systems.

In terms of technical specifications, Lead Acid batteries have both attractive specifications and some drawbacks. They are known for their ability to provide high surge currents, a feature that makes them ideal for automotive applications where short, high-current bursts of power are required to start engines, even in extreme temperatures. However, they have a mediocre recommended depth of discharge of 50%, meaning only around 50% of their total capacity is usable. Deep discharges can shorten the battery's life cycle significantly, which typically ranges from 300 to 1,000 cycles.

Although Lead Acid batteries self-discharge at a rate of about 5% per month, which is relatively high compared to lithium-ion batteries, they remain an appealing option for their reliability in many industrial and energy storage applications. While newer battery technologies offer advantages in weight, energy density, and form factor, Lead Acid batteries continue to be a reliable, low-cost solution for many applications.

3.7.2 Lithium-Ion (Including LiPo and $LiFePO_4$)

Lithium-ion batteries are a family of rechargeable battery technologies that are commonly used in consumer electronics, electric vehicles, energy storage systems, and more. They are known for their high energy density, low self-discharge rate, high efficiency, and lightweight form. Their versatility makes them popular choices for a range of applications. In fact, most modern smartphones use them for these reasons. Lithium-ion batteries work by moving lithium ions between the graphite anode and lithium-based cathode during charge/discharge cycles. [57] The specific material used for the cathode, such as lithium iron phosphate ($LiFePO_4$), heavily influences the characteristics of the battery. Compared to lead-acid batteries, they offer significantly higher energy density, lighter weight, and longer cycle life, making them an appealing choice for many applications. However, they tend to be more expensive and require more sophisticated battery management systems to perform well.

Lithium-ion batteries typically have a high energy density, ranging from 90 to 200 Wh/kg depending on the specific chemistry. This high energy density makes them ideal for portable electronics, electric vehicles, and other weight and size-constrained applications. Additionally, lithium-ion batteries have a low self-discharge rate of around 2% per month, allowing them to retain charge longer than many other rechargeable batteries when not in use. However, as previously mentioned, they require a more sophisticated implementation because they need a battery management system to monitor parameters like temperature, overcharging, and over-discharging. That being said, some chemistries have quite unique characteristics. Two specific chemistries/types are further explored below:

Lithium Polymer (LiPo) Batteries: Lithium polymer batteries are a subtype of lithium-ion technology that use a gel polymer electrolyte, unlike other lithium-ion batteries that use liquid electrolytes. This allows for flexible shapes and sizes, which

makes them particularly useful in space-constrained applications. Compared to other lithium-ion battery types, they are also rated for higher amperage (higher discharge rate), which means they are better suited for applications that require bursts of high energy. Given their benefits, they are typically found in radio-controlled (remote-controlled) cars, drones, hybrid electric vehicles, and some mobile devices. However, LiPo batteries are more sensitive to overcharging and overheating and are more prone to fire or explosion due to damage. [58]

Lithium Iron Phosphate ($LiFePO_4$ or LFP) Batteries: Lithium iron phosphate batteries are a newer type of lithium-ion technology. As the name suggests, the cathode material is made of lithium iron phosphate. The characteristics of these batteries differ enough from other lithium-ion batteries that they are often classified separately. This battery chemistry prioritizes safety and longevity while falling short in energy density, making them great selections for electric vehicles, electric bikes and go-karts, energy storage systems, and more. Compared to other lithium-ion batteries, LFP batteries have a more stable chemical structure, minimizing the risk of thermal runaway, and in the event of failure, they do not release oxygen as other lithium-ion chemistries do. This prevents combustion and significantly lowers the risk of fire. On top of that, the extraction process for iron and phosphate is considered more environmentally friendly and less geopolitically sensitive. [59]

3.7.3 Nickel-Metal Hydride (NiMH)

Nickel-metal hydride batteries have been a cornerstone of rechargeable battery technologies for many years. They were developed after nickel-cadmium (NiCd) batteries and offered improvements in energy capacity without the environmental concerns associated with cadmium, a toxic heavy metal. Despite their higher self-discharge rate of around 30% per month [60], they are significantly cheaper than lithium-ion, offering an appealing alternative in particular applications. Although they are not suitable for applications such as energy storage systems, applications requiring constant or frequent charging and discharging make NiMH batteries an appropriate, cost-effective solution. They can be found in handheld power tools, cameras, hybrid vehicles, and more. Notably, Toyota has successfully utilized NiMH batteries in its hybrid cars for over two decades [61], demonstrating their reliability in hybrid vehicle applications. However, NiMH technology has been largely taken over by lithium-ion due to the higher energy density and significantly lower self-discharge rate of lithium-ion.

Table 3.13 Comparison of Battery Technologies

Features	**Lead Acid**	**Lithium-ion**	**NiMH**
Energy Density (Wh/L)	50-90	200-300	140-300
Specific Energy (Wh/kg)	30-50	150-250	60-120
Nominal Cell Voltage	2V	3.2 - 3.7V	1.2V

Cycle Life (cycles)	300-500	500-2,000	500-1,000
Self-discharge per Month	3-5%	1-3%	20-30%
Recommended Depth of Discharge	50%	80-95%	60-80%

Based on Table 3.13's comparison, lithium-ion batteries are the most ideal for the kiosk's battery because they offer an optimal combination of lightweight design, reliable power, and longevity. Lithium-ion batteries support a deep discharge without impacting their lifespan significantly, which is ideal for devices like SCAN that may have variable usage and charging intervals. Additionally, they retain their charge well during periods of inactivity, which helps ensure consistent performance even in low-traffic periods. Compared to alternatives like NiMH or Lead Acid, lithium-ion is more efficient and compact, reducing the size and weight of the battery pack, which are important factors for the kiosk's form. While lithium-ion does require a battery management system for safe operation, its high energy density and durability make it the most practical and efficient choice for this project.

3.7.4 Component Selection for Lithium-Ion Battery

For the kiosk's main power source, selecting an appropriate lithium-ion battery helps to ensure adequate runtime and physical constraints of the kiosk. Achieving a balance between energy capacity, physical size, and cost is therefore a key factor. Additionally, certain minimum requirements must be met. For example, the battery capacity must meet the minimum battery runtime requirement specified in **Section 2.4**. Below are three potential battery options that vary in capacity, size, and cost, each with its advantages and potential tradeoffs.

Lithium-Ion Battery Choice 1: The first option is a 12.8 V, 12 Ah $LiFePO_4$ battery pack. This battery offers a compact and efficient solution with a built-in BMS, featuring F2 terminals for integrating it into our design. Measuring 5.95 inches in depth, 2.56 inches in width, and 3.7 inches in height, this battery would be slightly large for our kiosk size. At a capacity of around 150 Wh, this would easily power the kiosk for a few days without needing to be recharged, increasing the usability of the kiosk.

Lithium-Ion Battery Choice 2: The second option is a 12.8 V, 6 Ah $LiFePO_4$ battery pack. This battery offers a compact and efficient solution with a built-in BMS. Measuring also at 5.95 inches in depth, 2.56 inches in width, and 3.7 inches in height, this battery would similarly be slightly large for our kiosk size. However, the main benefit of this option is the cost. At a capacity of around 75 Wh and being half the cost of option 1, this option is certainly appealing. This battery would easily power the kiosk for at least one day, increasing the usability of the kiosk.

Lithium-Ion Battery Choice 3: The third and final option is a 6 V, 6000 mAh $LiFePO_4$ battery pack. This battery offers a cheaper solution relative to option 1, lower voltage,

and smaller size. Measuring 2.76 inches in depth, 1.85 inches in width, and 3.94 inches in height, this solution is a much smaller form. However, one of the tradeoffs of this is the lowest capacity, which is just 38.4 Wh, which is about half of the capacity of option 2. Although this should be able to power the kiosk for more than two hours, it would likely have to be recharged almost every day.

Table 3.14 Comparison of Lithium-Ion Batteries

Features	**Option 1**	**Option 2**	**Option 3**
Brand	XZNY-C1212	Talentcell LF060A1	Talentcell LF4011
Size	5.95” x 2.56” x 3.7”	2.76” x 1.85” x 3.94”	3.43" x 2.68" x 3.86"
Capacity	153.6 Wh	38.4 Wh	76.8 Wh
Voltage	12.8 V	6.4 V	12.8 V
Cost	$39.99	$21.99	$38.99

After considering the three lithium-ion battery pack options, we ultimately chose Option 3 from Table 3.14 due to its compact size and low cost, which aligns better with the design constraints for our kiosk. Despite its lower capacity, this option is more manageable for the kiosk’s limited space, allowing for easier integration. Although it requires more frequent charging, this battery still makes a practical choice and meets our project’s needs, including the minimum runtime. By prioritizing size and cost, we can ensure that SCAN adheres to our design requirements and remains functional.

3.8 Database Technologies

SCAN aimed to provide users with a fast and modernized way of checking in using NFC-based ID tags. In order to facilitate this functionality, a database is needed for storing user profiles, accessing user check-in data, and generating occupancy metrics. This section explores the various types of databases that could be employed for our project, focusing on relational, NoSQL, in-memory, and cloud databases. By evaluating these options, we aimed to identify the most suitable solution that meets the technical requirements and scalability needs of our self-check-in kiosk system.

3.8.1 Relational Databases

Relational databases are an important type of database used since the very start of database systems. They are widely used in industry for applications like banking systems, inventory management, and customer relationship management (CRM) applications. Relational databases can be queried using SQL, which is a standard language adhering to firmly established protocols and standards for communicating between databases and edge nodes - in this case, a SCAN kiosk or web application instance - making it a very strong option.

Relational databases are very simplistic in nature, consisting of tables structured with typical rows and columns where each table row represents an entry in the database, and each table column represents a data type. Each entry in a data table requires a unique ID called a primary key used to distinguish between table entries. Table entries can be linked to entries in other tables using foreign keys. For example, Amazon may structure its database with separate tables for customers and orders. Each row of the customer table would represent an individual customer with a unique primary key, customer ID, while each column would represent a data field tied to each customer like name, address, phone number, etc. Each row of the order table would represent an order with a unique primary key, order ID. Customers can be linked to their orders using foreign keys, which are implemented using columns in the customer table containing a primary key from the order table, linking directly to a specific order in that table.

Relational databases can vary in setup complexity depending on configuration. This type of database can either be hosted locally or in the cloud via a web service. Implementing a local database requires a local server hosting middleware to handle HTTP requests sent over WIFI from the microcontroller to the server's API endpoints and a relational database management system (RDMS) - like MySQL or PostgreSQL- for managing queries. This method can become complicated because of the need for IP forwarding and other issues related to manual setup, management, and security of local databases. Opting for a cloud-based solution requires no extra hardware, just software for querying the remote database. The remote database can handle HTTP requests and sometimes even run event-triggered functions in the cloud to respond to changes. Remote databases also consume less power and typically provide easy-to-use UIs for viewing and interacting with user data. Most of the complexity when it comes to cloud databases comes with initial setup with things like schema, but overall, they are a lot more simple. [37]

3.8.2 NoSQL Databases

Unlike relational databases, which are composed of uniformly structured datasets, NoSQL databases contain loosely or even completely unstructured datasets. While this at first may seem like a disadvantage, NoSQL databases actually offer greater flexibility and scalability compared to traditional datasets. As opposed to tables, as we saw with relational databases, NoSQL databases employ a variety of data models, such as document-based, key-value, wide-column, and graph databases. They are widely used across many application domains due to their adaptability to large volumes of unstructured data. Some example application areas are web and mobile applications, real-time analytics, content management systems, and internet of things IoT.

Unlike relational databases, NoSQL databases do not adhere to the many protocols and standards that other, more traditional, databases do. They utilize various APIs and query languages specific to their data models. A common method of interacting with these databases is through RESTful APIs, which allow HTTP requests to perform CRUD (Create, Read, Update, Delete) operations.

Implementation for NoSQL databases is similar to other databases, such as relational databases. They can either be implemented locally or via a cloud provider. Depending on

the NoSQL type chosen (e.g., MongoDB, Firebase Firestore, Cassandra), the respective database engine will need to be installed or accessed through cloud services. If the database engine is installed and run locally, both hardware and software will be needed to facilitate communication between the embedded application, middleware running on the server, and database. Many libraries are available to facilitate communication between endpoints and these databases. Due to the lack of structure with NoSQL databases, they are relatively easier to setup initially as compared to more structured databases. [38]

3.8.3 Cloud Databases

Cloud databases abstract away the complexity that comes with setting up local databases by storing data in the cloud. Cloud service providers manage the complexities of database management, hardware provisioning, software installation, maintenance, and storing backups, making these databases ideal for modern applications that demand rapid development and deployment and applications like SCAN where our focus is more on the embedded workings. The databases can support relational or NoSQL frameworks and offer the benefits of scalability, flexibility, and high availability. Cloud databases are used in many applications from big data analytics to IoT devices.

Cloud databases employ standard web protocols for communication, including RESTful APIs for interacting with the database using HTTP requests, database connection protocols like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity) for connecting applications to the database, and SSL (secure socket layer) for ensuring safe communication between the applications and the database.

Setting up a cloud-hosted database is a lot simpler than hosting a database locally. Setting up a cloud database involves choosing a cloud provider (e.g., Amazon Web Services, Microsoft Azure, Google Cloud Platform), selecting the appropriate database service (e.g., Amazon RDS, Google Cloud Firestore, Microsoft Azure Cosmos DB), and deciding which libraries and SDKs to use. As a plus, cloud databases typically come with user-friendly interfaces and wizards for configuring and managing your database. The only drawback here may be costs related to using a cloud service, although many providers offer free tiers for small application purposes and testing.

Overall, cloud databases offer a flexible and efficient solution to storing and managing shared data between SCAN kiosks and web application instances. For this application, using a cloud database can enhance the project's reliability and ease of implementation, allowing for more secure transmission of user data and check-in statuses. [39]

Table 3.15 Comparison of Database Technologies

Feature	**Relational Databases**	**NoSQL Databases**	**Cloud Databases**
Data structure	Fixed schema - rows and columns	Flexible data models (key-value, document,	Can be both relational and non-relational;

		column-family, graph)	structure depends on the service
Query Language	SQL	Database dependent	REST APIs, SQL, or proprietary query languages
Performance	Good for complex queries but may suffer with large datasets	High performance for large volumes of unstructured data	High performance with optimizations for speed and availability
Deployment	Typically on-premises or hosted on a dedicated server	Can be on-premises, in the cloud, or hybrid	Fully managed by cloud service providers, accessible via the internet
Cost	Cost of infrastructure and management; licenses may apply	Usually lower infrastructure costs, but may require more management	Pay-as-you-go pricing, typically more cost-effective for variable workloads
Examples	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, Redis, Couchbase	Amazon RDS, Google Cloud Firestore, Microsoft Azure Cosmos DB

Factoring in all of the details listed in Table 3.15, we ultimately decided to use a cloud database for our chosen database technology. Cloud databases offer more flexibility, ease of integration, and performance, all at little to no cost. Other database solutions require a more complex setup involving additional hardware and software systems to perform middle-man operations. Running a local database is also both less secure and costly in terms of power consumption. Given the scope of this project and constraints like time and knowledge, a cloud-based solution was the best option for this project.

3.8.4 Database Selection

There are a few popular options when it comes to selecting a cloud-based database. Some of those include MongoDB, Firebase, and Microsoft Azure. Each of these comes with its own strengths and drawbacks in terms of scalability, ease of integration, real-time data handling, security, and cost. This section explores these three databases and their particular aspects in consideration of this project's objectives.

Firebase is Google's cloud-based platform for mobile and web application development. It offers a variety of tools and products to help developers build and run applications, the most important for the purposes of SCAN being FireStroe. Firebase's Firestore is a NoSQL database that allows users to store, sync, and query data stored in collections and documents. It offers an automatically scaling database, offline support, real-time synchronization, easy integration with other Google Cloud products, and the capability to handle complex queries, all at a minimal cost. [40] On the implementation side, Firebase is a commonly used tool for data storage, meaning there are ample resources online, even a Firebase library for Arduino. Firebase also handles all data as JSON objects, making data easy to work with, especially in real-time applications where user activity data needs to be updated instantly. Another positive aspect is that Firebase offers a free tier for limited-scope projects, for which our design met the requirements, meaning we only incurred minimal costs.

While there are many positive aspects to using Firebase as a cloud database solution for this project, there are also a few potentially negative side effects to consider. One is its aptitude for handling complex queries. Firebase is less capable of handling complex queries due to its NoSQL structure and definitely lacks sophistication compared to more customizable solutions like MongoDB and Microsoft Azure. Another potential downside is cost. While using Firebase is initially free, fees may be incurred at larger scales, especially if high-frequency transactions occur regularly. Firestore is also heavily reliant on an internet connection, meaning a good amount of custom-written logic for storing check-in data locally would have been needed to be implemented to handle internet outages or a simple lack of connection.

MongoDB is another NoSQL cloud database provider that offers a more customizable document-based model. Similar to Firebase, it uses a JSON-like document structure that allows for more complex management when designing complex relationships. MongoDB has its strengths in horizontal scalability, meaning it can easily be scaled to support vast amounts of data. It also provides a more economical cost model than other solutions like Firebase.

There is, however, one central area of concern when it comes to MongoDB. MongoDB lacks the out-of-the-box real-time capabilities that Firebase's Firestore offers, meaning a lot of extra work would have been needed to use this method of storing data if we decide to go in this direction. SCAN relies on real-time querying as all check-in data is stored in the cloud and needs to be accessed in real-time to ensure accurate check-in information. [41]

Microsoft Azure offers Azure Cosmos DB, a flexible database option that provides both NoSQL and SQL-based data models, meaning it offers a great amount of flexibility. Comparable to Firebase, it offers real-time processing, but it also offers complex querying and indexing like MongoDB. Microsoft Azure is commonly used in larger companies for its strong data protection policies. Azure also offers a free tier for limited usage, similar to Firestore.

While Azure offers an abundant number of advanced features, these may not all be suited for the scope of this project. In fact, the learning curve for using and implementing many of these features makes this tool nearly infeasible for this project. While Azure does offer a free tier, it's important to note that this advanced tool can quickly exceed the budgeting scope of this project. [42]

Table 3.16 Comparison of Cloud Databases

Feature	Firebase	MongoDB	Microsoft Azure
Data Model	NoSQL (JSON)	NoSQL (JSON-like)	NoSQL and SQL
Real-Time Sync	Best	Requires additional layers	Complex Setup
Querying Capabilities	Limited for complex queries	Robust querying and indexing	Strong SQL/NoSQL querying
Ease of Integration	Seamless, no middleware needed	More complex	Most complex
Security	Google Cloud basic encryption	Flexible, strong encryption	Enterprise-grade security features
Cost	Least	Middle	Most

Considering the factors detailed in the sections and Table 3.16, we ultimately selected Firebase's Realtime Database as the most suitable option for our database because of its real-time capabilities, ease of integration, and ample online resources and libraries. Since one of SCAN's core objective principles is to have a fast centralized data storage system for synchronized data transmission between various edge nodes, Firebase's real-time capabilities are essential to this project. They offer an optimal and fast solution that best meets our time and technical knowledge constraints. Firebase's data pricing model also offers a lot of leniency with its free tier, meaning we are unlikely to incur any out-of-pocket cost for this project. While Firebase may not provide some of the higher-level, more complex features of other cloud storage solutions, it is still our best option as these features are not only more difficult to learn and implement but out of scope for this project.

3.9 Frontend Frameworks

To design the analytics dashboard, which visualizes and displays the check-in data from Firebase, a more powerful frontend framework was required than the basic HTML, CSS, and JavaScript used in our Senior Design webpage. Three of the most popular frontend frameworks are the JavaScript frameworks Angular, ReactJS, and Vue. Ultimately, these frameworks have many of the same features and would have suited our purposes, but

they differ in terms of ease of use, documentation, and interfaces with our existing technologies.

3.9.1 Angular

Angular is an open-source framework developed by Google and written in TypeScript. It is suited for mobile and web app development for single-page applications (SPA). In Angular, applications are developed as a combination of components, each with their own styles, template, and logic. By using components, Angular applications are more modular, and code can be reused for similar elements. In Angular, data is synchronized between the model (backend) and view (frontend) automatically using two-way binding. With two-way data binding, if changes are made in the model state, such as updating the number of checked-in users, the view state will update automatically. Angular also has built-in routing, which allows single-page applications to have multiple views. This feature enables more complex user interface navigation with features such as nested routes, lazy loading, and route guards. Millions of sites and apps are built using Angular, including The Guardian, PayPal, Lego, and Netflix. [63]

3.9.2 ReactJS

ReactJS, or simply React, is an extremely popular Javascript framework, developed and maintained by Meta, used for web development and user interface design. It is a component-based JavaScript library that enables efficient creation of interactive UIs for primarily single-page applications. One of the main differences between React and Angular is that React is a JavaScript library, while Angular is a full-fledged framework, which means that React can require more third-party libraries for more complex features. ReactJS uses a virtual DOM, or document object model, to optimize web page rendering performance. When a component's state is altered, such as updating a counter for the number of students currently in the RWC, the virtual DOM is updated first. After updating the virtual DOM, only the affected portions of the physical DOM are updated. React uses the JSX extension for JavaScript, which makes visualizing the structure of a webpage by allowing developers to write HTML-like code within JavaScript. In contrast to Angular's two-way data binding, React follows a one-way data flow, where data flows from parent to child components.

React can be used to develop the frontend for a variety of applications, including SPAs, progressive web apps (PWA), E-commerce platforms, real-time data applications (such as SCAN), and mobile applications (using React Native). Importantly for our project, React can interface directly with Firebase using Firebase SDKs. Combined with Firebase's Cloud Functions, React applications can be extended to support server-side logic without requiring direct management of servers. In summary, React can be used to develop dynamic user interfaces that are responsive to changes made from real-time databases like Firebase. [64]

3.9.3 Vue

Vue is another open-source framework suited for designing single-page applications and responsive interfaces. It was released in 2014 by a software developer from Google. Vue is frequently described as a compromise between the flexibility of React and the rigid structure of Angular's framework. It supports the two-way data binding used in Angular to allow synchronization between the model and view but also supports React's Virtual DOM. Vue can be used for both simple and complex applications. Its popularity has rapidly increased in the decade since its release, and learning to use Vue can be easier than other frameworks, especially if developers are already familiar with Javascript, HTML, and CSS. Similar to the other frameworks, it can be used to develop SPAs, interactive dashboards, E-commerce platforms, and mobile applications. Some of the top applications developed using view are websites for Alibaba, Reuters, and Xiaomi. [63]

Table 3.17 Comparison of Frontend Frameworks

Feature	**Option 1**	**Option 2**	**Option 3**
Name	Angular	ReactJS	Vue
Key Features	Two-way data binding, Dependency injection	Component-based, Virtual DOM	Reactive binding, Virtual DOM
Ease of Use	Full-featured framework, more complicated syntax	Easy to learn for devs familiar with JS, extremely modular	Smaller ecosystem, easier to learn
Runtime Speed/ Responsiveness	Very good, Ahead-of-Time (AoT) compilation	Excellent, highly efficient for dynamic updates due to virtual DOM	Excellent, comparable to React
Support and Documentation	Smaller community than React but detailed documentation	Extensive and up-to-date documentation	Smaller community but concise and beginner-friendly documentation
Advantages	Complete framework, perfect for large, integrated applications	High reusability of components and efficient rendering	Simple syntax and smaller codebase
Disadvantages	Large size, steeper learning curve	Can become complex with many components and libraries	Smaller developer base and ecosystem compared to React and Angular

Ultimately, after summarizing the frontend frameworks in Table 3.17, we selected React as the framework for the SCAN webpage for its compatibility with our requirements, component-based architecture, and extensive set of libraries and add-ons. Through Firebase's JavaScript SDK, we can easily interact with the Firebase Realtime Database to extract user analytics. React's state management and virtual DOM handle real-time updates from Firebase efficiently, allowing us to meet our goal of a responsive refresh rate for check-in statistics. Furthermore, the straightforward interfacing of Firebase and React streamlined the process of fetching, displaying and synchronizing data.

4. STANDARDS AND DESIGN CONSTRAINTS

There were many design standards and constraints to be considered concerning the SCAN project that guided and shaped its design and implementation. Adhering to wireless communication standards like WIFI and NFC ensures interoperability and reliable transmission between devices. Industry standards like SPI and I2C were also important for peripheral interfacing. Additionally, programming standards were followed to enhance code readability, maintainability, and collaboration, while power standards ensured safe and efficient energy consumption. Outside of standards, our project was also shaped by various constraints including economic, sustainability, environmental, political, and ethical. This section details how these standards and constraints interacted to create a balanced, compliant, and innovative design solution.

4.1 Design Standards

As we developed the SCAN system, we adhered to relevant technology standards to ensure reliability, safety, and interoperability throughout its lifecycle. By following established and maintained design standards, the system maintains consistency in hardware and software, allowing for easy integration with existing technologies such as NFC readers, microcontroller interfaces, and databases. Standards also ensure that the system meets requirements for user safety in areas where it will be deployed, especially in environments with high traffic like the RWC. Furthermore, design standards helped in regulating quality control, ensuring that components such as the display, communication modules, and power supply were compatible and efficient. By following recognized industry standards, the SCAN system is easier to maintain, upgrade, and scale.

4.1.1 Wireless Communication Standards

The SCAN system requires secure, reliable wireless communication protocols to facilitate communication between the kiosk and external devices, such as the cloud database. By adhering to wireless communication standards, SCAN ensures reliable data exchange, user authentication, and real-time access to analytics data. This section examines two primary communication standards: Near-Field Communication (NFC) for short-range, secure user interactions and WiFi for broader, high-speed data transmission. These standards were analyzed to determine their technical specifications and operating protocols, ensuring the project met its goals for performance, accessibility, and security.

4.1.1.1 Near-Field Communication Standard

To enable hardware token authentication during the check-in process, an NFC tag and reader system were used. Adherence to the NFC standard was necessary to encode user data on the tag in the correct format and then receive the correct data with the reader. The NFC Forum is the governing body responsible for the NFC standard. It defines standards for all modes of operation, tag types, and data formats. [43]

As mentioned previously, NFC is a wireless communication technology operating over short distances of four centimeters or less. It operates at the 13.56MHz frequency band

and expands on existing standards for RFID. The standard defines an active NFC reader, which is also known as an initiator or interrogator, to generate the RF field necessary to communicate with NFC tags or other NFC-compatible devices. The reader facilitates the exchange of data through three modes of operation:

1. Reader/Writer Mode: Reads data from or writes data to NFC tags
2. Peer-to-Peer Mode: Manages the data exchange between two NFC devices
3. Card Emulation Mode: The NFC device acts as a contactless smart card

NFC readers support ISO/IEC standard communication protocols such as Type A and B **ISO/IEC 14443**, **ISO/IEC 15693**, and **FeliCa**, which ensures that they are compatible with existing contactless smart cards and tags. The SCAN project is most concerned with Reader/Writer mode, as this is the relevant standard for the passive tag, active reader configuration necessary for the SCAN kiosk.

Protocol Stack: The NFC standard defines a protocol stack governing how communication, data exchange, and power are managed at the application, digital encoding, and analog transmission levels. **Figure 4.1** provides an overview of the NFC protocol stack for each mode of operation. At the top of the protocol stack is the application layer, which governs how end users interact with the NFC technology. In our case, users interact with the NFC transceiver through Reader/Writer mode.

Located at the bottom of the stack, the digital and analog specification layers handle the physical layer of the protocol stack. These layers define the electrical characteristics of NFC communication, including the signal strength, frequency of transmission, modulation schemes, and error correction, ensuring the accurate transmission of data from NFC devices.

Tag Standards: Also from Figure 4.1, the tag specifications layer is used to define how an NFC-enabled device (such as the kiosk reader) reads and writes a message on an NFC tag. There are five tag specifications, each supporting different wireless standards, data capacities, transmission rates, and use cases. **Table 4.1** summarizes Type 1-5 NFC tags. The NFC tags used with the SCAN system are Type 2 tags, which are commonly used for smart business and ID cards. Although they have smaller data capacities, Type 2 tags support higher data rates and the common ISO/IEC 14443A standard for wireless communication.

Table 4.1 Summary of Standard NFC Tag Types

Feature	**Type 1**	**Type 2**	**Type 3**	**Type 4**	**Type 5**
Standard	ISO/IEC 14443A	ISO/IEC 14443A	Based on FeliCa (JIS X 6319-4)	ISO/IEC 14443A/B	ISO/IEC 15693
Memory Size	Up to 2KB	Up to 48B	Up to 1MB	Up to 32KB	Varies
Features	Read/write	Like type 1	High-speed	Pre-configur	Longer read

	capable, low cost	but with faster data rate	communicat ion	ed; supports high data rates	range (up to 1 meter)
Use cases	URL redirection and other simple uses	Smart posters, business cards	Electronic payments and transit systems	Secure transactions, government IDs	Inventory management , logistics

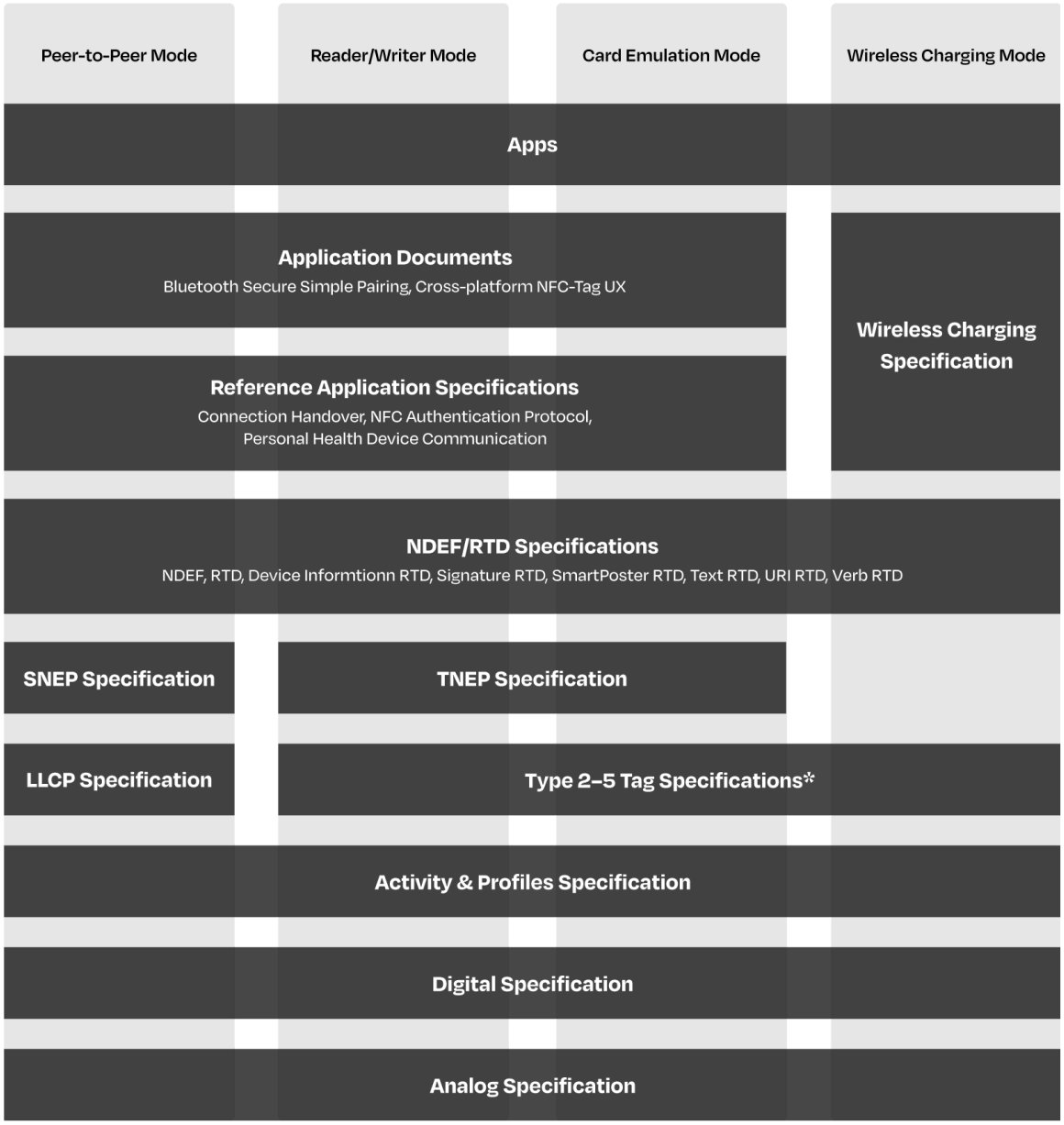


Figure 4.1 NFC Specification Architecture (Obtained from NFC Forum)

The data on all tag types is formatted according to the NFC Data Exchange Format (NDEF). In NDEF, messages are composed of one or more NDEF records, which have a defined structure and carry a specific payload. Table 4.2 displays the structure of an NDEF record.

Table 4.2 NDEF Record Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
[MB]	[ME]	[CF]	[SR]	[IL]	[TNF]		
TYPE LENGTH							
PAYLOAD LENGTH							
ID LENGTH							
RECORD TYPE							
ID							
PAYLOAD							

The bits in the record byte are defined as follows:

- MB: Indicates the start of an NDEF message
- ME: Indicates the last record in the message
- CF: Chunk Flag: Indicates whether it is the first record chunk
- SR: Short Record Bit: Set to 1 if PAYLOAD LENGTH field is <= 1 Byte
- IL: ID LENGTH Field: Set to 1 if ID length field is present
- TNF: Describes record type and structure

The type length, payload length, and ID length are all used to define the size of their respective sections. Type length will always be zero for certain record types defined in the TNF field. Payload length will be a one-byte field if SR is set to 1, else it will be a four-byte field, giving the theoretical payload size a maximum value of 2^32 bytes. The ID length field is only present if the IL flag is set to 1. The record type and ID fields describe what type of record format the payload will take and what its associated ID is. A full list of records is described by the specification, but two formats include a MIME Media Record and a Well-Known Record, which is one of the most frequently-used record types. Finally, the payload field contains the data for the record, which is the exact size in bytes specified by the Payload Length Field. [44]

4.1.1.2 WiFi Standard

For centralized communication between SCAN's embedded application, backend server, and web application, our design team settled on WiFi as the most feasible option for data transmission. WiFi standards are some of the most widely adopted transmission standards

used today, facilitating generic communication between a wide range of devices, ultimately serving as the backbone of the internet.

Since the pioneering of WiFi with the release of the first WiFi standard, 802.11, in 1997, there have been six more WiFi standards released, with a seventh expected to be released this year (2024), each of which bearing its own set of revelations and consequently specifications. Each standard bolsters tradeoffs in terms of range, speed, and power consumption which are extremely important considerations in ensuring embedded applications meet project requirements. This section gives a brief history of WiFi standards and details a few specifications specific to this project's application purposes.

WiFi technology has seen rapid advancements bringing about a near 20,000x speedup over the course of its ~25-year lifespan. WiFi began with the release of the IEEE 802.11 standard in 1997, supporting data transmission rates of only about 2 Mbit/s on the 2.4 GHz spectrum. This standard quickly evolved with the release of revolutionary Apple products, quickly fortifying this technology and leading to the creation of the 802.11b standard bolstering data rates of up to 11 Mbits/s in 1999. That same year, the standard 802.11a (Wifi 2) was released, featuring a multi-carrier modulation scheme offering 20 Mbits and stepping into the 5 GHz region. 2003 saw the release of WiFi 3 (standard 802.11g), featuring data rates of 54 Mbits/s. WiFi 4 (802.11n) came about in 2009, significantly jumping to 600 Mbps/s transmission rates and providing various other features. WiFi 5 (802.11.ac) jumped to 3.5 Gbits/s utilizing multiple input/output technology in 2013. More recently, WiFi 6 (802.11ax) was released in 2021, offering nearly 10 Gbits/s but improving efficiency when it comes to many devices operating in proximity. Finally, we expect to see the release of WiFi 7 this year, offering a remarkable 40 Gbits/s and increased bandwidth.

For the SCAN project, we used a microcontroller with built-in communication capabilities via the **WiFi 4 standard** (**IEEE 802.11n**). IEEE 802.11n, again, was introduced in 2009 and offers a very capable and yet manageable specification that can be met by less powerful microcontrollers. Compared to previous standards like 802.11b and 802.11g, these standards offer significant improvements in terms of coverage, reliability, and throughput, making it a viable option for many embedded microcontrollers.

The 802.11n standard offers many features that contribute to its unique and attractive characteristics. It allows for data transmission up to 600 Mbits/s, meaning it can support hyper-fast data transmission by today's standards, definitely more than enough for SCAN's use case. This is supported by the use of multiple input/output (MIMO) technology that allows multiple antennas to transmit and receive data in parallel, allowing for more reliable data transmission. This standard also allows for connection ranges of up to 70 meters, with further ranges achievable over the 2.4 GHz band. Again, this is more than enough for this project's application. [45]

4.1.2 Wired Communication Standards

To establish efficient, reliable, and high-speed connections between SCAN's internal components, including the display, proximity sensor, and NFC reader, wired communication standards such as Serial-Peripheral Interface and Inter-Integrated Circuit communication were used. The following section provides an overview of the wired communication standards integral to the operation of the SCAN system. These standard protocols are widely used in embedded systems for connecting microcontrollers to peripherals such as sensors, displays, and memory modules. These protocols ensure accurate, low-latency, and reliable data transmission over short distances. By examining the standard features, configurations, and limitations of SPI and I2C, we were able to optimize SCAN's performance, compatibility, and scalability.

4.1.2.1 Serial-Peripheral Interface (SPI) Standard

Serial peripheral interface (SPI) is a de facto standard for a synchronous protocol that transmits data over two wires. It is widely used to interface over a short range between microcontrollers and peripherals like sensors and ADCs. The protocol operates using a master-slave technique, where one device (the master) controls the communication and the other device(s) (slaves) respond based on instructions from the master.

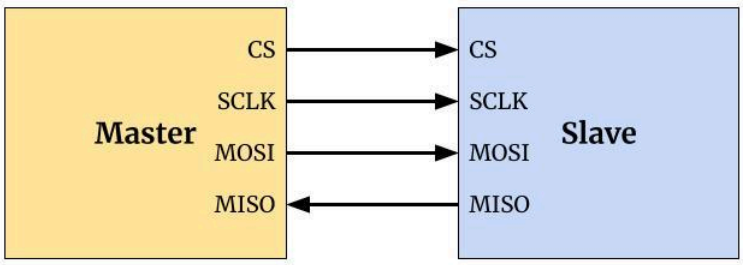


Figure 4.2 SPI Bus Diagram

As seen in **Figure 4.2**, there are typically 4 wires in SPI. At the top is the chip select signal (CS), which is used to identify a specific slave device. The master toggles this line to ensure that only one slave is active at any given time. Below that, there is the serial clock (SCLK). The master generates this clock signal, sharing it with its slave devices to synchronize data transmission. Lastly, there are the MOSI and MISO lines. The Master Out Slave In (MOSI) line is used for the master to send data to the slave, while the Master In Slave Out (MISO) is used by the slave to send data to the master. Given the two separate signals for communication, SPI is able to perform in full-duplex mode, meaning the master and slave can send and receive data simultaneously. This increases the protocol's efficiency compared to other protocols like I2C.

In SPI communication, the packet size can be either fixed or variable. Although it is flexible, most common packet sizes are 8, 16, or 32 bits. One important part of the SPI protocol is its mode, which is determined by the clock polarity (CPOL) and clock phase (CPHA).

Table 4.3 SPI Modes of Operation

SPI Mode	CPOL	CPHA	Description
0	0	0	Sampled on rising edge, clock idles low
1	0	1	Sampled on falling edge, clock idles low
2	1	0	Sampled on falling edge, clock idles high
3	1	1	Sampled on rising edge, clock idles high

Table 4.3 summarizes the four SPI modes. The clock polarity has two options. For CPOL = 0, the clock line idles at a logic low level (low voltage). For CPOL = 1, the clock line idles at a logic high level (high voltage). Similarly, there are two options for the clock phase, which determines at which clock edge data is sampled in each cycle. For CPHA = 0, data is sampled on the rising edge of the clock. For CPHA = 1, data is sampled on the falling edge. [52]

Despite the efficiency and speed of SPI, it also has limitations. Since each slave requires its own CS line, systems with many slave devices can face complex implementation and management with all of the signal lines. Additionally, SPI does not include any acknowledgment or error-checking systems by default. This means developers need to implement these features independently if desired.

4.1.2.2 Inter-Integrated Circuit Standard

I2C, or Inter-Integrated Circuit, communication is a standard protocol governing short-range communication between two or more integrated circuits. It was developed and is maintained by NXP Semiconductors and is used in a variety of consumer electronics, telecommunications equipment, and other embedded systems. The I2C protocol has a bidirectional 2-wire bus, which consists of a Serial Data Line (SDA) and Serial Clock Line (SCL). [46]

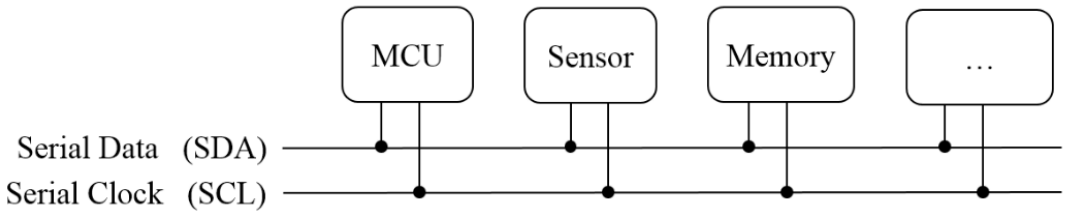


Figure 4.3 I2C Bus Diagram (Obtained from embedded systems lab manual)

As seen in Figure 4.3, all devices connected to the bus share the SDA and SCL lines. There is typically one master and one or more devices in a standard I2C bus, although there are configurations for multi-master buses. Using a master-device configuration, the master device initiates all communication and controls the clock line. The master is able

to address individual devices using a seven-bit device address, meaning that the master can address up to 128 devices. Each I2C bus is active low, meaning the wires are pulled to high when data is not being transmitted. The standard frequency of I2C communication is 100KHz, but most MCUs and I2C devices also support a fast mode, which goes up to 400KHz. Some I2C protocols can reach data rates up to 5 MHz.

To issue read and write commands to an I2C device, the master follows a standard communication protocol, where it issues a start command to the bus followed by the address of the device and the read/write flag. The Master then waits for the device to acknowledge the request before reading or writing the data. Data is read/write bit-by-bit, MSB first, at each edge of the Serial Clock Line. The full transmission sequence for an I2C read command can be seen in **Figure 4.4**. In the figure, the master is reading two bytes, 0x12 and 0x34, from a device at address 0x22. To end the transmission, the master will issue a stop command by pulling SDA high.

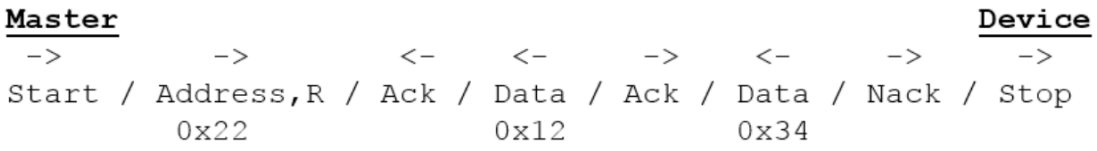


Figure 4.4 I2C Read Command (Obtained from embedded systems lab manual)

Similarly, the I2C write command shown in **Figure 4.5** involves the master issuing a start command followed by the address of the device to be written to (0x33) and the write flag. The master waits to receive an acknowledgement from the proper device before writing the two data bytes (0xAB and 0xCD). Finally, the master pulls the SDA line high, signaling the end of transmission.

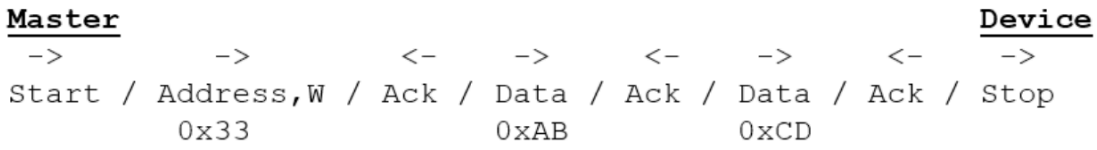


Figure 4.5 I2C Write Command (Obtained from embedded systems lab manual)

4.1.3 Programming Standards

Adhering to established programming standards ensures that SCAN's software system is reliable, maintainable, and sustainable. There are various coding standards all relating to various aspects of coding that can serve to make the codebase more consistent and accessible, which is invaluable for working in a time constrained environment with multiple team members interacting with the same code base. Coding standards are especially important when working with lower level embedded software where code efficiency is important for real time processing.

One of the most fundamental standards in the realm of embedded software is the C standard (ISO/IEC 9899:2018 or C18). This standard ensures code written for the SCAN system adheres to universal rules and conventions lending to a much more portable, clean, and efficient code base. It also ensured type safety, proper memory management, and compatibility with embedded systems, which helped when developing firmware that interacts with hardware components like SCAN's NFC readers and proximity sensor.

There are also various standards for web development that were taken into account when designing our web application. These include HTML5, CSS3, Javascript (ES6+), and Accessibility (WCAG 2.1). Like the C standard, each of these standards comes with their own rules and conventions for uniform code generation. We implemented these standards using linters and code checks via Github.

4.1.4 Power Standards

In addition to communication standards and programming standards, complying with power standards is a necessity for ease-of-use and compatibility with power supply devices. A suitable power standard ensures compatibility, consistently delivers adequate power, offers safety for users, and provides flexibility for future changes or scalability. The power standard that is used to power the kiosk and recharge the kiosk's battery is explored below to understand the technical specifications and requirements necessary for its operation.

4.1.4.1 USB Power Delivery Standard

USB Power Delivery (USB PD) is a specification that allows for more flexible and higher power levels in the charging devices compared to traditional USB standards. Introduced by the USB Implementers Forum (USB-IF), USB PD enables devices to "negotiate" power requirements and deliver up to 240 watts over USB Type-C, which is sufficient for charging even power-hungry laptops. [53] This standard enhances the overall charging experience by allowing devices to communicate their power needs, facilitating faster charging times and the versatility of a single port for charging multiple devices.

The negotiation process in USB PD is conducted when two USB PD devices are connected. The steps are as follows:

1. The device requesting power (sink device) sends a "Discover" message to the device providing power (source device) to identify its capabilities.
2. The source device responds with its available voltage and current charging capabilities in a Source_Capabilities message followed by a CRC.
3. The sink device sends a GoodCRC message.
4. Once the sink device has received and analyzed the available charging options, it sends a "Request" message indicating the desired power level.
5. The source device sends a GoodCRC message.
6. The source evaluates the request, then sends an "Accept" message followed by a CRC.
7. The source device moves the VBUS voltage to the new negotiated voltage level.
8. The sink device sends a GoodCRC message.

9. The source device sends a PS_RDY (power supply ready) message followed by a CRC.
10. The sink device sends a GoodCRC message. [54]

One of the key features of USB PD is its capability for bi-directional power delivery. Although this is not a feature every device supports, the capability means that not only can a device draw power from a source, but it can also supply power to other sources. The negotiation process specifically allows the devices to communicate their power requirements and capabilities in the form of available voltage and current levels to determine power to be delivered.

Another important aspect of USB PD is its support for multiple power profiles. As mentioned before, USB PD Revision 3.1 supports up to 240 watts of charging, but there are many power level profiles within the USB PD standard. It includes typical voltage levels, like 5V, 9V, 15V, 20V, and now all the way up to 48V to support 240 watts. Additionally, USB PD incorporates safety features, like over-voltage protection and temperature control, to prevent overheating and device damage.

Overall, the USB PD standard provides a uniform charging and power sharing technique for devices. It makes USB a more versatile and efficient way of charging. Its latest integration into Apple products, among other smartphones, laptops, and electronics, exemplifies a shift towards greater acceptance of the USB PD standard.

4.2 Design Constraints

We also considered the relevant constraints of our project. A constraint is a key decision in the design process imposed by stakeholders or the environment that affects or limits the design. Constraints were used to specify how the SCAN system was implemented and guided the SCAN team in making informed design choices. The decisions influenced key aspects of the system, such as its cost, feasibility, and societal/environmental impact.

4.2.1 Economic Constraints

One of the most important constraints of this project is the economic constraint. Economic constraints involve the costs of a system's design, manufacturing, maintenance, and sale. The SCAN system’s affordability is a distinguishing factor that sets it apart from other smart kiosk competitors. Furthermore, this project was self-funded by its group members, and we intended to keep the individual cost per member as low as possible. For these reasons, we set a target constraint of $120 for each SCAN kiosk, and aimed to keep the budget under a fixed amount of $200 for each group member. This cost includes the expenses related to the design, prototyping, and manufacturing of the SCAN system.

To meet this constraint, a careful trade-off of cost and performance was considered when selecting components and manufacturing the SCAN system. Components were weighed in terms of performance, reliability, and affordability. For example, we selected the ESP32 as the microcontroller to power our smart kiosk, as it has a sufficient amount of

GPIO pins and supports all necessary communication protocols to interface with our sensors, display, and cloud database. Other more powerful microcontrollers such as the Raspberry Pi may have provided the same or even increased functionality, but the ESP32 is sufficient for the design requirements and is more cost-effective.

Another example is the display technology selected for the kiosk. On paper, OLED displays offer many benefits over TFT displays in terms of image quality, brightness, and color contrast. However, the higher cost and potentially higher power consumption of OLED displays make the TFT display a much better option, given the economic constraints. The decision to use TFT displays not only saves on immediate costs but also keeps power consumption lower, reducing potential long-term costs for power or battery replacements if the check-in kiosk is deployed in locations without access to a power outlet.

In addition to component selection, we also considered cost-saving measures during the prototyping and manufacturing stages. The use of 3D printing and off-the-shelf fasteners and attachments are cost-effective alternatives to a custom-ordered enclosure for the kiosk, which would have increased both material and labor costs for the project. One member of the group already possessed a 3D printer and the necessary filament, eliminating the need for any external labor and material costs. Sourcing components from manufacturers and distributors with student discounts and other price-saving options such as buying in bulk further reduced the material costs for the project.

Lastly, consideration was given to the scalability of the SCAN system for future design iterations or commercial production. By ensuring that the initial prototype and design were cost-effective, we laid a foundation for future improvements or mass production of SCAN kiosks, where the bulk order of readily available components can further reduce costs. This careful attention to cost management across all stages of the project helps position the SCAN system as an economically viable solution in its current form and any future implementations.

4.2.2 Time Constraints

In addition to economic constraints, time constraints presented another significant influence on our design. With a completion date within less than a year, timing was everything. This project faced a tight schedule that drove us to make thorough decisions and meticulously determine our use of time. Some of these decisions include the technologies to use, programming language to develop in, testing framework to adopt, and PCB design and 3D CAD software to utilize. The necessity to develop a working prototype in such a short amount of time means that we had to choose technologies and programming languages that we were familiar with, while the quick turnaround for a final product required selecting parts and using software that we were already familiar with.

For example, in the early stages of development, we needed to select parts for prototyping. Some of these parts for SCAN include NFC scanners, rechargeable batteries, and processors. In the case of our selected NFC scanners, we need time to familiarize

ourselves with the relevant protocols and data formats used in near-field communication. Furthermore, understanding how to interface with NFC scanners and NFC tags was necessary to start prototyping. The integration of NFC in this project required both hardware considerations and software development to manage data from user check-ins and authentication. Understanding the transfer of data in software was another important aspect of developing the NFC portion of SCAN.

In addition to NFC, working with Wi-Fi presents more time constraints. Not only is Wi-Fi a complex networking functionality, but there are also tons of security standards and data transfer interactions to understand on the backend. For example, using Wi-Fi requires knowledge of protocols like TCP/IP and DHCP and security standards like WPA2. Being able to implement Wi-Fi therefore presented a significant time constraint due to the learning curve associated with the networking protocols. One of the ways this time constraint directly affects the project is in the processor selection. Although one option is to purchase a separate Wi-Fi module and processor, integrating the two is arguably necessary because it eliminates the need to spend additional time interfacing the two components. Lastly, in terms of components, the power supply selection was limited by time constraints. Since this section was not the main focus of the product, selecting a battery pack and charger that could be quickly implemented was highly beneficial.

After selecting parts and initial prototyping, the project timeline continued to be constrained by factors like PCB design time, shipping times, and testing and debugging time. For instance, learning a new PCB design software would have added more time that we could not have guaranteed that we would have. As a result, schematic and PCB design software selection is limited to software we are already familiar with. In terms of external factors (shipping times, manufacturing times, etc.), the selection of the PCB manufacturer must have been well-informed. Shipping times for components such as PCBs and other hardware could have introduced significant delays if not planned for adequately. If we selected PCB components that had longer lead times or were backordered, it could compromise our integration and testing timeline. Therefore, selecting suppliers with reasonable shipping times, though a constraint, helps ensure that our product could be delivered on time.

Finally, the testing and integration phase was affected by our time constraints, requiring us to take a strategic approach to our programming, testing framework, and integration technique. For example, a more profitable business approach might include automated testing to be able to rapidly verify many products at once. However, with SCAN, attempting to automate certain testing portions may have brought more drawbacks than benefits. Therefore, while some level of testing automation was sought after, balancing the complexities of automation with a stringent timeline required effective collaboration and project management practices.

4.2.3 Sustainability Constraints

Sustainability constraints consider the long-term environmental, social, and economic factors that shape the viability of the SCAN system. These constraints ensure that the

system not only functions effectively now but also maintains a feasible path to longevity. For our project, sustainability touches on key areas such as component sourcing, energy consumption, and end-of-life disposal.

One of the most significant sustainability constraints is energy efficiency, which plays a large role in the SCAN system's longevity. Since SCAN is used primarily in public spaces and runs continuously, using power-efficient components was essential to minimize its environmental footprint. Respecting this constraint involved incorporating low-power components, such as energy-efficient microcontrollers and sensors. The implications of this constraint were seen in our use of a low-power mode, which activates higher-power components only when a user is detected. To achieve this, the system uses a passive infrared (PIR) sensor, which adds an extra layer of efficiency by reducing unnecessary energy consumption.

Another large consideration for the sustainability of the SCAN system was component sourcing. Components must be sourced from a reputable source where we are able to ensure enough supply for the lifetime of the product. If components are likely to become obsolete or are hard to source, substantial revisions would need to be made to the product design. To avoid this, we considered our sources' reliability and purchased a lifetime supply of the desired products.

When purchasing and building SCAN's components, we also considered disposal towards the end of SCAN's lifetime. This means SCAN's components must be able to be responsibly recycled or disposed of in an environmentally friendly way. Careful attention was paid to product documentation to ensure the environmental sustainability of components. For the parts that we designed on our own, we were mindful of the materials used. Since these structures and housing were 3D-printed, we were primarily concerned with the type of plastics used. The plastics we used were biodegradable, renewable, easily disposable, and nontoxic. Being biodegradable means disposal is of no concern, and being renewable means this type of plastic can be easily sourced long into the future. Nontoxic materials also contribute to the ease of disposal and safety constraints. One of the most widely used plastics that meets these constraints is PLA. Because of this, we decided to use PLA wherever possible to ensure sustainability constraints were met.

Another sustainability constraint is keeping costs low, which directly ties into environmental impacts. A more energy-efficient system not only reduces SCAN's environmental footprint but also lowers its operational costs, making it a more attractive, long-term solution. By keeping SCAN affordable, we created a more economically feasible option for organizations, ensuring it stands out in the market. Ultimately, the success of SCAN hinged on balancing these environmental, economic, and social considerations, which were key to its sustainable future.

4.2.4 Environmental Constraints

Environmental constraints refer to the constraints imposed by the physical conditions of where the SCAN system is operated and used. With a primary use case of providing check-in analytics for the UCF Recreation and Wellness Center's indoor facility, the

SCAN kiosk is not exposed to any extreme weather conditions. Still, some degree of moisture and temperature resistance was necessary to ensure reliability. All electronics are kept within a reasonable temperature range, 50 to 95°F, for example. Furthermore, the electronics are protected from humidity in the kiosk enclosure, which is airtight and lacking panel gaps.

Although the kiosk is primarily deployed in indoor facilities, a stable and reliable power supply was important. To accomplish this, the SCAN kiosk has a rechargeable battery supplying over two days of battery life, the internal microcontroller is configured to idle in light sleep mode, and the kiosk has a reliable power supply and battery charger when running on outlet power. Finally, a stable network connection was needed to facilitate communication between the kiosk and the database for real-time authentication and analytics display. For this reason, we chose a capable microcontroller with native support for WiFi.

4.2.5 Social and Political Constraints

Another constraint for this project involves the handling of user data. In our example use case of the UCF Recreation and Wellness Center, the kiosk and its associated database handles sensitive user data, including a user's name, ID number, password, student status, and more. To protect user data, security was of high importance in the design of the authentication process and database structure. For this reason, we chose to use hardware authentication via an NFC tag and reader, which provides an added level of security over traditional authentication methods due to the short range of NFC and the complexity of the password contained within the tag. By making security a priority in the design of the SCAN system, we protect user's right to privacy and adhere to local, state, and national data and privacy laws.

4.2.6 Ethical Constraints

Ethical considerations in designing an NFC-enabled kiosk include concerns like the privacy of user data, battery material sourcing, and electronic waste. Regarding data privacy, the kiosk must protect user data and work with information in good faith. Ensuring the kiosk complies with the privacy rights of individuals provides a more ethical final product. With this constraint, the kiosk must minimize the amount of private data collected, provide users with clear consent options, and implement security measures to protect private data from unauthorized access or misuse. For a university setting, transparency about data usage is especially important, as users deserve to know how and why their data is shared and stored.

Another primary ethical consideration involves the batteries used to power the kiosk. Many batteries come with significant ethical concerns. The metals required for lithium-ion batteries, such as cobalt, lithium, and manganese, are often concentrated in politically unstable regions and pose human safety and environmental risks. For example, 60% of all cobalt is mined from the Democratic Republic of the Congo (DRC), where human safety is neglected, militia and armed groups control mining operations, and the

U.S. Department of Labor believes child labor is involved in cobalt mining. [55] These ethical concerns put a moral responsibility on people purchasing batteries and those involved in the battery manufacturing process to consider alternatives, such as using certain suppliers or even different battery technologies when possible

In addition to the ethical concerns of battery material sourcing, the electronic footprint of the kiosk was considered. Like batteries, sourcing other components for the kiosk can involve resource-intensive and unethical processes. In addition to this, at the end of the kiosk's lifetime, the kiosk becomes electronic waste that must be handled ethically and properly. Due to this ethical constraint, to mitigate these concerns, we designed for longevity and considered components that have sustainability certifications.

4.2.7 Health and Safety Constraints

Health and safety was at the top of our priority lists when making this product, especially considering the intended number of users needing to interact with this system on such a frequent basis.

One of the top constraints regarding health and safety was that the design should be electrically safe. All electrical components and wires were safely secured within the system's housing, with an insulating barrier (PLA) to ensure no risk of electrical shock. The system is sealed so that electrical equipment is immune to light spills or environmental conditions, like rain or adverse weather conditions. If the system operates outdoors, these conditions could interfere with circuitry and potentially cause hazards. The electrical design also ensures proper heat dissipation to avoid burns. Low-power parts were chosen to prevent both electrical shock and burn hazards. Proper ventilation was also incorporated into the design for heat dissipation.

Another health and safety constraint is ergonomics. The kiosk is positioned at an accessible height and angle to ensure users of varying heights and physical abilities are able to properly use our system. This is especially important for SCAN, as users frequenting the gym interact with the kiosk very often, and repeated unergonomic movements could cause discomfort. We chose this height to be around four feet from the group to the top of the kiosk.

4.2.8 Manufacturability Constraints

The manufacturability of SCAN is influenced by several constraints related to part selection, PCB design, and assembly processes. Choosing components that are available and straightforward to integrate or assemble was necessary to meet deadlines and ensure reliability. For example, the processor, sensors, and other ICs we selected come in standard package sizes. Additionally, the PCB design for the kiosk adheres to standard sizes and materials, such as trace widths and board materials. Complex packages, custom parts, and irregular PCB design decision would have complicated assembly, introducing other issues. Manufacturability of the kiosk design was necessary to deliver a finished product.

The enclosure design also needed to consider several manufacturability constraints. When dealing with printing 3D parts, factors like print resolution, material selection, and 3D design needed to be considered. For example, higher print resolutions create more precise parts but increase printing time, therefore decreasing manufacturability. Every printing material also has its own unique properties, tolerances, and temperature requirements. While some are better suited for strength, others are faster and easier to print. Lastly, the size of the parts needed to be considered because it must be within the limits of the 3D printer bed size.

5. USE OF LARGE LANGUAGE MODELS IN REPORT WRITING

Large language models are advanced artificial intelligence systems designed to produce human-like text based on large datasets. These AI tools have taken the world by storm in recent years due to advances in natural language processing (NLP)—the enabling machine learning technique behind LLMs—and computing hardware. LLMs and LLM-powered chatbots have become extremely popular with millions of daily users. ChatGPT, one of the earliest and most successful LLM-based chatbots today, acquired its first million users in the first five days after being released in November 2022. This is a faster adoption rate than nearly any other application or technology in history. In the context of senior design, LLMs are great tools for finding and synthesizing information quickly, but they should not be relied upon completely. [48]

5.1 Statement on Use of LLMs for Report Writing

We hereby declare that we have not copied information directly from Large Language Models (LLMs). Any writing in this report is the original work of its authors, and areas where sources and/or LLMs have been used are noted and cited. During the writing of this report, we utilized LLM resources such as ChatGPT, Perplexity, and GitHub Copilot for drafting, outlining, comparing, summarizing, and proofreading only.

5.2 Large Language Models

An LLM's ability to take plain human input text and generate unique and particular output responses based on trained datasets makes them incredibly useful for synthesizing and outputting information. For many, this practically replaces the need to use standard search engines for general information retrieval. Along with text generation, LLMs excel in summarizing text, correcting grammatical errors, and writing/executing code.

In the context of senior design, LLMs are a powerful tool for streamlining research, aiding in documentation, providing technical insights, and debugging code snippets. They have the potential to enhance our productivity by automating many of the time-consuming and routine tasks associated with senior design, including researching various technologies, proofreading our report drafts, and providing high-level details for how to implement our final design. It is important, however, to acknowledge the limitations of existing LLMs, which is what this section of the report hopes to accomplish.

5.2.1 ChatGPT

OpenAI's ChatGPT is one of the most widely known and used LLMs available today. It is built using the generative-pre-trained transformer (GPT), which enables it to generate human-like text. [49] ChatGPT is trained by OpenAI on websites, books, articles, and web pages like Wikipedia, meaning it has a large base background of data, helping it to more accurately use prediction to generate precise and targeted text outputs. Using this

information, ChatGPT can assist with a myriad of tasks like drafting reports, explaining technical concepts, and even generating code. Its capability to build off of this data makes it a quintessential tool for quickly and efficiently researching and designing our senior project idea.

Although ChatGPT is a powerful tool for learning, writing, and generating creative content, it suffers from a few negative aspects that should be considered when making use of this tool. One of the key downsides of ChatGPT is its "hallucinations," where the model will generate plausible seeming and yet incorrect outputs. This is particularly relevant when it comes to ultra-specific queries and prompts as the model is less trained in these areas. Therefore, its text prediction methods tend to work against the user, generating convincing yet false outputs. This issue is unique to certain language models like ChatGPT in that it tends to only occur in models with limited access to information. In models with access to real-time data, this is a less likely occurrence. Overall, ChatGPT is a very strong large language model, particularly capable in areas like writing, technical explanation, and code generation.

5.2.2 Perplexity

The release of LLMs has spawned countless more AI applications, which integrate LLMs into their products and services for additional functionality. This can be seen in nearly all major search engines and social media platforms, including Google, Bing, and Instagram, which all have AI assistants powered by Gemini, Microsoft Copilot, and Meta's LLaMA, respectively. An issue pervasive in LLMs and all of these AI-enabled applications is that of hallucinations and providing factually inaccurate information. Perplexity is an LLM-powered search engine that aims to solve this issue by providing citations within the text response. Users on the free version of Perplexity are able to receive responses from Perplexity's own proprietary LLM, while Perplexity Pro users are able to select between various LLM backends such as GPT-4o, Claude 3.5, Mistral Large, and Llama 3. By selecting various backends, users are able to utilize the strengths of different LLMs for different applications while still receiving cited information. [50]

5.3 Strengths of LLMs

Overall, LLMs have three primary strengths: their natural language processing ability, efficiency in synthesizing information, and accessibility.

Natural language processing allows LLMs to understand and converse in human-readable text. This ability makes LLMs incredibly useful for writing, documenting, and even generating code. One of the most powerful ways this strength can be used is in simplifying complex topics. This ability is particularly useful for senior design since we can form complex prompts and get unique and targeted output explained to us rather than spending lots of time searching the internet.

LLM's time savings brings us to the next strength of LLMs, which is efficiency. LLMs are incredibly efficient at sifting through and synthesizing large datasets, like thousands of online resources. This means it can directly answer a specific prompt rather than a user

needing to perform multiple web engine searches and sift through websites and other resources manually to find specific information.

Finally, one of the greatest strengths of LLMs is their accessibility. Because of LLM's ability to synthetically understand and process human-readable text, they are extremely easy to use. Gone are the days when users must carefully craft searches to find exactly what they're looking for on the internet. LLMs are able to synthesize all of these sources and find the data most relevant to you. LLMs are also accessible through many different means, even now showing up on the iPhone. They can be accessed through APIs and have applications in nearly every field.

5.4 Limitations of LLMs

While LLMs can be incredibly useful, especially in the context of senior design, they have a few limitations that should be taken into account as well. One of these limitations is factual inaccuracy. As discussed earlier, even though LLMs are trained on a large set of data from many sources, they aren't trained on everything and are likely to be less accurate regarding very niche topics. Since LLMs use predictive text models, the agent is likely to output persuasive yet false information. This is especially dangerous in the context of senior design, and it should be taken into account that not all information generated by LLMs will be correct. One solution to this potentially inaccurate information is to use Perplexity, which provides citations to all websites used to generate a given response. Furthermore, there are filters that can be applied to restrict the types of sources used. If a specific website is known to be unreliable or provides inaccurate information, it can be removed from the source list, and a new response can be generated.

Another issue that many users of LLMs have likely encountered is a lack of contextual understanding. LLMs tend to "forget" the context of conversations, especially over longer conversations and may need to be refreshed. This issue is one that is being addressed with newer models of LLMs but is still a big concern to many. With senior design, it is important to remind the LLM of the overall context of our project as we have a large number of components.

One of the biggest flaws/limitations of LLMs is their limited access to training data. Each model is trained using varying sources, and there are many forms of media that cannot be accessed by LLMs for training either because of modality, as in maybe a textbook only distributed via hardcopy or legality reasons. Another big factor in this is that LLMs do not have access to real-time data. This means they are, in a way, stuck in the past, limited by when they are trained on more contemporary data.

5.5 LLM Case Study

To demonstrate the effectiveness and limitations of AI tools in report writing, we will consider a few example prompts and their corresponding responses in this section. These prompts are associated with the selection of a microcontroller for our project and are engineered to test LLM reasoning in comparing different options. To ensure that the information provided is backed up by sources, we use Perplexity's search engine with

GPT 4o as the backend LLM. The hyperlinks in each response link to the websites that Perplexity pulled information from. The LLM responses in this section have been shortened for brevity, but the full responses can be found in Appendix E.

Case Study 1: "Which WiFi-capable microcontroller should I select to interface with peripherals such as an NFC reader and TFT display for my senior design project?"

> "When selecting a WiFi-capable microcontroller to interface with peripherals like an NFC reader and a TFT display for a senior design project, consider the following options:
> *ESP32*
> - Features: The ESP32 is a popular choice for IoT projects due to its built-in WiFi and Bluetooth capabilities. It has dual-core processing power, which is beneficial for handling multiple tasks such as reading NFC data and updating a TFT display simultaneously[2][4].
> - Compatibility: There is a wide range of libraries available for interfacing with NFC readers and TFT displays, making it easier to develop…
> - Cost: The ESP32 is cost-effective and widely supported by a large community, providing plenty of resources and examples[5].
>
> *STM32F407*
> - Features: The STM32F407 offers more I/O pins and better analog-to-digital conversion compared to the ESP32. It is suitable for projects requiring extensive peripheral connections…
> - Performance: This microcontroller can handle more complex tasks and offers better documentation for low-level programming, which might be beneficial if your project demands precise control over peripherals[4].
> - Considerations: While it does not have built-in WiFi, you can add this capability using external modules…
>
> RA6M3
> - Features: The RA6M3 from Renesas includes a dedicated TFT controller and is designed for IoT applications. It supports high-speed USB and Ethernet, which could be advantageous if your project involves high data throughput[3].
> - Ecosystem: Supported by the Flexible Software Package (FSP) and FreeRTOS, it offers a flexible development environment. This might be beneficial if your project requires advanced features like security or large RAM usage[3].
>
> *Recommendation*
> For most student projects focusing on IoT with moderate complexity, the ESP32 is likely the best choice due to its integrated WiFi, ease of use, and strong community support. If your project demands more I/O flexibility or specific performance characteristics, consider the STM32F407 or RA6M3…"

The response given by Perplexity recommends three different microcontrollers and lists the basic features of each device. Interestingly, perplexity recommended the

STM32F407, which does not natively support WiFi, one of the constraints in our prompt. It also provided different bulleted information for each microcontroller. For the ESP32, it wrote sections on features, compatibility, and cost. For the STM32F407, it wrote sections on features, performance, and considerations. Finally, for the RA6M3, it wrote sections on features and ecosystems. With different features being described, it is difficult to provide a one-to-one comparison for each device.

Another important detail to note is that many of the sources cited by Perplexity in its response are forum posts and Reddit articles. While these articles include valuable knowledge, they might not be reliable or credible enough to be used in a professional design report. This can be mitigated by using the "remove source" feature to remove any unreliable sources and regenerate the response. Perplexity did, however, identify many credible sources, including the Adafruit website, NXP, and the official Renesas page for the RA6M3. Ultimately, Perplexity determined that the ESP32 is the ideal microcontroller for this project, which was the same conclusion we drew from our technology review and component selection.

Case Study 2: "Using the same microcontrollers as before, generate a table comparing features including but not limited to the number of IOs, memory sizes, and operating voltage of each MCU."

Table 5.1 MCU Comparison by Perplexity

Microcontroller	**Number of I/O Pins**	**Flash Memory (MB)**	**SRAM (KB)**	**Operating Voltage (V)**
ESP32	36[42]	4[43]	520[44]	2.2 - 3.6 V[45]
STM32F407VGT6	100[46]	1[46]	19246	2.0 - 3.6V[46]
RA6M3	-	2[47]	640[47]	2.7 - 3.6 V[48]

As seen from **Table 5.1**, Perplexity correctly identified each requested feature, located the specifications, and summarized the results in a table. Interestingly, although we gave Perplexity the freedom to include additional features in the table, it limited the response to only the requested features. Furthermore, it was unable to locate the number of IO pins for the RA6M3, even though it was clearly listed in source 48.

The included information for each MCU is accurate to the datasheets, except for the number of IO pins on the ESP32. The linked sources for the ESP32 refer to the development board using the MCU's IC. Meanwhile, the information for the other two MCUs is based on their respective ICs. This inconsistency makes direct comparisons between the three microcontrollers more difficult.

Case Study 3: A solution to this problem is more efficient and explicit prompt engineering. By providing more information in the prompt such as "Directly compare the ICs", "Use the datasheet for each IC as a source", and "Include three additional key

features of each IC", the underlying LLM can have more information and context to work with when generating a response. By adding this additional context, **Table 5.2** was generated by Perplexity.

Table 5.2 Expanded MCU Comparison by Perplexity

MCU	# of I/Os	Memory Sizes	Operating Voltage (V)	Processor Speed (MHz)	Power Consumption	Supported Comm. Protocols
ESP32	34[7]	Flash: 4096 KB, SRAM: 520 KB[8]	2.7V - 3.6V[7]	240 MHz[9]	Active: 160 mA; Sleep: varies[10]	UART, I2C, SPI, Bluetooth, Wi-Fi[11]
STM32 F407	100[13]	Flash: 1024 KB, SRAM: 192 KB[14]	1.8 V to 3.6 V[14]	168 MHz[13]	Active: 238 µA/MHz, Sleep: unknown[14]	UART, SPI, I2C, USB, CAN[13]
RA6M3	176[15]	Flash: 2048 KB, SRAM: 640 KB[15]	2.7V to 5.5V[15]	120 MHz[15]	Active: 66 mA, Sleep: 1.5 uA[15]	USB, Ethernet, CAN, UART, I2C[15]

This table is much improved, citing data from the ESP32 IC's datasheet and including additional features such as processor speed, power consumption, and supported communication protocols. It also correctly identifies the number of IO pins on the RA6M3. While the validity of all these metrics must still be verified, the inclusion of links to each datasheet streamlines the comparison process between each microcontroller IC, enhancing the productivity of component selection.

6. HARDWARE DESIGN

This chapter details the hardware design-related aspects of the SCAN project. It begins with a block diagram of the full schematic for the SCAN system, presented in Figure 6.1, which highlights the power, input, and output connections to the ESP32-C6. Following this, this chapter provides an in-depth view of the full system schematic and its sub-schematics, including those for the power delivery sink controller, power management controller, automatic boot mode, the NFC transceiver, and all other relevant peripheral devices. Additionally, this chapter discusses the hardware interfaces between components, focusing on the implementation of SPI and I2C communication protocols.

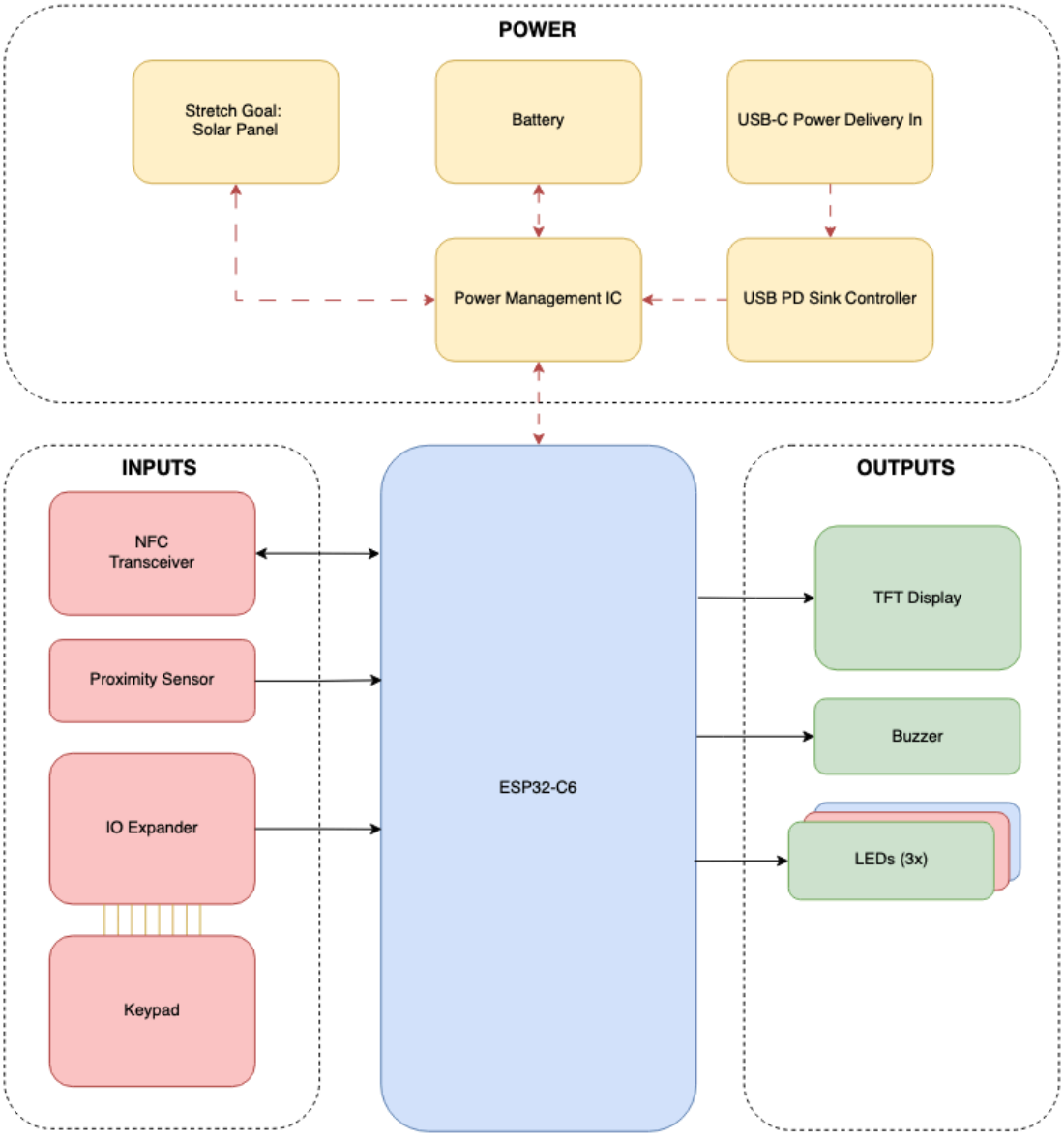


Figure 6.1 SCAN Schematic Block Diagram

A top-level view of our full schematic design is shown in Figure 6.2. Using KiCad, we took a hierarchical approach to designing this schematic, linking portions of the schematic such as the USB Power Delivery system, USB to UART Bridge, and PN532 transceiver circuits designed on other pages to the root schematic using hierarchical labels. A high-resolution copy of the schematic in its entirety can be found in Appendix E.

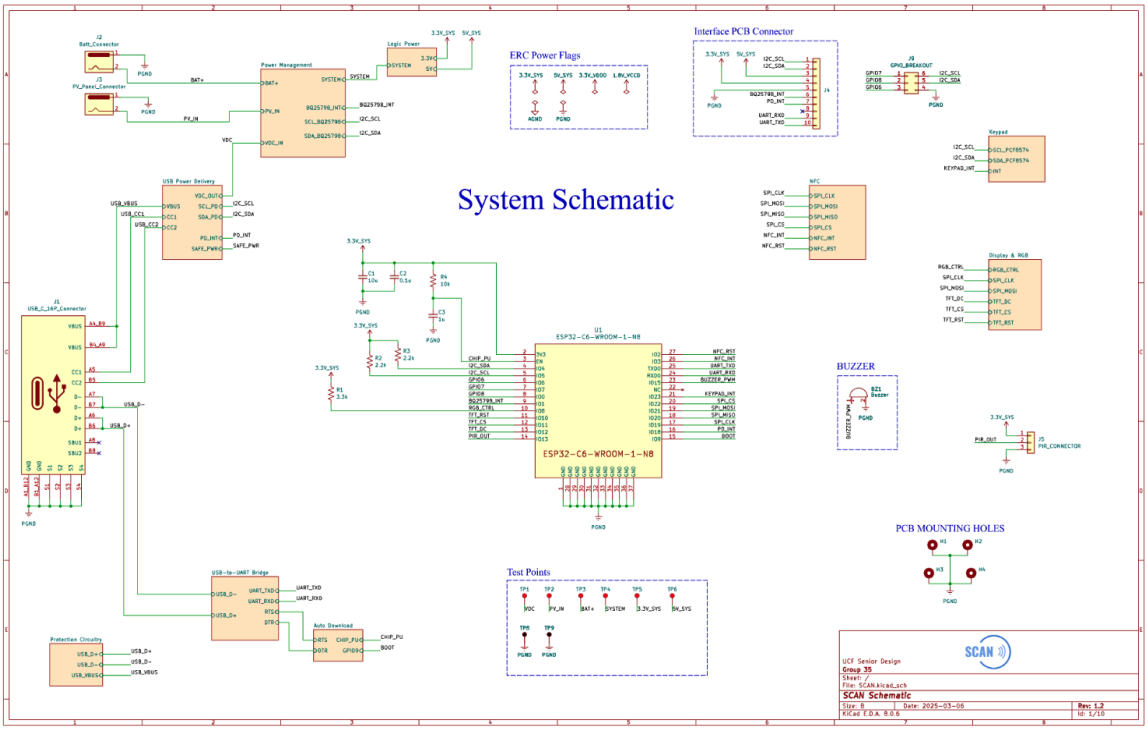


Figure 6.2 SCAN System Schematic

The figure above lays out both hierarchical blocks and specific components. Starting from the left side, we have the USB-C connector and DC battery connector (at the top). These devices are connected to a network of hierarchical schematic blocks that handle everything from powering the kiosk to flashing the ESP32. The ESP32 is located in the middle of the figure, with connections between all major components in the sheet, either through direct wires or net labels. At the bottom of the schematic are LEDs for debugging and displaying check-in status to users.

Moving on to the right side of the schematic, we have the TFT display, NFC reader circuit (PN532), header pins to probe nets and connect the PIR sensor, and bus expanders for the keypad and various reset signals. One important schematic block in particular is the PN532, which includes the circuitry for the NFC transceiver and the built-in PCB antenna. In the following sections, we break down each subsection of the SCAN schematic and explain how they interface with the ESP32-C6.

6.1.1 Power Supply Design

To achieve SCAN's ease-of-use and power system goals, and ensure consistent and reliable operation across various scenarios, the design incorporates an intricate power management system. Figure 6.3 shows a high-level block diagram of the power supply design. The system integrates two key components, the CYPD3177 USB Power Delivery Sink Controller for USB-C power input and the BQ25798 for power management and battery integration. The following sections detail the functionality of the USB PD sink controller and power management controller.

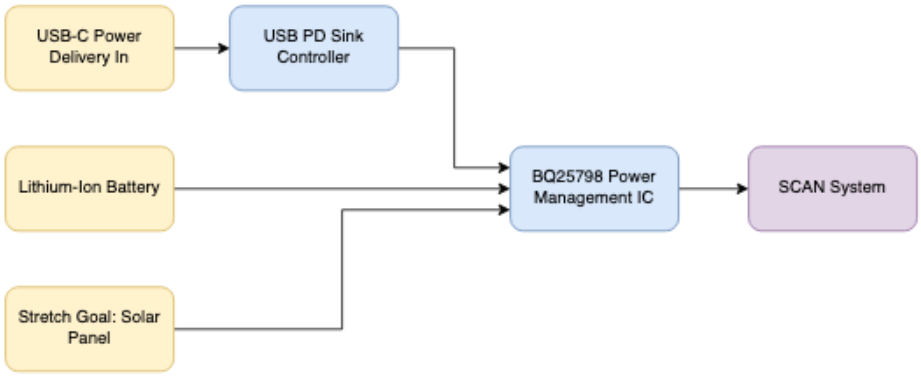


Figure 6.3 Power Delivery Block Diagram

6.1.1.1 USB Power Delivery Sink Controller

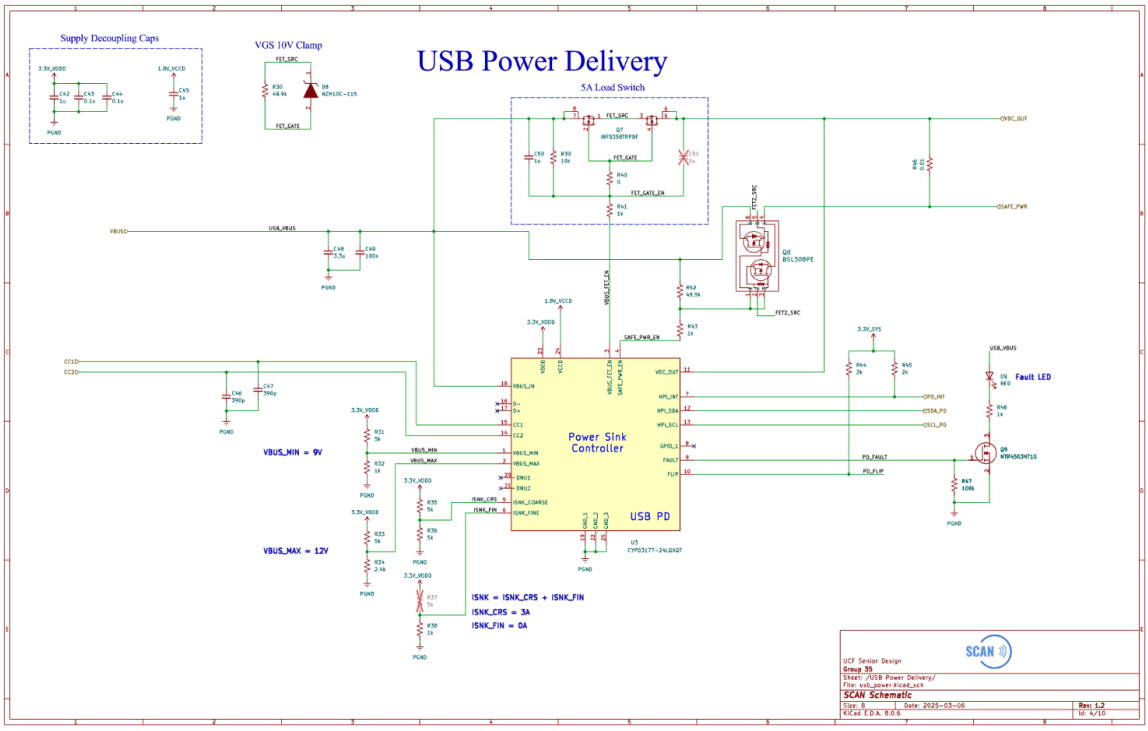


Figure 6.4 USB Power Delivery Schematic

The Infineon CYPD3177 integrated circuit serves as the primary interface for power input from the USB-C source. Figure 6.4 shows the schematic design surrounding the chip, which is based on the datasheet reference schematic provided in Appendix D. Inputs are routed in via the 16-pin USB-C connector, shown in the system-level schematic in Figure 6.2. Of these inputs, we first have four power delivery pins for VBUS. This signal path is the main power delivery route. Depending on the Power Delivery (PD) contract, the VBUS path can normally support up to 100 W (20V at 5A) when used with the CYPD3177. It is important to note a few things. Since USB-C is fully reversible and bi-directional, the pins on each side of the connector must be repeated on the other side. For the VBUS pins, this means each side of the connector (top and bottom) has 2 VBUS pins. Another thing to note is that the VBUS pins can still carry current when no contracts are available from the USB-C device. This enables additional devices to be used with a USB-A-to-USB-C cable. Moving down the USB-C connector, we have the CC1 and CC2 pins connected to the power sink controller that manage the power negotiation as described earlier. Lastly, we have the data and ground pins. The data pins are used to flash the ESP32 through a USB-to-UART bridge described in a later section, while the ground pins are all connected to the system ground.

This design also includes a few significant features. First off, each USB-C signal path is protected from ESD, noise, and transients using appropriate TVS diodes and capacitors. Each TVS diode is specifically selected based on the voltage and current traveling on that path. Additionally, the CYPD3177 features a set of resistor divider networks to set the default minimum and maximum voltage and currents requested from the connected USB-C device. Once a specific PD contract is negotiated, the CYPD3177 outputs a gate signal that is used to turn on the output power. Since this signal is on the high side (between the power supply and the load), instead of using a schottky diode, P-channel MOSFETs are used with their source pins tied to the power (VBUS) rail. Dual P-channel MOSFETs are necessary to prevent current flow from the source to drain when the FET is not conducting. Lastly, the CYPD3177 includes fault, interrupt, GPIO, and I2C pins. The combination of these pins provides a comprehensive system for controlling the USB-C input power. For example, the I2C interface can be used to directly monitor and tune operational parameters, while the fault pin can be especially useful for testing and integration.

6.1.1.2 Power Management Controller and Buck Converter

The BQ25798 by Texas Instruments is the main “brain” to the kiosk’s battery management system, managing and regulating the USB Power Delivery voltage for the $LiFePO_4$ battery. Figure 6.5 shows the surrounding circuitry for the controller, based on the datasheet reference schematic shown in Appendix D. This IC operates in buck-boost mode to convert the variable USB-C voltage into a steady, yet reconfigurable 8.4 V for the 2s $LiFePO_4$ battery. Regarding the output power port, the flexibility of the BQ25798 allows it to seamlessly transition between USB-C input power, battery power, or a combination of both, ensuring uninterrupted operation even in varying power conditions.

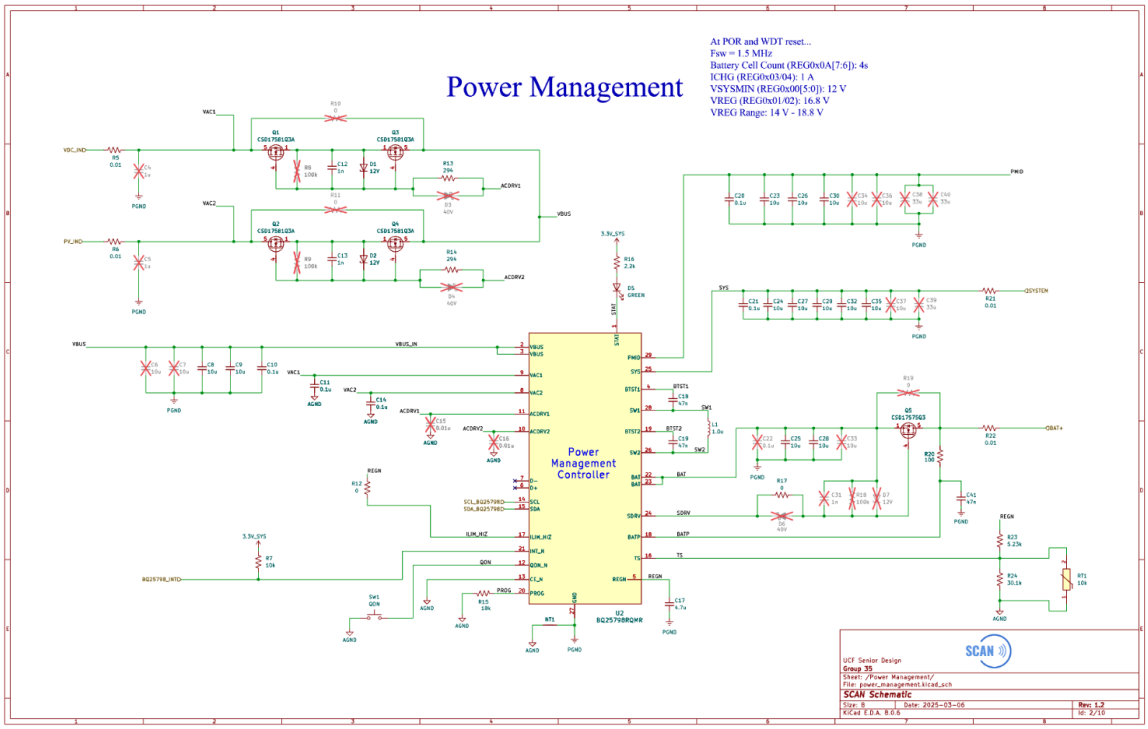


Figure 6.5 Power Management Schematic

Although the controller provides an uninterrupted output power port, The BQ25798 is mainly selected to manage the charging of the kiosk's $LiFePO_4$ battery pack. The IC connects to the battery terminals, and the charging current is programmed via an external sense resistor. The input voltage for charging is supplied from the USB-C source through the CYPD3177. Each battery charging cycle includes different phases necessary for safe $LiFePO_4$ charging, including trickle charging, pre-charging, constant current (CC) charging, and constant voltage (CV) charging. To minimize leakage current, the IC uses a MOSFET to disconnect the battery when it isn't being used. Its various integrated protections and monitoring capabilities are accessed through the I2C interface, allowing precise control over the charging process.

Lastly, an important feature of this controller is its built-in maximum power point tracking (MPPT) algorithm and dual-input capability. Although Figure 6.5 currently shows only USB-C power and battery power, one of SCAN's stretch goals was to include a solar panel. Fortunately, the BQ25798 is fully equipped out of the box to handle a small photovoltaic panel up to 24 V. The MPPT algorithm ensures that the solar panel operates at its optimal power point (based on the open-circuit voltage and other parameters), maximizing energy extraction even under varying light conditions. This feature not only added flexibility to our design but also aligned with our goal of making the kiosk adaptable to diverse use-cases.

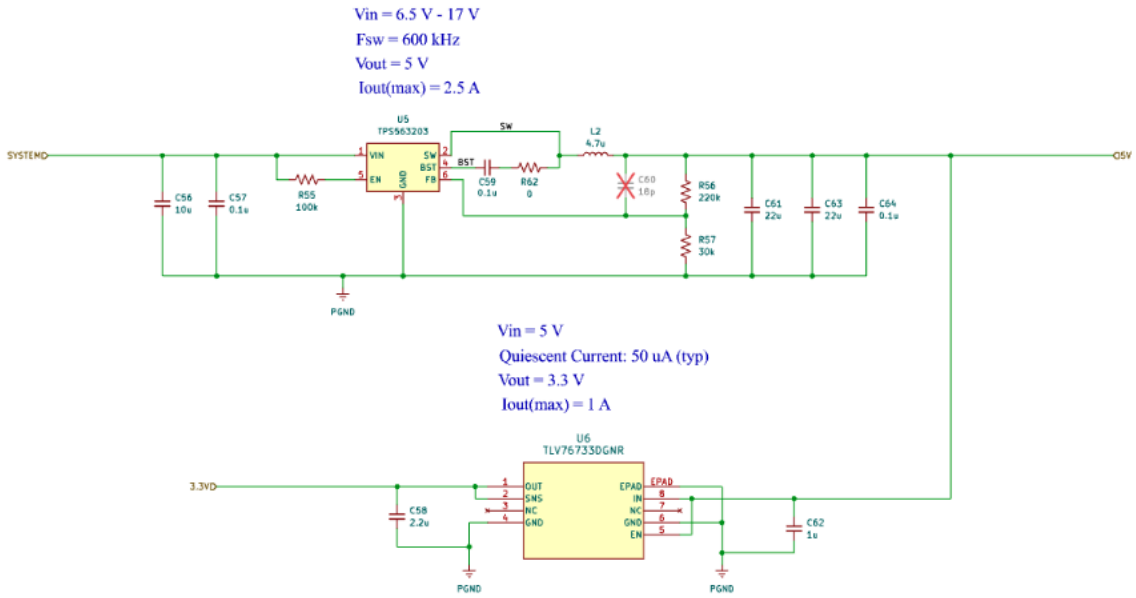


Figure 6.6 Logic Power Schematic

Following the BQ25798, the uninterrupted power output is fed into a buck converter IC (LM62460-Q1). The converter can handle an input voltage range of 3 V to 36 V, meaning it can handle all possible input voltage levels from USB-C. Using the datasheet schematic provided in Appendix D, the surrounding circuitry is designed to output 3.3 V at 400 kHz switching frequency with up to 6 A. From there, the 3.3 V is distributed to the kiosk system. This takes care of all power requirements for the SCAN system and allows flexibility in the future if other supply voltage levels are required.

6.1.1.1 Senior Design II Updates

Initially, we had decided on using the LM62460 dual buck converter to output 5V and 3.3V. However, during Senior Design II, we switched this with a 5V buck converter, the TPS 563203, and a 3.3V linear regulator, the TLV76733. The main reason for this was the high complexity of the original device. If the LM62460 did not work, we would have had to redesign the whole PCB, and since both voltage outputs relied on that one device, any issues would have required major changes. Splitting the design into two simpler devices with common footprints made the design easier to troubleshoot and fix if needed.

6.1.2 Automatic Boot Mode for ESP32

The schematic for the ESP32 automatic boot mode sequence facilitates a seamless hardware configuration to enable flashing and boot mode operations. It relies on certain ESP32 pins being asserted, which is initiated by the USB command sequence.

The design shown in Figure 6.7 integrates a CP2102N USB-to-UART bridge chip for UART communication based on Espressif's development board schematic shown in Appendix D. The USB data signals come in from the USB-C connector, meaning we can

use the same USB-C port for powering the kiosk, charging the battery, and programming the ESP32, significantly reducing complexity for users. The UART bridge chip interfaces with the ESP32 through DTR (Data Terminal Ready) and RTS (Request to Send) signals.

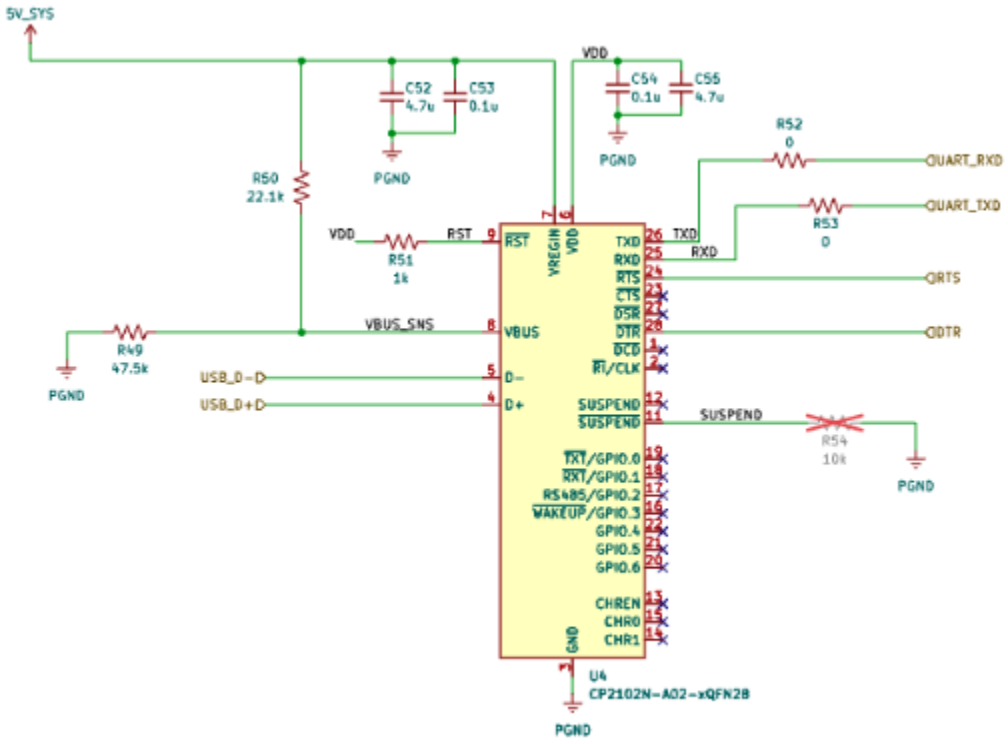


Figure 6.7 ESP32 USB-to-UART Bridge

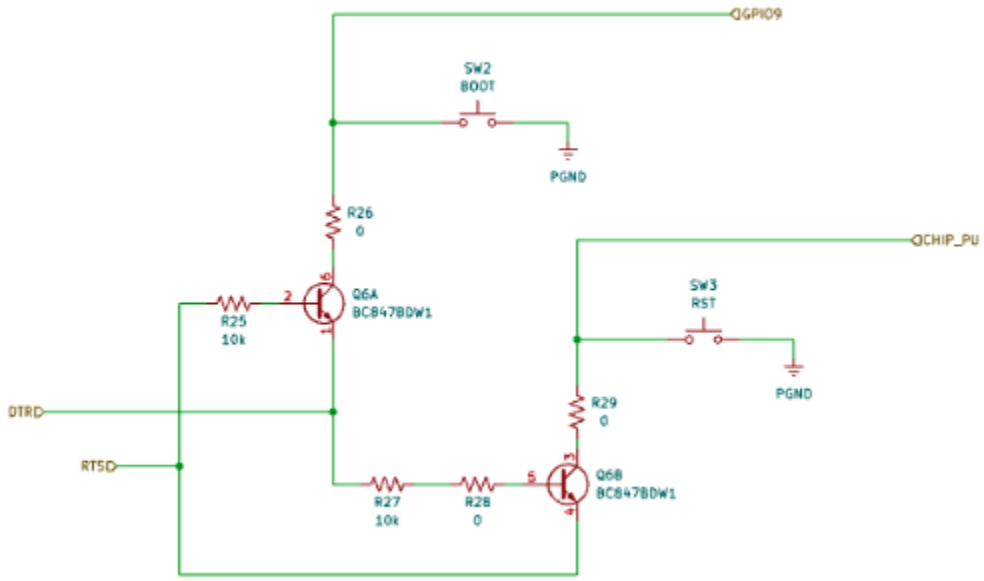


Figure 6.8 ESP32 Auto Downloader

Similarly, based on Espressif's development board schematic in Appendix D, these signals from the bridge are routed through a resistor network and a dual NPN transistor array in Figure 6.8, to control the CHIP_PU (chip enable) and reset pins on the ESP32. The resistors in the circuit above act as pull-up resistors to stabilize the control lines when signals are not actively driven. This hardware setup streamlines development by enabling the automatic programming of the ESP32 without manual intervention, reducing the complexity for developers during firmware flashing and debugging processes.

6.1.3 NFC Transceiver Schematic Design

For SCAN's NFC transceiver, we have chosen NXP's PN532. This is a highly integrated module with support for six operating modes, three host interfaces, and numerous RFID and NFC standard protocols. We have selected an NFC development module from Elechouse, which provides convenient GPIO interfaces for all features of the PN532 as well as an integrated antenna. The board's power input can support a voltage of 3.3-5V, and it has an on-board level shifter to convert between 3.3 and 5V logic levels.

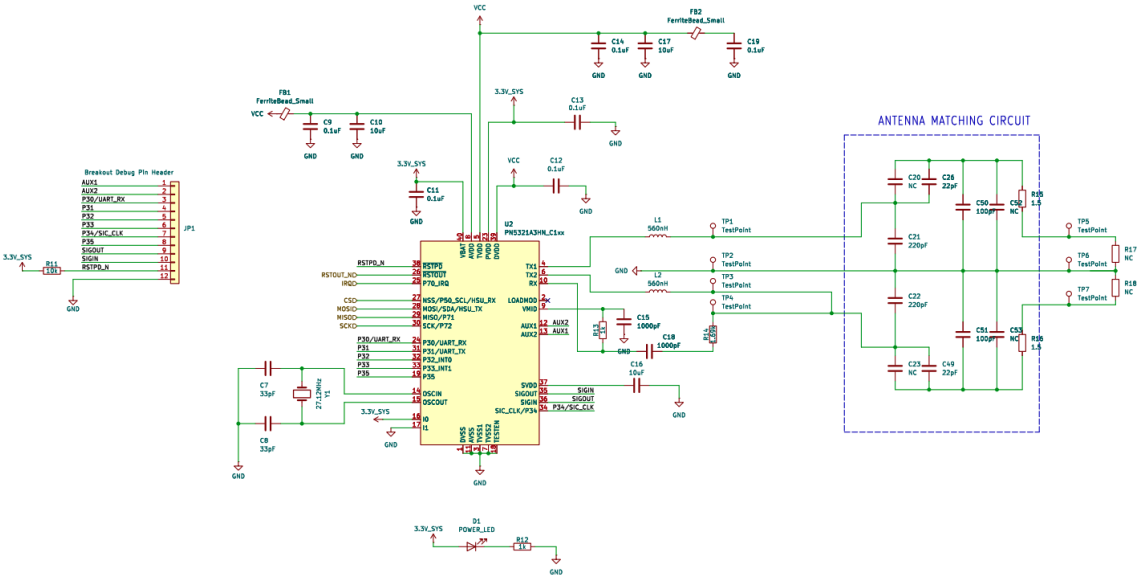


Figure 6.9 NFC Transceiver Schematic Design (Referenced from Adafruit)

Figure 6.9 shows the schematic for the PN532 transceiver circuit. The primary voltage supply to the PN532 is 3.3V from the PCB's main regulated supply. The other voltage level, VCC, represents the regulated voltage generated by the PN532's internal low drop-out (LDO) voltage regulator, which is used to supply SVDD, DVDD, AVDD, and TVDD. The last three voltage inputs must be separately decoupled with capacitors. This subcircuit is interfaced with the ESP32-C6 via a 4-pin SPI connection, active-low interrupt line, and active-low reset line. A breakout pin header is used to expose pins from the PN532 for debugging and testing purposes. Probe points are also located around the schematic for testing purposes after the board is manufactured. Finally, a 27.12 MHz

crystal oscillator is included to provide a stable clock for the low-power microcontroller internal to the PN532.

As mentioned previously, the PN532 supports three host device interfaces: I2C, SPI, and high-speed UART (HSU). We have chosen to interface the PN532 and ESP32-C6 using SPI for its higher data rates compared to I2C. The device-host interface for the PN532 is configured using the PN532's Multi-InterFace (MIF) register, which is set using pins I0 and I1 on the PN532 package. The bit values for each select line can be seen in Table 6.1 from the PN532 datasheet. Although the Adafruit reference design includes support for HSU, I2C, and SPI by manually configuring pin headers via jumpers, we only require the SPI interface.

Table 6.1 Host Interface Selection Bits (Taken from PN532 Datasheet)

Selif[1:0]	00	01	10	11
Package pin number	HSU	SPI	I2C	Reserved
27	HSU_RX	NSS	P50_SCL	-
28	HSU_TX	MOSI	SDA	-
29	P71	MISO	P71	-
30	P72	SCK	P72	-

6.1.3.1 Senior Design II Updates

Initially, we had hoped to incorporate the circuitry for the PN532 directly onto our SCAN PCB. However, due to the group's inexperience with RF antenna PCB design and a desire to focus on the complexity of SCAN's power management subsystem, we decided to instead use the off-the-shelf development module for the PN532. One of our review committee members also advised us against going down the route of a custom antenna PCB design.

6.1.4 Display Design

We utilized a TFT display with the GC9A01A graphics driver mounted on a development board. We will transfer the associated circuitry for the display development board to our PCB, which will connect to the display via a ribbon cable. The display circuitry will interface with the ESP32-C6 via 3-pin SPI, a reset signal, and a data control signal, in addition to 3.3V power and GND. To design the schematic for the display interface, we used a reference schematic found for the GC9A01 online. This schematic is located in Appendix D. Interestingly, although the display supports a variable backlight through LEDA and LEDK, our specific display hard-wires this backlight.

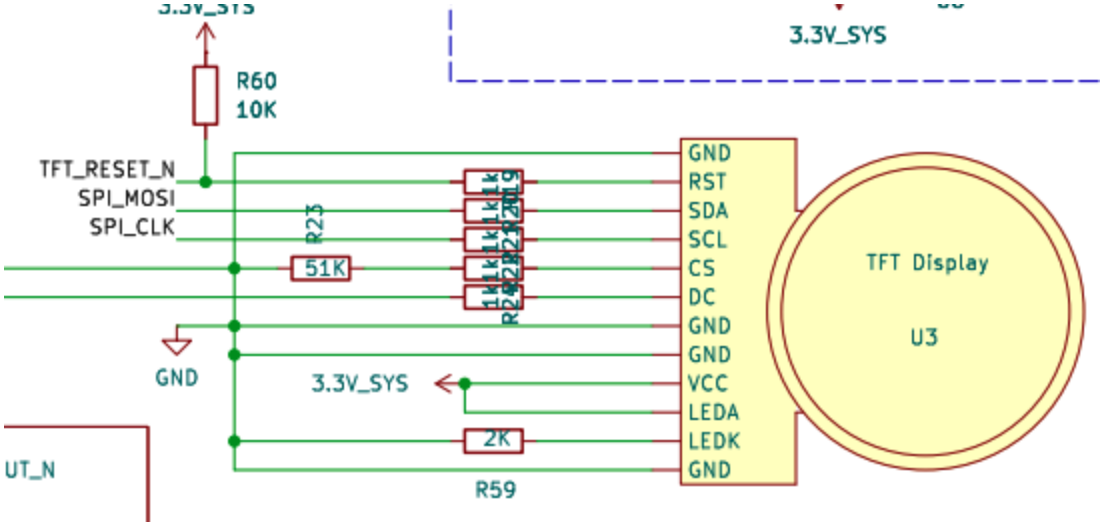


Figure 6.10 TFT Display Schematic

While the ESP32-C6 supports multiple SPI buses, we are constrained by the number of available GPIO pins on the MCU package. Therefore, we utilized a shared SPI bus between the controller device (the ESP32-C6) and two peripheral devices (the NFC module and the TFT display). By using the shared SPI bus described in Figure 6.11, the TFT display and NFC transceiver can share the serial clock (SCK) and MOSI lines and be addressed by their individual chip select (CS) lines. The display does not require a MISO line, as it is purely an output device and does not send any data back to the ESP32-C6.

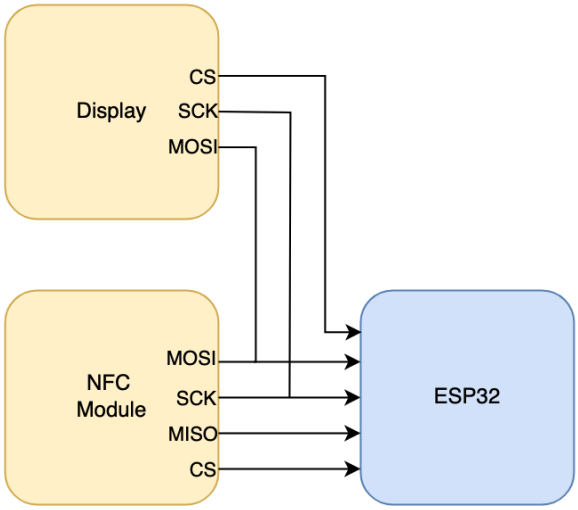


Figure 6.11 Shared SPI Bus Interface

6.1.5 Keypad and IO Bus Expander

To directly interface with the ESP32-C6, our 4x4 keypad would require access to 8 GPIO pins—each row and column in the 4x4 key matrix requires a dedicated GPIO pin. As the available number of GPIOs on our MCU was already limited, we instead opted to use an

IO bus expander. For our project, we selected the PCF8574 8-bit IO bus expander, which can address up to 8 IO pins via I2C. The PCF8574 uses 7-bit I2C addressing and has an address range from 0x20 to 0x27. This address can be set by tying pins A0-A2 to GND for an I2C address of 0x20, tying A0-A2 to VDD for an address of 0x27, and so on. Figure 6..12 displays the fixed and configurable portions of the PCF8574 I2C address.

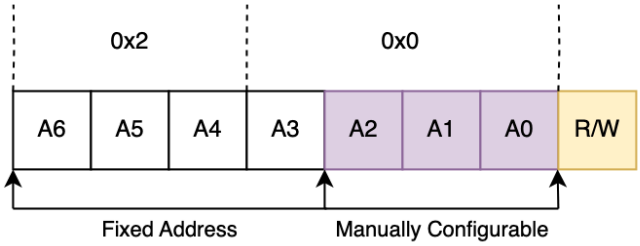


Figure 6.12 PCF8574 default I2C address configuration

In the schematic for the keypad interface, the IO pins of the keypad are connected to pins P0-P7 of the PCF8574. VDD is connected to the regulated 3.3V supply of our power management circuit, and VSS is connected to GND. The SDA and SCL lines are connected to the ESP32-C6's default I2C bus. The PCF8574 provides an open-drain interrupt output pin (INT), which can be connected to the interrupt input of the ESP32. It is pulled to HIGH and connected to GPIO3 of the ESP32-C6.

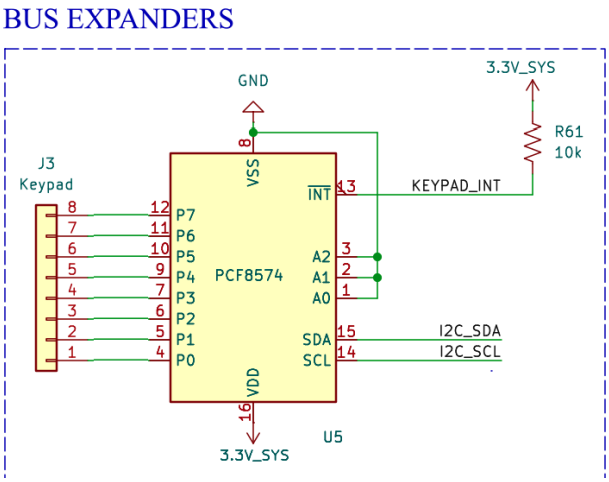


Figure 6.13 IO Bus Expanders Schematic

In addition to the IO bus expander used by the keypad, another expander is used to interface with the reset pins of the PN532 and display and also the red, green, and blue status indicator LEDs. Between the 8 GPIO pins of the keypad and 5 GPIO pins of the reset bus, this frees at least 11 GPIO pins on the ESP32-C6 for other functions. These devices are ideal to interface with I2C in comparison to SPI. Unlike peripherals such as the display and PN532, these devices would not benefit from SPI's higher data rates. Each I2C device has been configured to have its own I2C address, as seen in Figure 6.14.

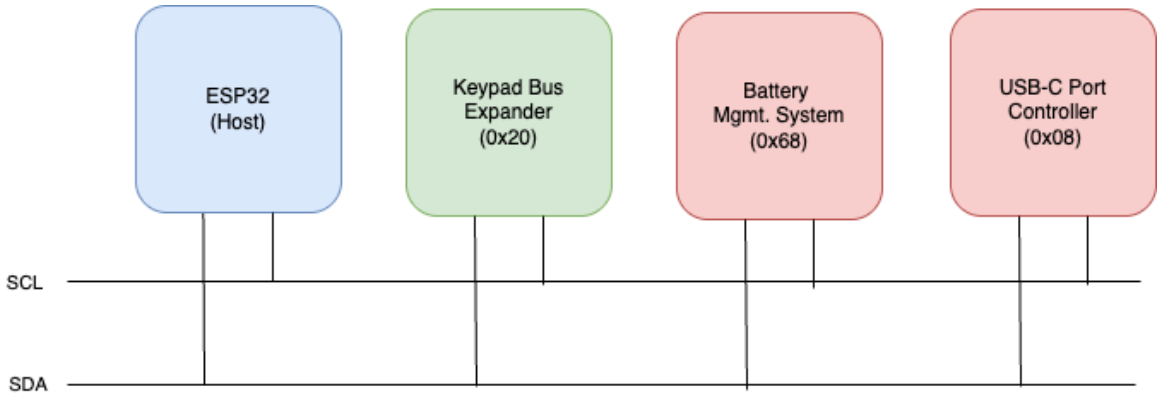


Figure 6.14 Shared I2C Bus Interface

6.1.6 Other Connections

In addition to the circuitry already discussed, the PIR sensor is connected to the PCB via a 1x03 pin header. The piezo buzzer is mounted on the PCB and interfaced with the ESP32-C6 via a PWM pin. Three status LEDs (for authentication success, error, and admin mode: see Chapter 7) are also mounted directly to the PCB with current-limiting resistors. Finally, the schematic contains four mounting points, which are grounded and used to attach the PCB to the SCAN enclosure.

Here, we choose to break out, or expose, certain signals to external jumper cables and connections for debugging purposes. GPIO16 and GPIO17 remain unused on the ESP32-C6, so we include them in the breakout header to connect any waveform analyzers, multimeters, or other testing equipment in the event of operational issues. Furthermore, we break out the I2C clock and data lines, reset, and boot signals to ensure that the proper signals are being transferred. The I2C pins can be used to add additional I2C devices to the bus, and the lines can be analyzed with an oscilloscope. Furthermore, the reset and boot signals can be manually pulled up/down for rebooting and entering boot mode for the ESP32 if needed.

For the power breakout header, we expose various signals to ensure proper voltage regulation and operation of the selected power electronics parts. In the case that the voltage levels are not generated as expected, external supplies can be connected. This would allow us to continue testing and integration of other functionalities while debugging the power circuits. We can also use these header pins to measure things like voltage drop, voltage regulation accuracy, and ripple and switching noise from the DC-DC converters. Additionally, schematic symbols are included for the PCB mounting holes that attach the PCB to the SCAN kiosk enclosure. These holes are grounded to protect the PCB against any electrostatic discharge.

6.1.7 Senior Design II Major Schematic Revisions

While the overall functionality and components selected for our schematic remain unchanged from SD1 to SD2, there were a few major changes. For one, we removed the

second IO expander for the reset bus and LEDs, as we opted to use addressable RGB LEDs that can be directly chained together, freeing up two GPIOs. We also decided against integrating the circuitry for the display module directly to our PCB, as it needed to be placed inside of our kiosk assembly and the FPC connector on the display would have been too short to reach our PCB.

7. SOFTWARE DESIGN

The software design of the SCAN system encompasses the embedded kiosk application, database, and frontend components that work together to provide a seamless user experience throughout the check-in process. By using modern development frameworks and tools, our software design ensures real-time responsiveness, secure data management, and an intuitive user interface. Through detailed diagrams and descriptions, this chapter outlines the technical architecture, libraries, and workflows of the SCAN system.

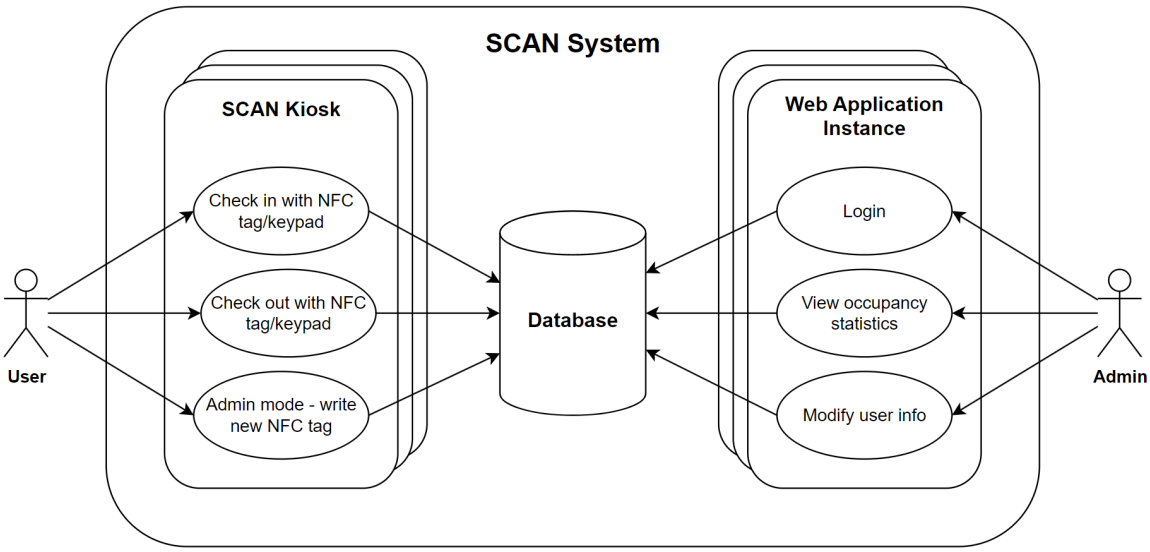


Figure 7.1 SCAN Use Case Diagram

The use case diagram in Figure 7.1 illustrates the interaction between SCAN's users and the underlying software components. The SCAN system is depicted using a large box encompassing two main subcomponents—the SCAN kiosk and web application instances—connected via a centralized database. Multiple instances of the SCAN kiosk and web application illustrate SCAN's ability to scale to numerous check-in kiosks or browser instances. Depicted to either side of the diagram are users and admins, who interact with these two main subcomponents via various functions. The "User" represents individuals, like students or faculty, using the SCAN kiosk to check in and out of the RWC, while an "Admin" represents system administrators, like RWC staff, who access system data and statistics through the web application. Admins are also able to write new ID cards using the SCAN system, which will be outlined in a later section.

The SCAN Kiosk serves as the physical interface for users, offering two prominent use cases for interaction: check-in and check-out. Checking in via keypad allows users to

initiate the check-in process by entering their UCF ID, whereas checking in or out via NFC tag offers a tap-to-authenticate check-in process as users can simply tap their NFC tag. Additionally, the kiosk gives admin users the ability to write new NFC tags. Each of these actions point directly to the database symbol, showing the interconnection between interactions on the physical hardware and cloud database.

The web application subcomponent provides administrative functions, allowing the admin actor to manage and monitor the SCAN system. The administrator can log in to the website, view occupancy metrics, and update user information. The login method allows admins to securely access the web application, granting them the ability to view a variety of graphically displayed live user statistics like occupancy counts and statistical measures for future planning and even the ability to modify user data, like names and check-in status. Similar to the SCAN kiosk, each action on the web application side of the SCAN system also points to the database, indicating that user log-in information, occupancy statistics, and user profiles are stored there.

7.1 Software Development Framework and Tools

The development of SCAN's software relies on modern and widely-used tools to write, test, and manage software. This section describes the key software frameworks used for the development of SCAN.

7.1.1 Visual Studio Code

Visual Studio Code is a widely used, lightweight source code editor that was chosen as the Integrated Development Environment of choice for SCAN's software systems. Many code editors exist that would suit our needs for this project, but VS Code stands above and beyond due to several features. For one, VS Code supports integrated debugging with breakpoints, variable inspection, and call stack navigation. It allows efficient navigation around even large code bases, where you can easily trace a function call to its declaration in a source library. VS Code also supports an expansive library of extensions, adding support for syntax highlighting in various languages used in our project, including C, C++, JavaScript, and more. Its real-time code assistance features make software development much more efficient. Finally, VS Code has a high degree of support for version control, which the SCAN team uses to facilitate the addition of new features, code revisions, and software integration.

7.1.2 GitHub

At the beginning of Senior Design 1, the SCAN team created a GitHub repository for the project to manage version control as we developed our software. GitHub's cloud-based repository system enables efficient team collaboration and keeps a consistent record of progress and changes as we develop SCAN's software. When a new feature is added, we can create a new branch to develop the feature independently of the main feature branch. By keeping a record of changes and commits, we were able to easily recover any lost work or roll back any unintended features at any point in the design process. By using

GitHub throughout SCAN's development, we had a transparent, organized, and reliable method of software development.

7.1.3 Docker Development Environment

Docker serves as the basis of our development environment, providing a standardized and isolated environment for the project. By containerizing the development environment, we ensure that every team member works with identical tools, libraries, and configurations, eliminating the "it works on my machine" problem. The Docker setup also integrates seamlessly with GitHub, enabling advanced workflows through Continuous Integration and Continuous Deployment (CI/CD) pipelines. These pipelines automatically build and test the firmware whenever new code is pushed to the repository, ensuring that all changes are validated before being merged into protected branches. This integration improves collaboration and minimizes the risk of broken builds in the main branch, maintaining a stable codebase while streamlining team workflows. Docker's portability also allows developers to onboard quickly without spending time setting up complex environments.

Since Docker containers operate in isolated environments, accessing physical hardware resources like USB COM ports directly is challenging. To address this, we implemented an RFC2217-compliant virtual COM port server to bridge the gap between the containerized environment and the host machine's serial interfaces. This server allows us to communicate with the ESP32 over a virtual serial connection, facilitating seamless flashing and debugging workflows. To enhance this capability, we developed a custom Python script to manage the RFC2217 server, enabling flexibility and precise control over the communication process. This reduces complexity for our team when we need to quickly develop code or flash the ESP32. By integrating this solution into our CI/CD pipeline, we can validate serial communication during automated tests and maintain efficient debugging capabilities. This setup ensures that even within a containerized development framework, we achieve smooth hardware interaction without compromising the benefits of Docker.

7.1.4 ESP-IDF Framework

Lastly, ESP-IDF is the official development framework for ESP32 devices and forms the basis for our software within Docker. Its expansive set of APIs and libraries simplifies the development of features essential to the project like I2C, SPI, and Wi-Fi communication. The framework also includes tools for configuring peripherals, writing custom drivers, and optimizing power management, which align perfectly with our hardware requirements, such as managing the LiFePO4 battery system and USB-C power input. ESP-IDF's integration with Docker enhances its usability in CI/CD pipelines, enabling automated builds and tests as part of the development process. This combination allows us to leverage ESP-IDF's extensive capabilities while maintaining consistency across all development environments, improving productivity and reducing debugging time.

7.2 Kiosk Software

In this section, we describe the technical details and implementation of each software component of the SCAN system. Here, we describe the functionality of each system, the libraries and frameworks used, software flowcharts, and other implementation-specific details of each system.

7.2.1 Kiosk User Mode

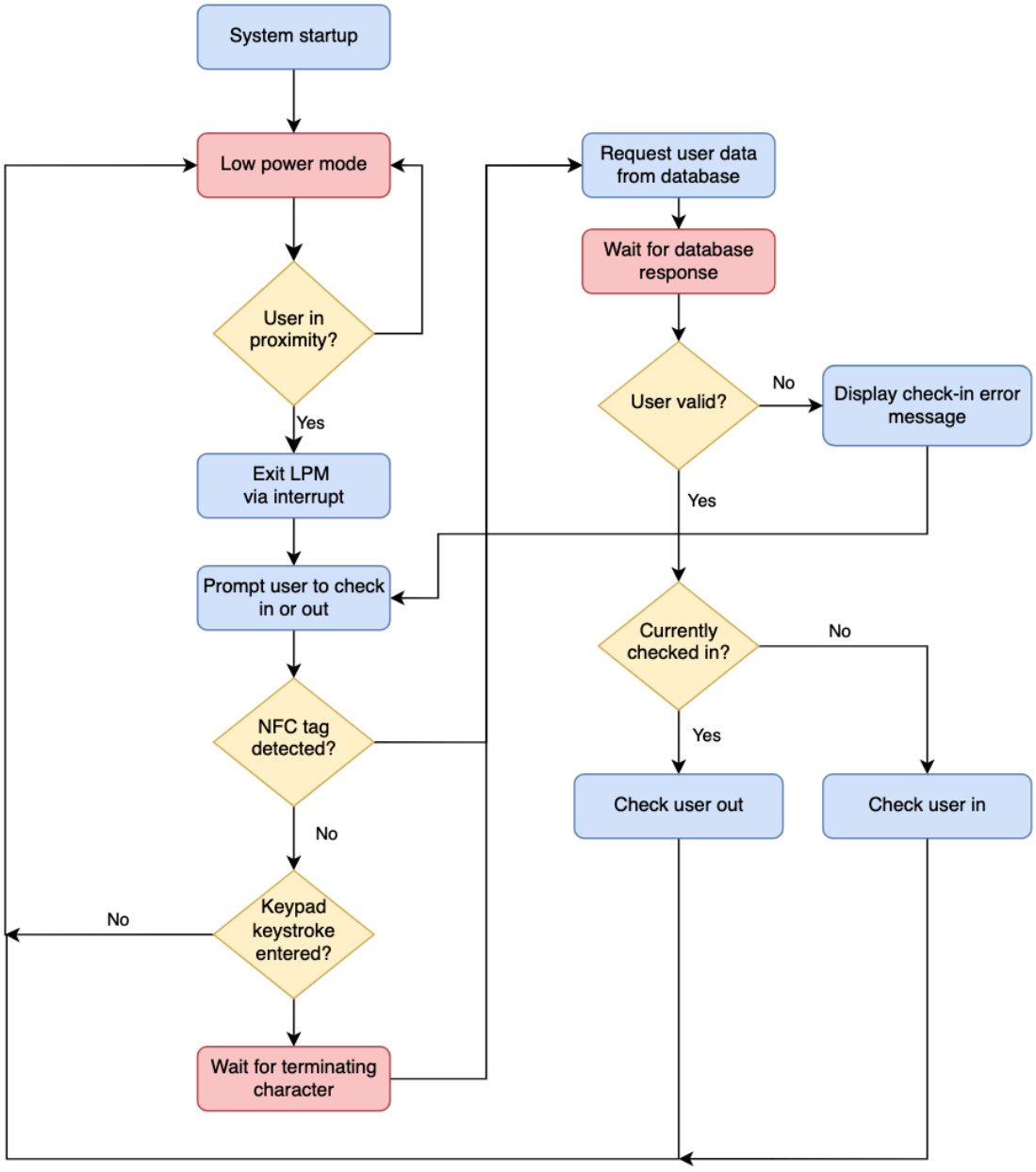


Figure 7.2 Software Flowchart for Kiosk Application

The control flow for the kiosk application is summarized in Figure 7.2. The ESP32-C6 is normally in low-power mode (LPM) until an interrupt via the proximity sensor wakes the system to process a check-in. When a user is in proximity, the ESP32-C6 begins polling the NFC reader and keypad for input. To enter an ID on the keypad, the numerical sequence is terminated with the '#' key, after which the ID is transmitted to the database to look up the user's credentials and check-in status. A similar process is followed for card authentication. After receiving a response from the database, the kiosk makes a decision based on the results and checks in/out the user or displays an error message.

7.2.2 Kiosk Administrator Mode

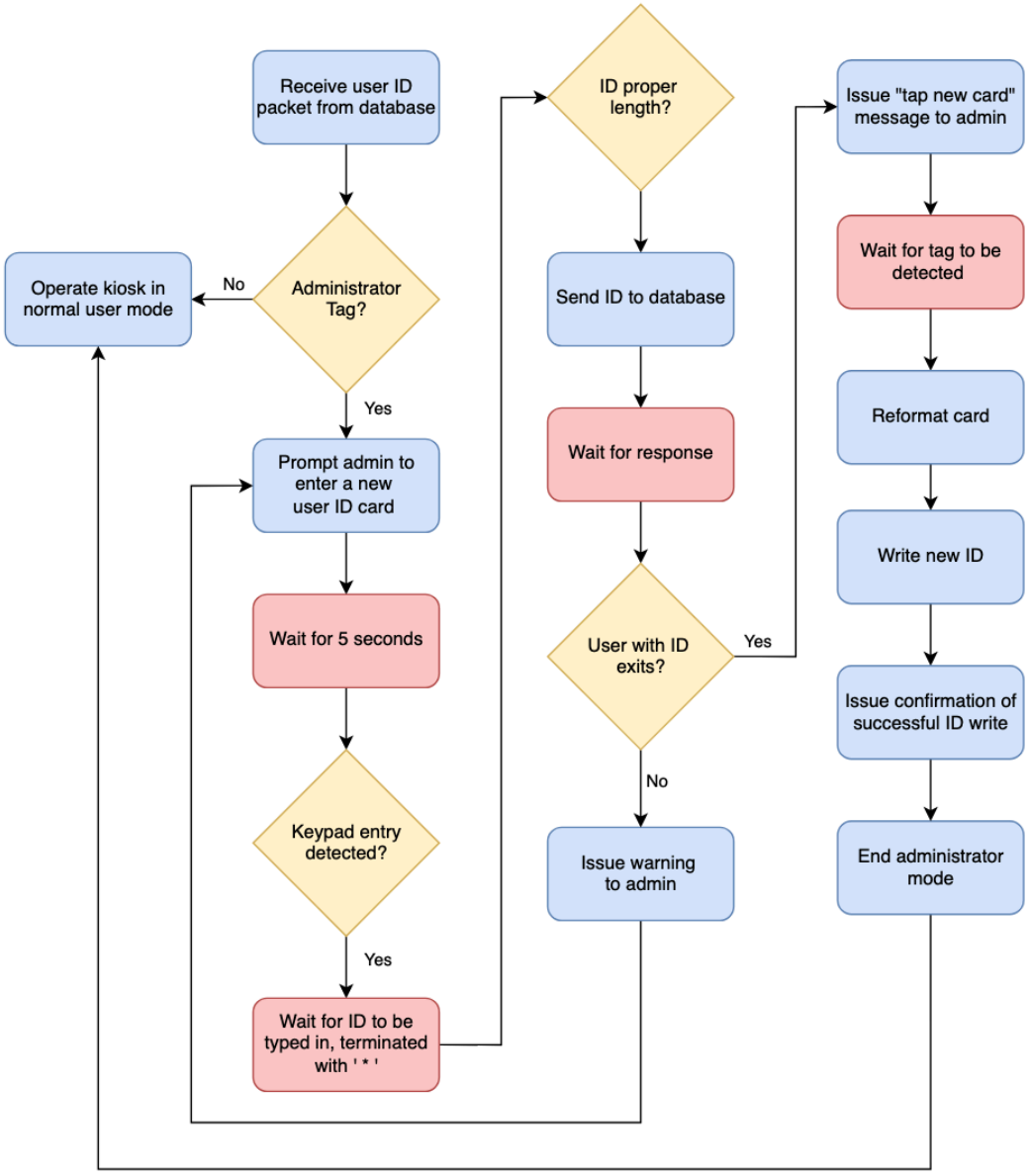


Figure 7.3 Software Flowchart for Administrator Mode

In kiosk administrator mode, a user with a dedicated administrator ID is able to format and write new user ID cards directly from the SCAN kiosk. This feature uses both the NFC transceiver and the keypad. Figure 7.3 shows the software flowchart for the kiosk in administrator mode. When the database returns the user ID packet of an administrator, the kiosk activates administrator mode. The administrator is then able to specify the ID number of a new user using the keypad. While the admin is able to format cards and write new IDs from the kiosk, the user's accounts must be created and added to the database through the SCAN webpage before the card can be written. If the administrator enters the ID of a user that does not exist, the kiosk issues a warning to the administrator via the display and prompts them to re-enter the ID they wish to format. Not shown in the diagram, administrator mode has a time-out feature to avoid unauthorized users from creating new ID cards.

7.2.3 Proximity Sensor Software

The SCAN system utilizes a Passive Infrared (PIR) proximity sensor to detect when a user approaches the kiosk. Movement near the kiosk triggers the system to exit low-power mode and enter an active state. The interaction is managed in software by a custom library, which handles the setup and response functions for the PIR sensor. The primary working principle behind the PIR sensor library is to set up and respond to motion detection events.

7.2.4 Card Interface Software

Card reading and writing is handled by the PN532 RFID/NFC transceiver embedded in the SCAN kiosk. As part of the SCAN kiosk, the PN532 operates primarily in reader mode, although it is capable of writing and formatting cards as well in administrator mode. Many popular libraries are available for interfacing Arduino and ESP32 microcontrollers with the PN532. These libraries contain high-level APIs for reading, writing, and formatting operations.

- **PN532**: An NFC library developed for reading/writing tags and interfacing the PN532 transceiver via I2C, SPI, and HSU.
- **NDEF**: Calls PN532 library to read and write NFC tags in the NFC Data Exchange Format (NDEF). It supports reading from and writing to Mifare Classic tags with 4 byte UIDs.

Card reading: The SCAN system operates in tag reading mode during regular kiosk operation. It expects IDs to be formatted according to the NDEF standard, which was outlined in Chapter 4. The kiosk also expects the ID cards to contain a single NDEF record within each message transmission. If the NFC tag is not in NDEF format or does not contain a single record, the invalid ID state is triggered and the user is prompted to re-attempt authentication.

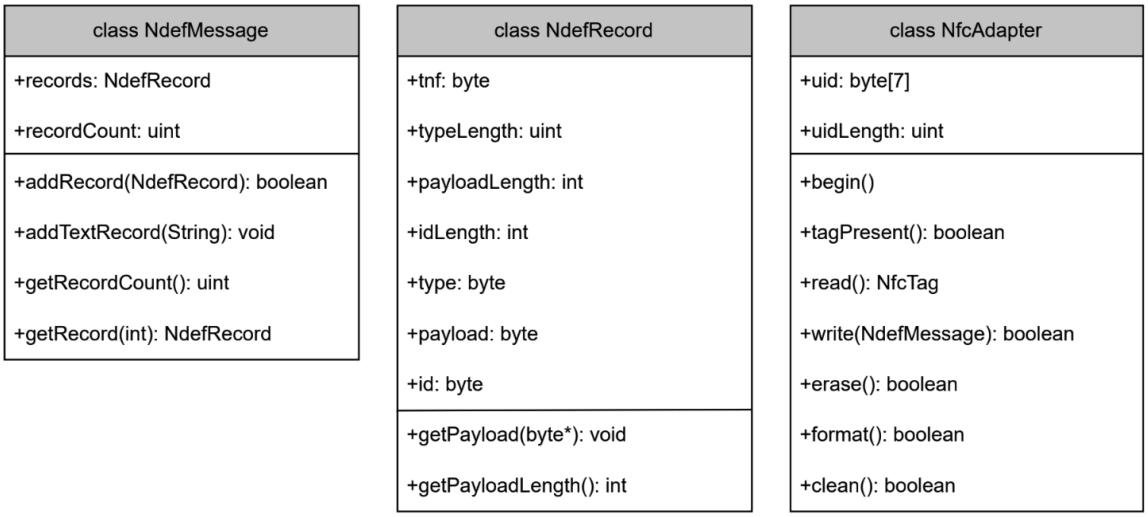


Figure 7.5 NDEF Library Classes Diagram

The NFC reader program used in this project utilizes the PN532 and NDEF libraries to read Mifare Classic tags, which represent UCF IDs in our RWC use case scenario. In this program, the PN532 library is included through the PN532.h and PN532_SPI.h headers, which allow communication with the PN532 module over an SPI interface. The NfcAdapter class from the NDEF library is used to abstract the interaction with NFC hardware and provides a straightforward interface for reading and writing NDEF messages. The NfcAdapter class is instantiated with the PN532 interface, enabling the program to detect and interact with NFC tags. When an NFC tag is detected, the program uses the NfcAdapter class to read the tag's contents. Once authenticated, it reads the data blocks from the card and decodes the NDEF message stored on it. The NDEF message is then processed to extract the user's ID into a String, which can be passed to the necessary functions to send the ID String to the SCAN database for authentication. A summary of the various states of the NFC reader software are described in Figure 7.4.

Card writing: In administrator mode, the kiosk can also write new ID tags using the nfcAdapter class. To create a new ID card, the card must first be formatted using the nfcAdapter method format(), which formats the card according to the NDEF specification. Then, a new NdefMessage object must be created, which stores an array of NdefRecord objects corresponding to URLs, ASCII strings, and other payload types. After the NdefRecords are added to the NdefMessage, they can be written to the card with the NfcAdapter's write(NdefMessage) method.

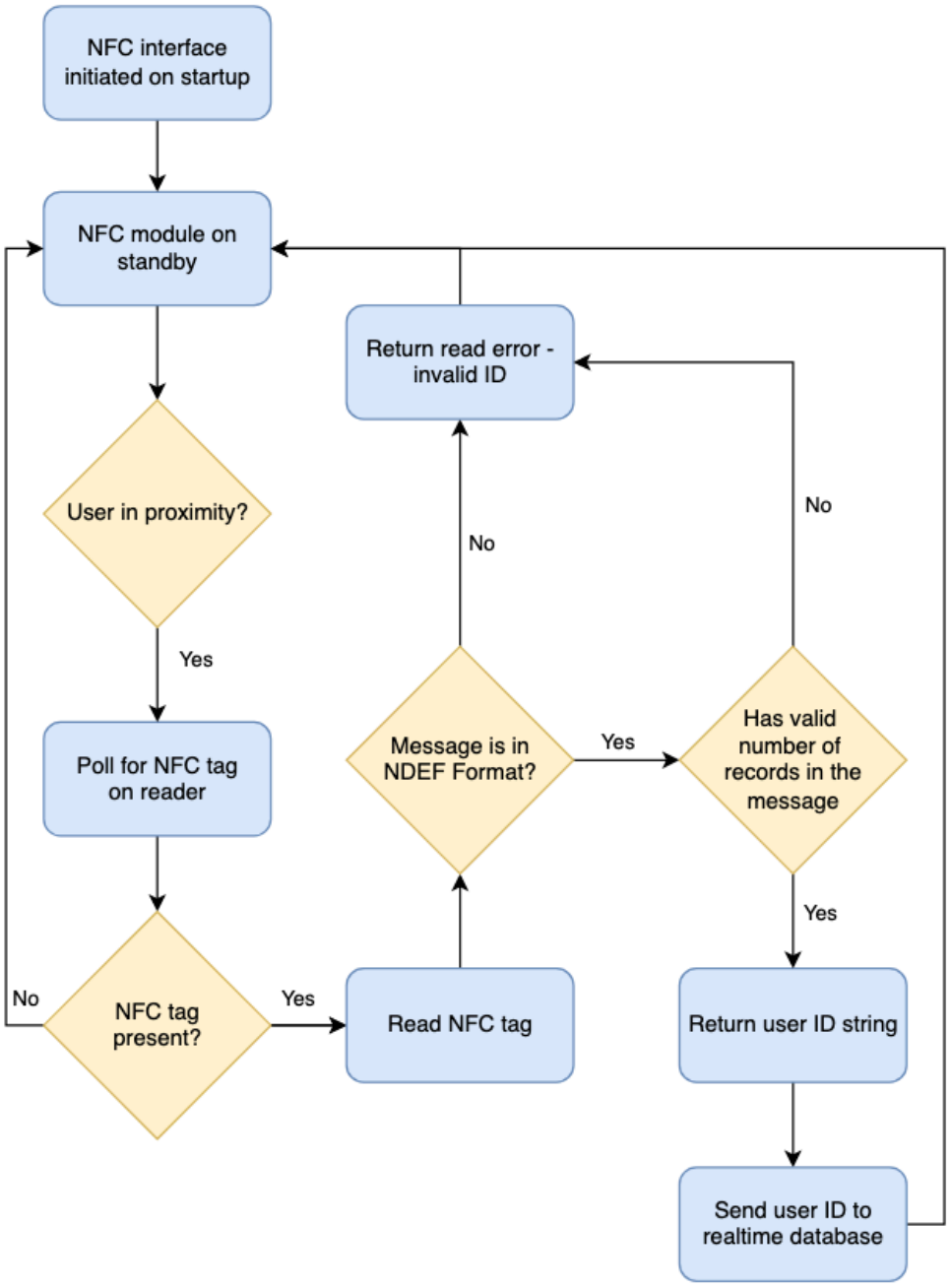


Figure 7.6 Card Reader Software Flowchart

7.2.4.1 Senior Design II Updates to Card Interface Software

In SDII, as we transitioned away from the Arduino framework to use ESP-IDF, we phased out the Adafruit PN532 libraries. For our final embedded firmware, we used an ESP-IDF SPI library to interface with the PN532, and we created a custom NDEF library based on the Arduino NDEF library. Setting up this library to work with our embedded firmware came with its challenges, but it was a necessary step to facilitate the transition to a professional grade, FreeRTOS-based development framework. Some of the challenges included fixing timing issues with the SPI driver, creating the APIs to format

text records (for user IDs) using the NDEF format, and improving the reliability and error handling of card read and write events. Overall, our card interface implemented ended up being incredibly reliable for both reading and writing.

7.2.5 Kiosk Display Interface Software

To create a responsive user interface, display states must be created for each key action of the SCAN kiosk. The display must guide users through the check-in and check-out processes. Two key graphics libraries are used to draw graphics for the display:

- **Adafruit_GFX_Library**: Core graphics library for all Adafruit and 3rd-party displays. This library is compatible with the Arduino framework and servers as a hardware abstraction layer (HAL) for hardware-specific display drivers. It contains graphics APIs for drawing shapes on the screen, font libraries, and conversion tools for displaying bitmap images.
- **Adafruit_GC9A01A**: This library extends the Adafruit GFX library to provide display driver support for the GC9A01A, which drives the TFT display used by the SCAN kiosk.

These libraries allow the programmer to address individual pixels in the display as well as draw shapes and graphics via the graphics APIs. Although the dimensions of the display are listed as 240 x 240 pixels, the visible display region consists of the circle circumscribed by the addressable display region, so any graphics drawn to the screen must take this into account.

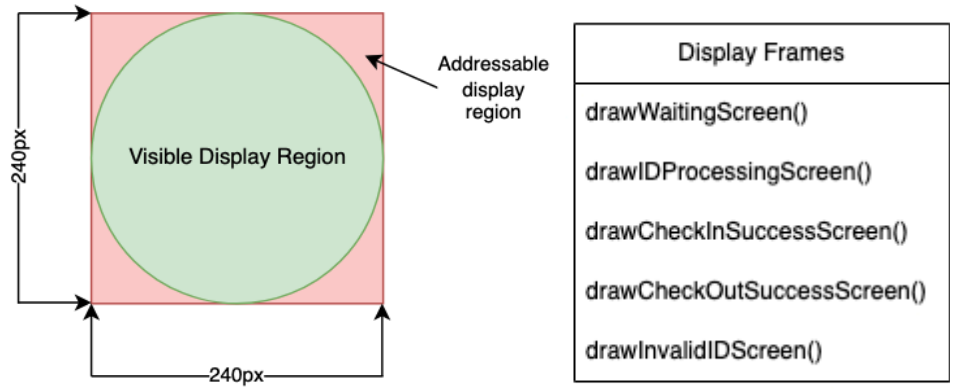


Figure 7.7 Display Addressable Region and Frames Library

A collection of display frames has been designed for each state of the check-in process. In the kiosk application software, the frame displayed at any given time is a function of the current state of the kiosk. The different states of the display during kiosk user mode are shown in Figure 7.8. These display frames are maintained in a library and may be drawn to the display with the corresponding function calls in Figure 7.7. In addition to the display, feedback to the user is provided via status LEDs and a buzzer. A green LED indicates check-in/check-out success, a red LED indicates an error or invalid state, and a blue LED indicates that the kiosk is operating in administrator mode. The buzzer is programmed to emit specific tones to correspond with these states.

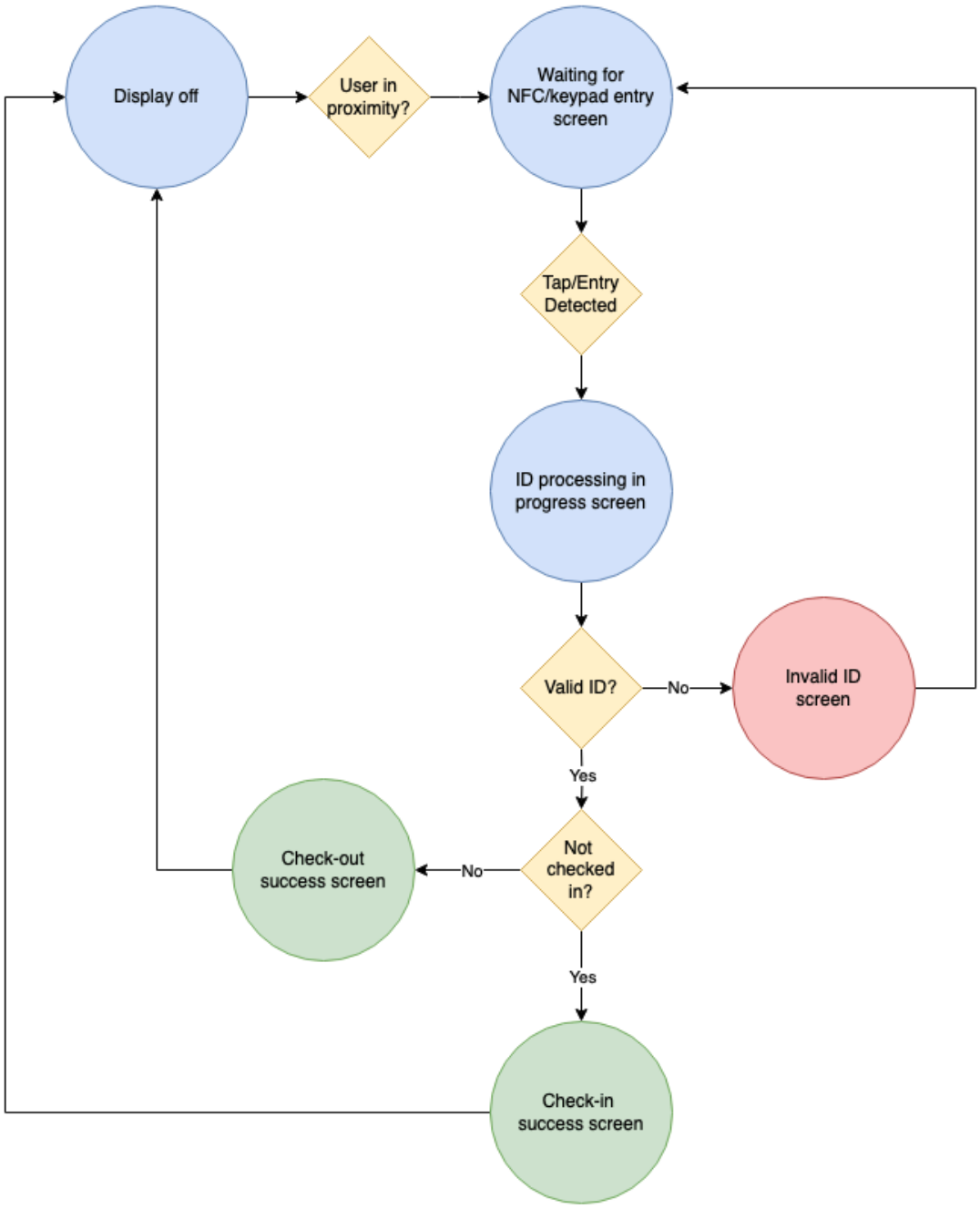


Figure 7.8 Display Software Flowchart (Kiosk User Mode)

7.2.5.1 Senior Design II Updates to Display Interface Software

As with the card interface firmware, the display libraries we selected in Senior Design I were incompatible with the ESP-IDF framework. Fortunately, ESP-IDF has support for a powerful GUI library, the Light and Versatile Graphics Library (LVGL), which allows for the creation of responsive, animated user interfaces through a display abstraction layer. With LVGL, we were able to incorporate icons, spinner animations, and text boxes into our kiosk displays to create a clean and intuitive user interface.

7.3 Database Design

The SCAN system uses a cloud-based database to manage check-in and check-out events and user profiles. To streamline interaction with the Firebase Realtime Database, we developed a custom library that provides high-level API functions, allowing for efficient and organized database operations.

Firebase Realtime databases, by default, use a NO-SQL/JSON tree structure. Our database takes advantage of this structure, dividing our data into four main collections, as shown in Figure 7.9.

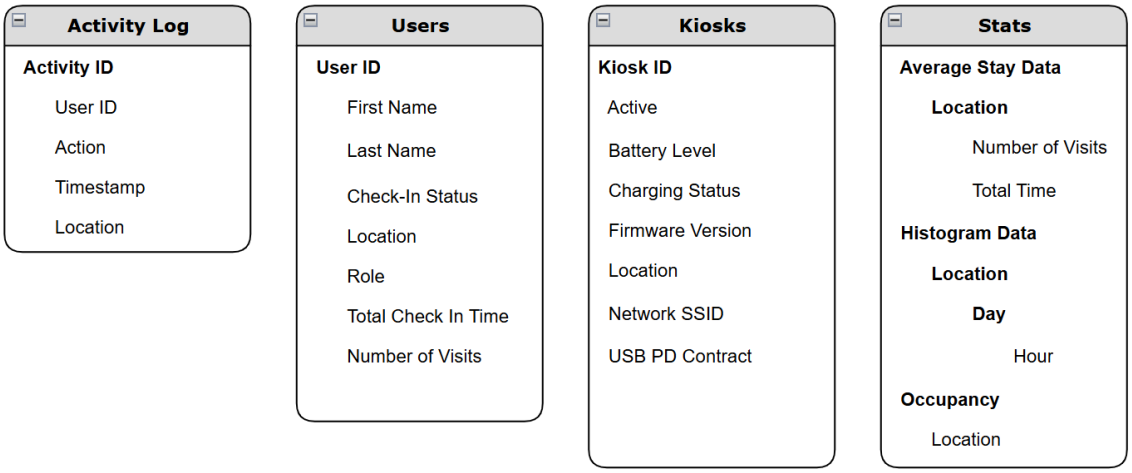


Figure 7.9 Database Fields

Users: The Users datapath contains a collection of user profiles consisting of the user's ID, passkey, role, check-in status, active status, and references to their most recent check-in/check-out events in the corresponding collection. Users in the user collection can be easily indexed using their unique user ID associated with their NFC tag. This design ensures that the SCAN system can easily retrieve relevant data, provide real-time feedback to users, and generate statistical insights for the admin.

Activity Log: The activity log collection is designed to track individual check-in and check-out events, allowing the system to record attendance, view real-time occupancy, and analyze usage patterns.

Stats: The stats node of the database keeps track of all statistical data processed by the Firebase Cloud Functions Backend including live occupancy, average stay duration, and histogram data.

Kiosks: The kiosks node of the database stores all information related to each running kiosk, including battery and charging information, software version information, and more. This data is updated by the ESP32 only and is displayed on the kiosk management page on the web app.

Activity Log: The activity log collection is designed to track individual check-in and check-out events, allowing the system to record attendance, view real-time occupancy, and analyze usage patterns.

Custom Database Library: The custom database library provides an interface between the main application and Firebase Realtime Database, simplifying code management and enhancing modularity. The library wraps the functionality of the ESP32 Firebase library, exposing only the most high-level functions to the main file.

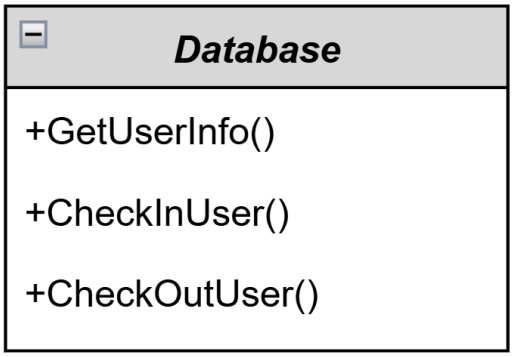


Figure 7.10 Database Class Diagram

- **GetUserInfo(userID)**: Retrieves user information, such as user ID, passkey, and check-in status, allowing for authentication and status updates during user interactions.
- **CheckInUser(userID)**: Logs a check-in event for the specified user, updating the user's profile status and adding a timestamped entry to the check-in log.
- **CheckOutUser(userID)**: Logs a check-out event, updating the user's profile status and appending a timestamped check-out entry in the database.

The main application code interacts only with these high-level API functions, which in turn handle the detailed database operations.

Database Operation Workflow:

1. **Check-In Process**: When users scan their NFC tag or enter their ID via keypad at the kiosk, the main file calls getUserInfo from the custom library. If the getUserInfo function returns a user profile, the user is valid and the checkInUser() function can register the user's check-in in the database. This function packages the user's ID and timestamp data, connects to the Firebase Realtime Database using the ESP32 library, and logs the event.
2. **Check-Out Process**: When a user checks out, the main file invokes checkOutUser(). This function updates the database with the user's check-out status and time, ensuring a complete log of attendance.
3. **Create new User Process:** When an NFC card or user ID is entered via the kiosk, the getUserInfo function is invoked. If the user role parameter in the returned user profile marks the user as an admin, the kiosk transitions into a mode for creating a new user and linking the ID to a new NFC card. Upon valid entry of a new user ID, the createNewUser function is called to register the user in the database.

The custom database software design integrates Firebase Realtime Database with SCAN's ESP32 microcontroller, providing streamlined, high-level API functions for managing user and occupancy data. This approach ensures efficient, organized, and reliable data handling for both user check-ins and administrative monitoring, enhancing the overall functionality and maintainability of the SCAN system.

7.4 SCAN Web Application

The SCAN web application is a frontend React app that serves as the main interface for administrators to analyze and manage user check-in data generated by the SCAN system. It integrates directly with the Firebase Realtime Database using the Firebase Realtime Database SDK, enabling it to provide real-time statistics, log historical activity, and facilitate user management through an intuitive graphical interface.

7.4.1 Web Application Key Features

The SCAN web application is divided into four main pages: the analytics dashboard, activity log, user management, and kiosk management pages. Together, these three pages allow for efficient analysis of user check-in data and management of user profiles and kiosks by admins.

Analytics Dashboard: The main dashboard displays real-time data insights, including the current number of attendees checked in, a histogram showing the peak attendance hours for specific days, and the average attendance duration for users. Data for these statistics is sourced from the SCAN database in real-time. Efficient data handling is achieved through periodic and targeted queries, with computations performed directly in the user's browser to minimize transmission delays and eliminate the need for intermediate processing. Once computations are complete, Chart.js is utilized to create visually engaging, interactive graphs, including the histogram illustrated in Figure 7.11, offering a clear and intuitive representation of user data.

Activity Log Page: The activity log presents a table of all events captured by the system, such as user check-ins and check-outs, serving as an optimal way for admin users to view historical attendance data. Each log entry includes the following details: user ID and name, action (e.g. check-in or check-out), time, and location. Additionally, this page provides a search feature for filtering user data, allowing admins to generate comprehensive activity logs specific to any event characteristic. As a stretch goal, our website implements lazy loading to ensure time-efficient access to this data.

User Management Page: This page is designed to manage user profiles in the database. Administrators can:

1. Search for users by name or user ID.
2. View user details like name, UCF ID, check-in status, location, average stay duration, and total occupancy time
3. Edit user details, such as name, UCF ID, and check-in status.

4. Add or remove users from the system.

Similar to the activity log page, we implemented lazy loading as a stretch goal to ensure efficient access to user data on this page. This means user data will only be loaded on request other than the small amount of data loaded on the initial loading of the page. Data further down in the table or hidden behind profile cards will only be loaded based on input triggers like scrolling and click events.

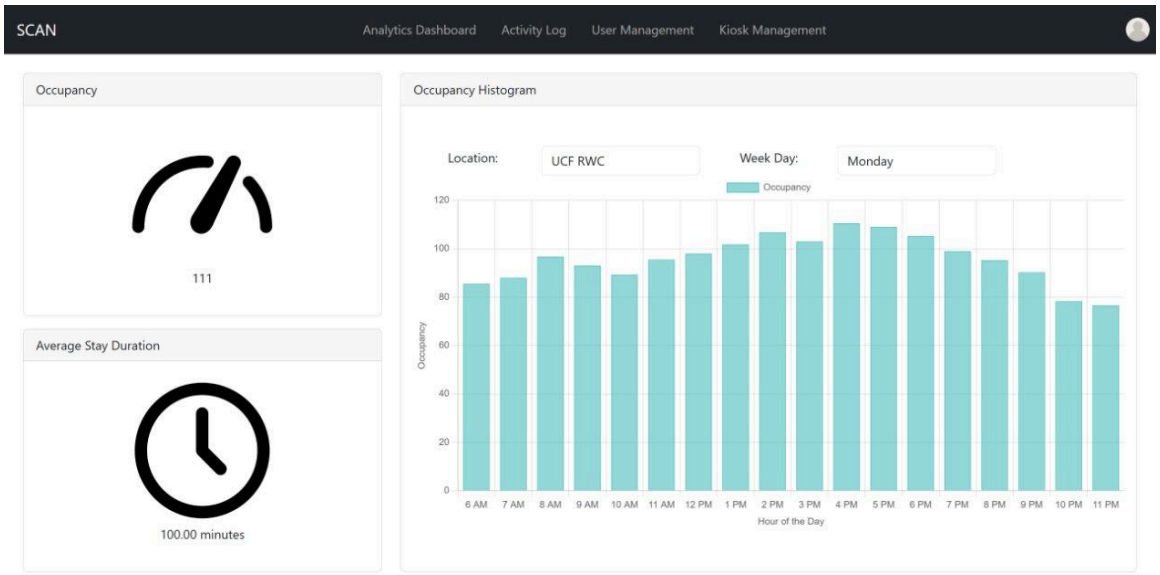


Figure 7.11 Analytics Dashboard Page

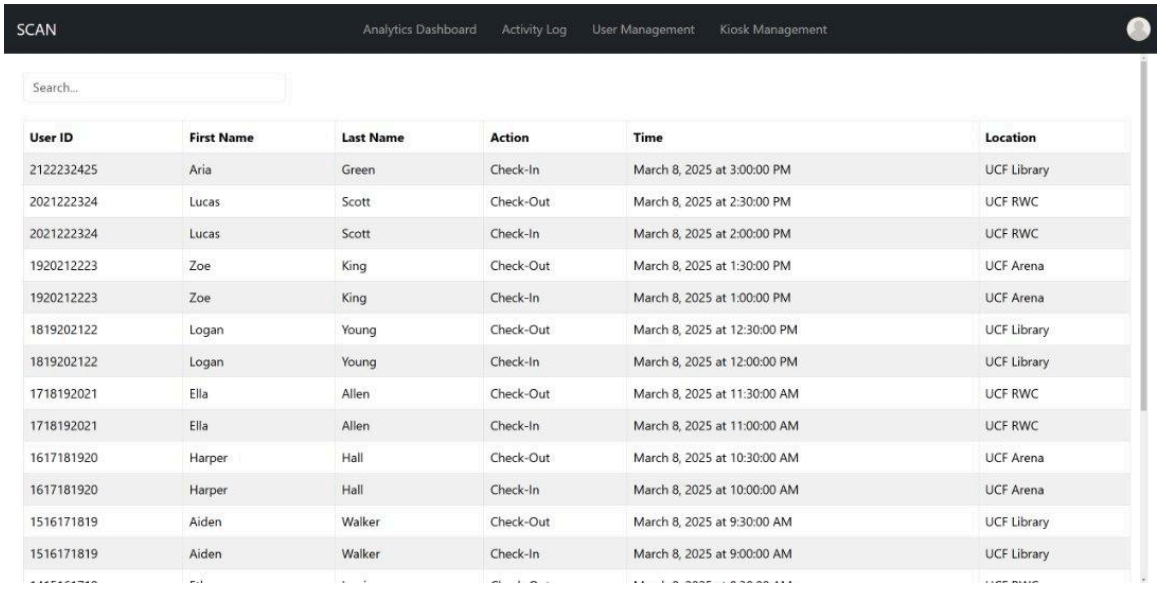


Figure 7.12 Activity Log Page

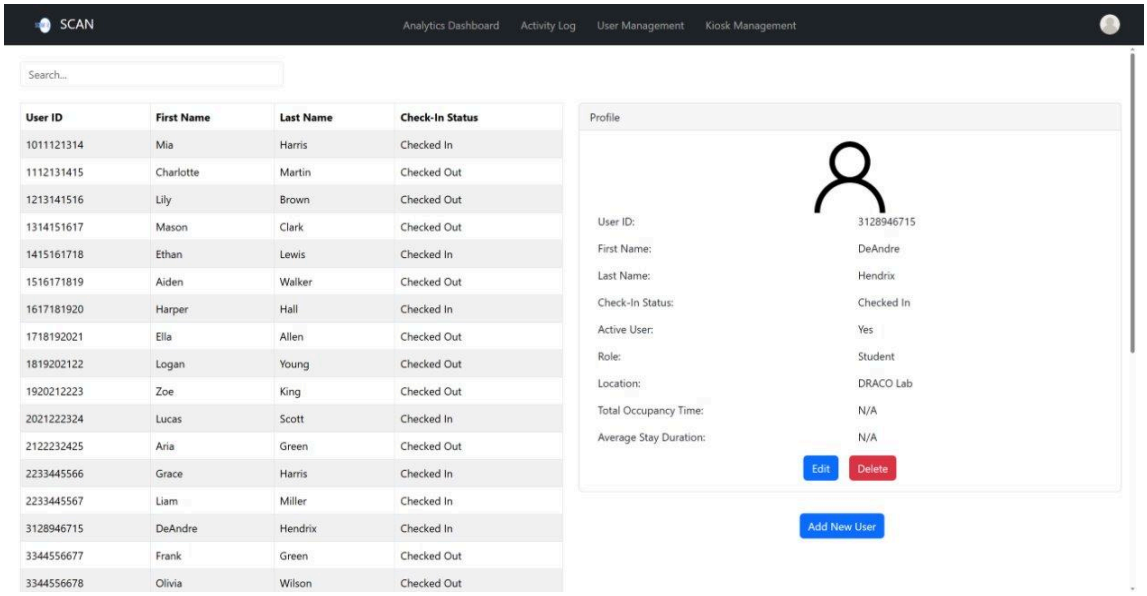


Figure 7.13 User Management Page Wireframe

Kiosk Management Page: This page is designed to separate each available kiosk. Each kiosk has its own information, such as battery voltage and current firmware version, stored in the database. Real-time messages are sent over MQTT to update this page. This page also includes a fully integrated way to update the kiosk firmware over-the-air.

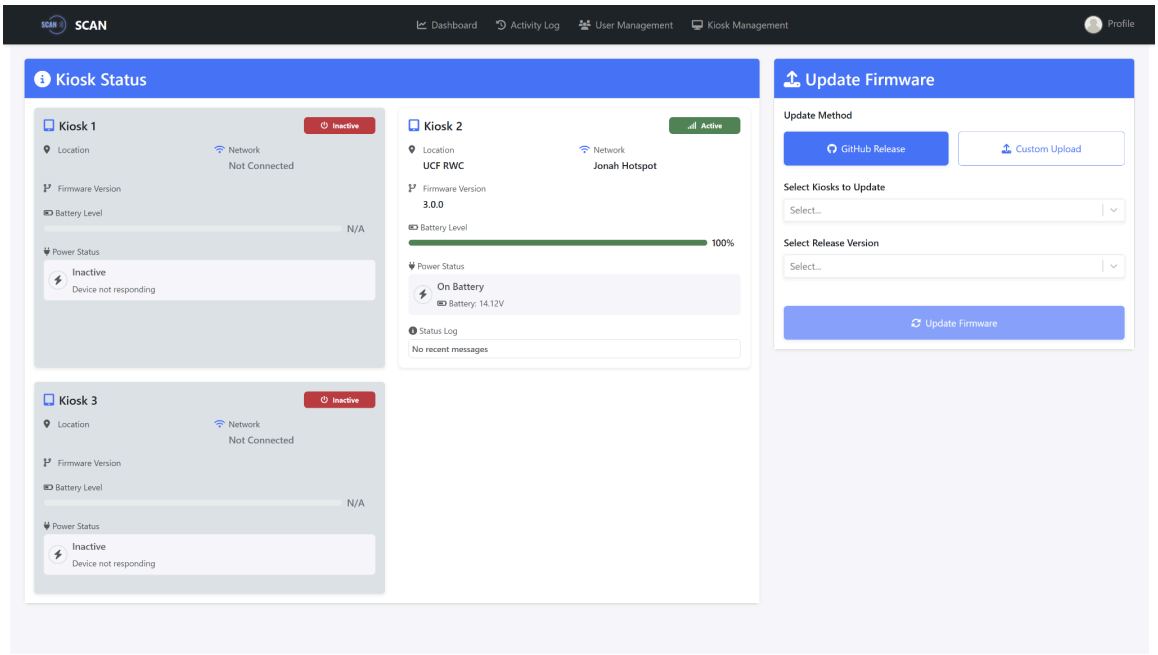


Figure 7.14 Kiosk Management Page

7.4.2 Technical Implementation

React Framework: For the implementation of the SCAN web application, we chose React for its modular component-based architecture, enabling efficient development and dynamic updates to the user interface.

Firebase Realtime Database SDK: To integrate our React Web app with SCAN's database, we opted to use the Firebase Realtime Database SDK which offers native support for database querying operations tailored to Firebase, enabling rapid development and seamless interaction. Firebase Realtime Database SDK provides a high-level API for performing data operations like complex querying that would otherwise be much more technically involved to implement. We took advantage of these API functions for faster and more efficient development, providing SCAN admins with information like the total number of attendees checked-in and historical attendance logs that can be used to compute statistical metrics.

Firebase Hosting: Like the database, the SCAN web application is also hosted on Firebase using Firebase Hosting. Firebase Hosting is an obvious choice for this project as it integrates cleanly with our database already hosted on the platform. Similar to the Firebase Realtime Database, Firebase also offers a free tier for their Cloud Services which means we didn't incur any costs for our use case.

Chart.js: Chart.js is used as the primary tool for displaying user data graphically, specifically, the user activity histogram shown in Figure 7.11.

7.5 Over-the-Air Update Implementation

Over-the-air (OTA) updates are a key feature for the SCAN kiosk, enabling seamless firmware updates without requiring physical connections or manual intervention by end-users. First described in Chapter 2 as a stretch goal, implementing over-the-air updates in software required a process to handle building new binary files (either on a host machine or server), uploading binary files to a secure server that the ESP32 can access, downloading the data contents to the ESP32, and running the new firmware. To achieve this, Docker was used in conjunction with the ESP-IDF framework, which provides an API to implement this functionality in an efficient manner.

The software for OTA updates takes advantage of the ESP32's ability to manage multiple flash partitions. To successfully update firmware on-the-go, the ESP32's partition table includes two slots - one for OTA app binary files to be stored and a small partition to specify which data should be used on the next reset. Espressif provides a high-level diagram, shown in Figure 7.15, explaining the full process for those running their own HTTP server. [70]

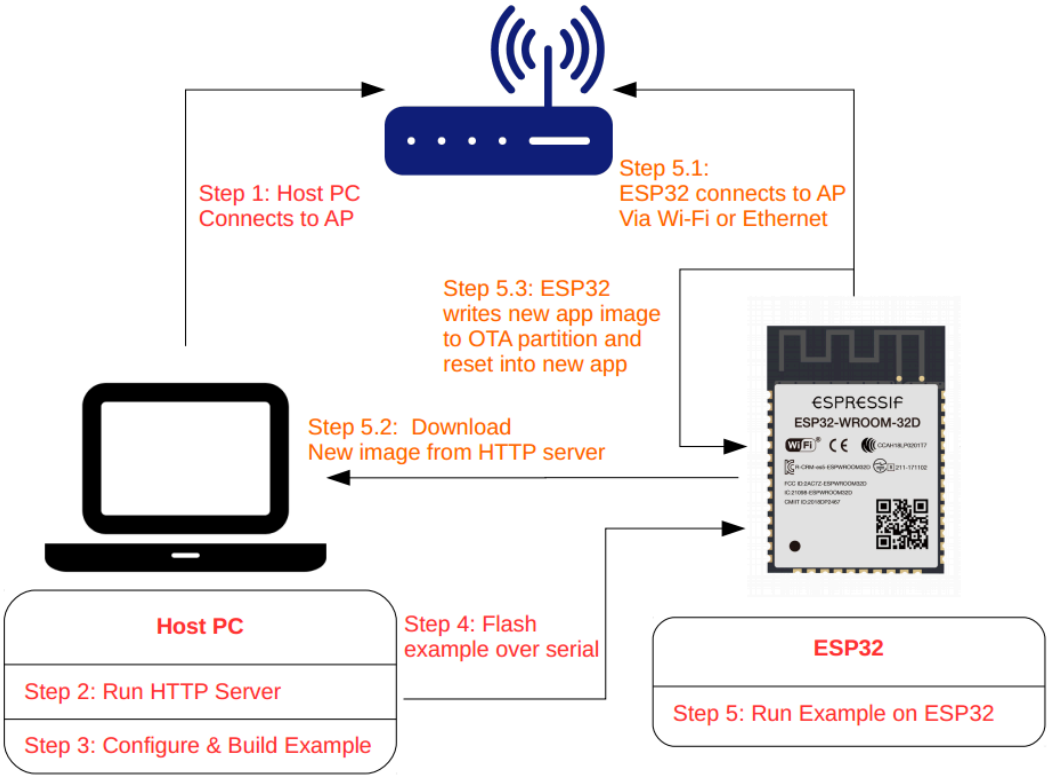


Figure 7.15 Sequential Diagram of OTA Update from HTTP Server

The first step described in Figure 7.15 is the host PC connecting to an access point. In our case, the PC is to the internet via a router/modem. In step 2, since we hosted our binary files on GitHub, we did not need to run an HTTP server. Moving on to steps 3, 4, and 5, example code is compiled, flashed over serial, and ran on the ESP32. This example code contains the software that fetches the new firmware over-the-air and the necessary flash partitions to store it. The example program must also set up the ESP32 to connect to the internet via Wi-Fi, which is required for step 5.1. Again, since we did not use a local server, the ESP32 just needed to be connected to the internet to access GitHub.

At this point, the new binary file is prepared for the ESP32 to download over-the-air. We took advantage of Docker and GitHub Actions for the automated build pipeline. When the SCAN team makes changes to the ESP32 firmware and submits a pull request or merges changes into the main branch of the GitHub repository, a GitHub Action is triggered. The GitHub Actions workflow uses a Docker container, as described earlier, to compile and build the ESP32 firmware. This ensures that the build process is consistent and independent of local development environments. If the workflow completes successfully, the compiled binary file is automatically hosted as a release on GitHub. The release can even be tagged and versioned for better firmware organization.

The last part of the OTA update required the ESP32 to check for new GitHub Releases or use an event-based or time-based trigger. For example, the software can be designed such

that an over-the-air update is triggered by entering a certain keypad combination or scanning a particular NFC tag on the kiosk. In our case, an OTA update is triggered from the kiosk management page on the SCAN web application. After the kiosk is triggered, the following steps occur:

1. The ESP32 establishes an HTTPS connection to GitHub, validating the server's authenticity using a root certificate to prevent malicious attacks.
2. The ESP32 sends an HTTPS GET request to receive the binary file contents hosted on GitHub. The data is read into one of the OTA app partitions in the ESP32 flash storage.
3. The ESP32 verifies the firmware and updates its bootloader settings to boot the new firmware.
4. The ESP32 reboots and starts running the updated firmware.

Steps 5.2 and 5.3 in Figure 7.15 correspond to the four steps described above, where the binary file is downloaded to the ESP32, and the ESP32 reboots with the new application firmware.

8. SYSTEM FABRICATION

As explained in the Senior Design lectures, any hobbyist can breadboard together development modules to achieve functionality. However, it is a true demonstration of the lessons learned in our analog circuit design classes to design, manufacture, test, and integrate a PCB assembly consisting of a microcontroller, major peripherals, passive components, and all necessary interconnects. To design the SCAN PCB, the SCAN team used KiCAD, which is a free and open-source PCB design suite used for schematic capture, footprint design, and PCB layouts. KiCAD's rich feature set and flexibility make it a powerful tool for both hobbyists and experienced PCB designers.

8.1 PCB Design Process

The first step in designing our PCBs was creating the schematic, which was detailed in previous sections. The schematic shows the electrical connections between all components. Following completion of the schematic, an Electrical Rules Check (ERC) was performed to ensure that all pins were connected or marked as "no connection". When transferring these connections to a PCB layout, each component was first defined as a footprint - a dimensionally accurate representation of the component's pins and profile. KiCAD comes pre-installed with many footprint libraries for frequently-used components such as resistors, capacitors, and microcontrollers, Footprints that are not found can usually be downloaded from the internet. In the event that a KiCAD footprint does not exist, the built-in footprint editor allowed us to define the pad placement, silkscreen, and keepout area for a custom footprint. Once all schematic components were assigned a footprint, the PCB layout process could be started. Listed below were the four main considerations for designing our PCB.

1). **Board dimensions**: the shape and size of the board is drawn and defined. When designing the SCAN PCB, we ensured that the board dimensions fit within the kiosk enclosure and had proper mounting points.

2). **Component placement**: this step involves placing the footprints onto the PCB. As a rule of thumb, components should be grouped by function. In the case of SCAN, any decoupling capacitors for voltage regulators should be located physically nearby. Additionally, significant components should be placed first, because these components often have the most physical constraints. For example, the ESP32-C6 should be placed first, as all of the other components must be routed to its location. The position of the USB-C connection is also important, as it must connect to a cable external to the kiosk. Finally, consideration must be given to the routing of signals. If signals must cross over one another or be routed on opposite sides of the board to avoid intersection, there may be a better method of footprint placement. The final layout (top layer only) for the SCAN PCB can be found in Figure 8.1. In addition, a 3D model of the PCB can be found in Figure 8.2.

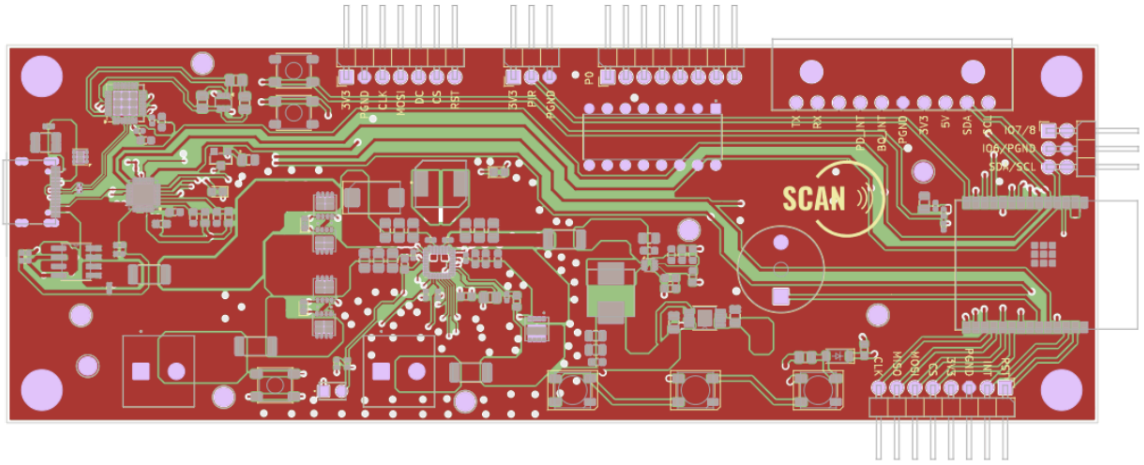


Figure 8.1 PCB Layout (Top Layer and GND1)

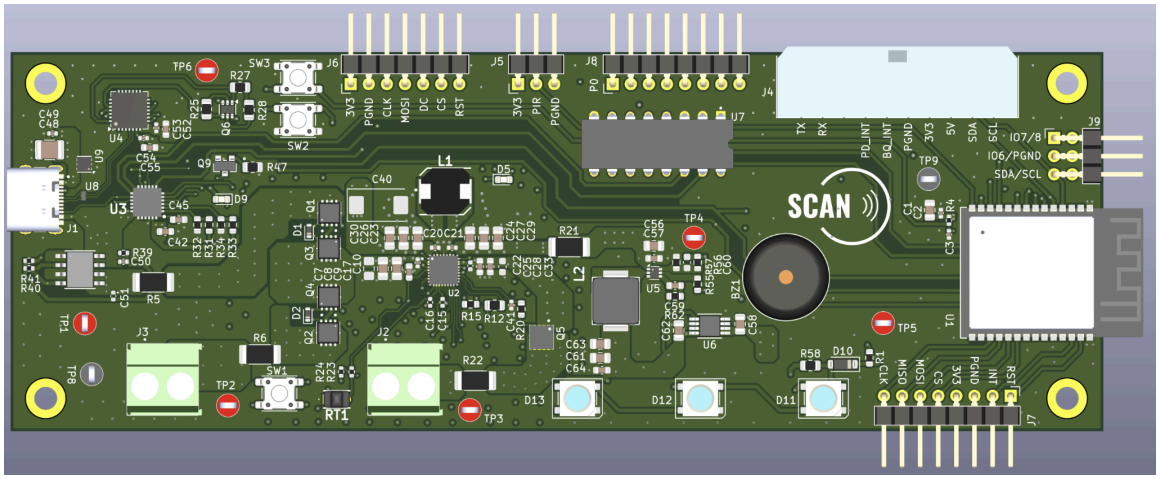


Figure 8.2 PCB 3D Model

3). **Trace routing**: routing is the process of connecting the pads of each component together with copper traces. Depending on the type of signal being routed (either data or power), traces may have different widths. Generally, traces for low-speed signals such as the ones for the SCAN project can be 8-10 mils (one thousandth of an inch). Meanwhile, power traces are often much wider to compensate for the larger currents they carry. In the case of SCAN, it was beneficial to assign wider trace widths to power traces in the battery management system circuit. Another rule of thumb is to avoid sharp turns greater than 45 degree angles when routing, which can disrupt the flow of current and introduce sharp changes in impedance along a trace.

As the number of components on a PCB expands, the routing process can become a significant challenge. When it becomes impossible to route traces on a single layer of the PCB due to paths being blocked by other traces, vias can be used to route through the board and continue routing the trace on the opposite side. For PCBs with more than two layers, vias can be used to access any layer of the PCB for routing.

4). **Ground planes and thermal management**: a solid ground plane is placed on one layer of a two-layer PCB to provide a low-impedance return path for current on an interconnect. This low impedance reduces noise and electromagnetic interference between traces. For PCBs with high-frequency signals, ground planes can minimize signal distortion and cross-talk between traces. Ground planes can enhance power delivery throughout the PCB by stabilizing the reference voltage for all components. Finally, ground planes can more evenly distribute heat across the PCB, which prevents any centralized hotspots around core components like the ESP32-C6.

8.2 PCB Manufacturers

After designing the SCAN PCB, we exported the design to a PCB vendor for manufacturing and assembly. When choosing a manufacturer for our PCB, the most important factors were cost, time to manufacture, and support for assembly services. For cost, we desired to keep the development cost of SCAN as low as possible. Additionally, we worked under a tight time constraint to obtain a working PCB assembly during Senior Design II. As a result, whichever manufacturer we choose needed to have short manufacturing times and quick shipping to allow for multiple iterations of the SCAN PCB. Finally, we wished to order a pre-assembled PCB to avoid the difficulties of manually placing and reflow soldering SMD components. We identified three potential manufacturers for our PCB.

JLCPCB is a popular PCB manufacturer for hobbyists and developers on a budget. It has options for PCB prices as low as $2 for 5 PCBs. From the time of order, JLCPCB usually completes production in 2-3 days. With express shipping, boards can be delivered within a week, which made JLCPCB a solid option for our time-constrained senior design project. JLCPCB offers assembly services and also integrates with EasyEDA to provide a seamlessly integrated PCB design process.

PCBWay provides a range of services, catering to more than just PCB manufacturing. PCBWay is known for its high-quality boards and manufacturing, and it offers competitive pricing, with 5 PCBs starting at $5. It also provides options for advanced features, making it a preferred choice for complex designs. Production times are flexible, with options like 24-hour production for a fee, and shipping times vary based on the chosen method. Beyond PCB manufacturing, PCBWay offers assembly services and even 3D printing.

OSHPark is known for having superior PCB quality than other manufacturers, distinguished by its purple solder mask and ENIG finish. This finish protects exposed copper pads and makes the PCB surface more solderable. While the pricing is higher compared to competitors—around $38.70 for 3 boards—OSHPark's quality and reliability can make it a strong choice for professional projects. OSHPark's ordering process is user-friendly, but lead times can be longer, with production and shipping often taking up to two weeks, meaning we could have run into issues with completing our PCB on time if it required multiple design iterations. Unlike PCBWay, OSHPark primarily focuses on PCB manufacturing, and it does not offer PCB assembly services.

Table 8.1 Comparison of PCB Vendors

Feature	OSH Park	JLCPCB	PCBWay
Cost (Two-layer PCB)	~$38.70 for 3 boards	$2 for 5 boards on the first order, $5 for 10 boards after	$5 for 10 boards
Shipping costs	Free within US	Starts at $8	Slightly higher shipping than JLCPCB
Turnaround time	Standard: 12-21 days, less at a higher shipping cost	Standard: 2-3 days manufacturing + shipping; 24-hour production for a premium	Standard: 2-3 days manufacturing + shipping; 24-hour production for a premium
Finish options	ENIG only	HASL (leaded or lead-free), ENIG	HASL (leaded or lead-free), ENIG
Quality	Known for excellent quality and distinct purple solder mask	Good quality for hobbyist projects, minor silkscreen clarity issues reported by users	Offers high-quality boards
Assembly services	No assembly services	Yes, with parts sourcing from in-house library and global distributors	Yes, with parts sourcing from authorized distributors

Ultimately, JLCPCB was the best option for fabricating the SCAN PCB. With quick manufacturing and shipping times, deals for first-time orders, and dedicated services for PCB assembly, JLCPCB greatly assisted in the iterative development of our PCB. We only required one iteration of our main PCB, but if more were required, JLCPCB's quick manufacturing and shipping times would have been of great benefit.

8.3 Enclosure Fabrication

The enclosure for the SCAN Kiosk houses and protects the hardware in an aesthetic package. The enclosure internally houses the PCB assembly, battery, and charging system, and it exposes the keypad, NFC reader, and TFT display to the user on a front panel. It is designed to ensure durability, user-friendliness, and aesthetic appeal. Autodesk Fusion360, a popular cloud-based computer-aided design software, was used to design the kiosk enclosure. We then fabricated the enclosure using the Creality Ender 3 V2 Pro 3D printer.

The use of 3D printing ensured that we could rapidly prototype and revise the enclosure design throughout the project's lifestyle. Any dynamic features such as PCB size, component changes, or customer requirements can be easily adjusted for by creating a parameterized, constraint-driven 3D model in Fusion360. This also ensured that we were able to abide by our manufacturing and sustainability constraints, as we can effortlessly reproduce the kiosk enclosure to the exact same specifications using 3D printing. In addition to the 3D-printed enclosure, a few non-3D printed components were acquired to assemble the stand for the kiosk. A PVC pipe attaches the base to the bottom of the enclosure by 3D-printed flanges (See Figure 8.3).

The first step of the enclosure design process was to conceptualize the design, giving thought to the ergonomics and functional requirements of the kiosk. This included determining the position and placement of features such as the display, NFC transceiver, and keypad. This model, shown in Figure 8.4, can then be converted into a dimensionally accurate, manufacturable design in Fusion 360. In the model, the PVC pipe stand has been shortened for ease of viewing. Cutouts with appropriate tolerances were made for each component and fixture point, and the design takes into account any cable management paths and internal circuitry.

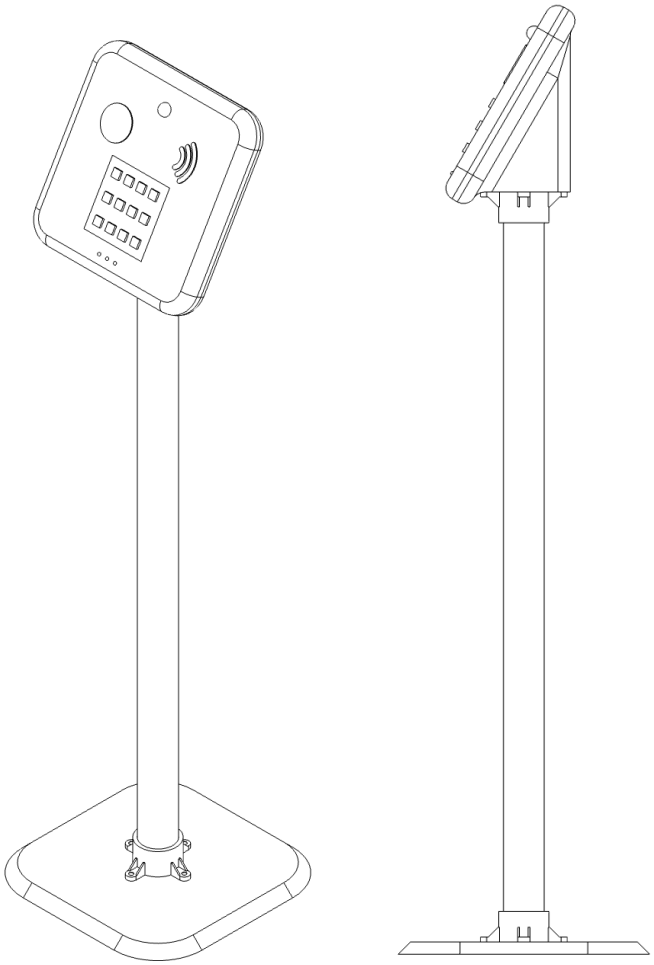


Figure 8.3 Wireframe Rendering of Kiosk Enclosure

The PCB was mounted with spacers inside of the SCAN enclosure on the slanted front plate that secures the keypad, display, and NFC transceiver. By mounting the PCB on the front surface, the NFC antenna could be positioned close enough to the surface for card reading. Furthermore, mounting the PCB on this surface meant that the keypad, display, and status LEDs were able to be mounted to the front surface and interfaced with the PCB via short ribbon connectors.

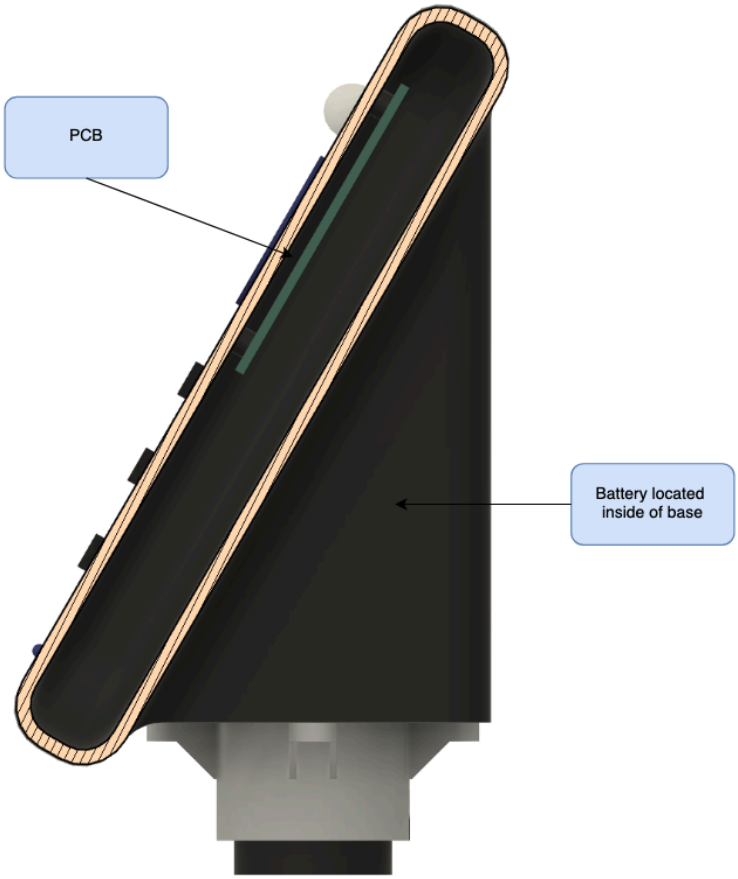


Figure 8.4 Proposed PCB Mounting Scheme

Once the first design iteration was completed, the 3DFM file, which contained the triangular mesh model of the design, was exported to a 3D printing slicer. This slicer allowed us to configure the ideal settings for the print and ultimately generate the GCode file that the Ender 3 Pro used to print our final design. By utilizing Fusion 360 for design and the Ender 3 Pro for fabrication, the enclosure achieved a balance between cost, functionality, and aesthetics.

9. SYSTEM TESTING AND INTEGRATION

This section will describe our prototyping, testing, and integration plans for each hardware and software system. For hardware, testing involves ensuring that each device is functioning as expected through a combination of physical analysis with test equipment and software testing for more detailed debugging. For software, unit testing was performed to verify the absence of any edge cases or bugs. The integration phases involved carefully merging code bases and shared data buses to ensure that no functionality was lost during integration.

9.1 Testing & Evaluation

Prior to integrating each hardware and software component of SCAN, we conducted a series of tests on each to ensure that they were functioning as expected. This testing helped us evaluate whether or not we met the requirements laid out in our requirement specification for the project. The software testing process for SCAN's kiosk system aimed to validate all embedded, frontend, and backend components to ensure they functioned reliably within the system's intended use cases. Testing covered firmware for various peripherals, the kiosk's frontend software, database integration, and the analytics dashboard. Dedicated PlatformIO projects were created in VSCode to test firmware-related system components, allowing us to test each component and development board before full system integration.

9.1.1 ESP32-C6 Testing

Central to the functionality of our prototype and final design is the ESP32-C6 microcontroller. The first test we performed was flashing the classic "blinky" test program, which blinks the LED on the development board. From this test alone, we learned that some of the third-party development boards we ordered for prototyping arrived dead or unable to flash. Next, the SCAN project uses many different aspects of the ESP32-C6, including the WiFi transceiver, PWM pins, host interfaces (SPI, I2C, and UART), and ADC channels. By testing each component in this chapter individually, we also verified the functionality of the ESP32-C6 itself. For development and debugging purposes, we referred to the ESP32-C6-WROOM-1 datasheet, which includes detailed accounts of features, GPIO functionality, and application circuits. [71]

9.1.2 Proximity Sensor Testing

Our team chose the AM312 passive-infrared proximity sensor for its low-power design and simple digital interface, which makes it ideal for detecting user presence and initiating a kiosk wake-up sequence from low-power mode. To test the hardware functionality of the AM312, we ordered a development module and set up a circuit with the sensor powered by 3.3V and GND connections. The AM312's signal output pin was connected to an external LED to visually confirm motion detection. From there, we manually tested the ability of the sensor to detect motion at varying ranges, ensuring it met our proximity detection range requirements for this project.

To test the prototyped firmware for the AM312, we created a component-specific PlatformIO project, connected the sensor's signal pin to a GPIO pin on the ESP32 development board, and uploaded our prototype firmware. Testing confirmed that both the external and onboard LEDs activated synchronously, validating the sensor's interrupt-driven motion detection functionality, response consistency, and accuracy.

9.1.3 NFC Transceiver Testing

For preliminary testing of the Mifare Classic cards, we used the IOS application NFC Tools to write specific NDEF records to a tag. This provided a user-friendly interface for formatting, clearing, writing, and reading tags. NFC Tools allows users to add records of different types and view the amount of space each record will take up on the NFC card. It also allows viewing of the specific memory contents and address space of any NFC card, which is extremely useful for debugging purposes.

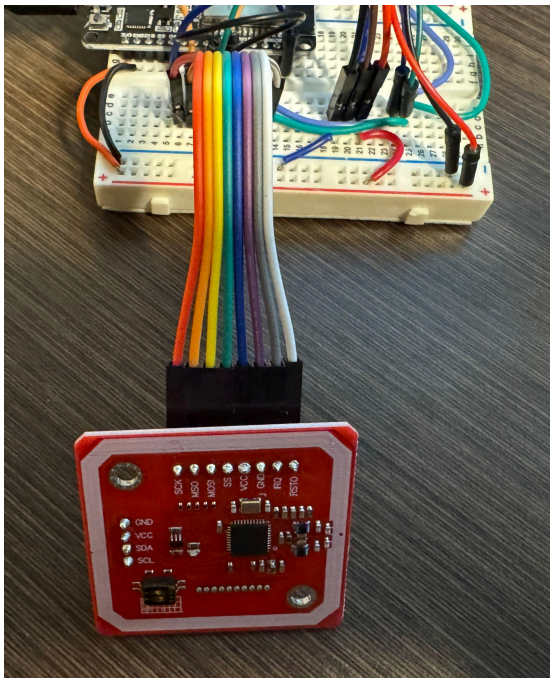


Figure 9.1 Breadboard Testing Setup for PN532 Development Board

To test the functionality of the PN532 transceiver module, we first connected the module to our ESP32 via a breadboard and jumper cables. The PN532 module supports high-speed UART, I2C, and SPI interfaces, and it was tested with both I2C and SPI configurations. From our testing, using the SPI interfacing yielded the most reliable and accurate results in terms of connection time, reading accuracy, and error rate. We first tested the variety of example programs included with the NDEF Library including FormatTag, ReadTag, and WriteTag. ReadTag identifies the tag type and UID, reads the entire memory contents of the tag, and dumps the information to the terminal via the serial monitor. The output of the ReadTag program for a card with 1 NDEF record (the user ID string) can be seen in Figure 9.1. In the output, we see that the tag type, Mifare

Classic, and its corresponding unique identifier (UID) are successfully detected, along with the single NDEF record for the user's ID.

```
Scan a NFC tag

Mifare Classic
UID: 4D 88 82 2F

This NFC Tag contains an NDEF Message with 1 NDEF Record.

NDEF Record 1
  TNF: 1
  Type: T
  Payload (HEX): 02 65 6E 35 39 33 37 35 33 37  .en5937537
  Payload (as String): ▫en5937537
```

Figure 9.2 Terminal Output of ReadTag Example Program

Figure 9.3 shows the serial monitor output for the WriteTag example program, which formats the Mifare Classic card as NDEF and writes one NDEF record, corresponding to the URL for the Adafruit website.

```
Place your Mifare Classic card on the reader to format with NDEF
and press any key to continue ...
---- Sent utf8 encoded message: "f\r\n" ----
Found an ISO14443A card
  UID Length: 4 bytes
  UID Value: 0x63 0x5E 0x7A 0x2F

Seems to be a Mifare Classic card (4 byte UID)
Card has been formatted for NDEF data using MAD1
Writing URI to sector 1 as an NDEF Message
NDEF URI Record written to sector 1


Done!
```

Figure 9.3 Terminal Output of WriteTag Example Program

After testing the core read, format, and write functions for the NDEF library, we were then able to adapt each example into our prototype software for the kiosk application.

9.1.3.1 Senior Design II Update

After migrating from the Arduino framework to using ESP-IDF, the use of MIFARE Classic RFID cards became much more troublesome, as the new PN532 library we selected had little support for MIFARE cards beyond providing the low-level device drivers. For this reason, in order to have the ability to read and write NDEF messages, we switched to using NTAG213 NFC Type 2 cards in our final project, which were cheaper and much easier to interface with our software.

9.1.4 Display Testing

Adafruit provides a standard graphics library test suite to showcase the various graphics APIs in the GFX library. For initial testing, we interfaced the TFT with our ESP32 using breadboard connections. We then flashed and ran the graphics test suite using the PlatformIO IDE as a development environment. From our testing, we observed that the number of pixels in the addressable and visible display regions were different, as discussed in Chapter 7. This required us to adapt our future designs to ensure that no text or graphics were clipped by the rounded display. Testing also revealed the physical variation in each display we ordered. Out of the three displays we ordered, one could not be written to, one's pixels were burnt out along the right border, and one was fully functional.

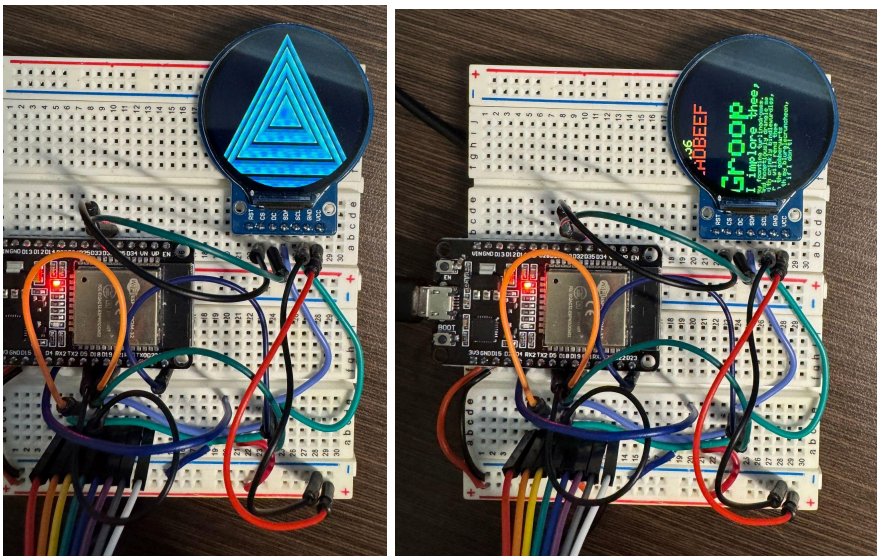


Figure 9.4 Display Graphics Test

9.1.4.1 Senior Design II Update

While the Adafruit graphics suite allowed us to test the physical functionality of the display, additional testing was needed after switching to the Light and Versatile Graphics Library to ensure that the display software worked as intended. LVGL provided graphics test suites very similar to the Adafruit graphics suite that we were able to use to verify performance.

9.1.5 Keypad Testing

To ensure the functionality of the keypad module prior to integration, we followed a three-step test procedure:

1. Verify the functionality of the keypad when connected directly to the Arduino GPIO pins.
2. Ensure operation with the Keypad library within our development environment.
3. Test the integration of the keypad module with the 8-bit IO expander using the I2C protocol.

Initial testing of the keypad was performed without the IO expander, directly wiring the eight select lines to the GPIO pins on the ESP32. The 8-pin header soldered onto the keypad module was tested for any accidental continuity between pins using a digital multimeter. Once connected to the ESP32, the keypad's functionality was tested using the Keypad library. The 8 GPIO pins were mapped to row and column indices, and ASCII values corresponding to the alphanumeric characters on the keypad were stored in a byte array. When a key was pressed, the corresponding ASCII value was printed to the serial monitor.

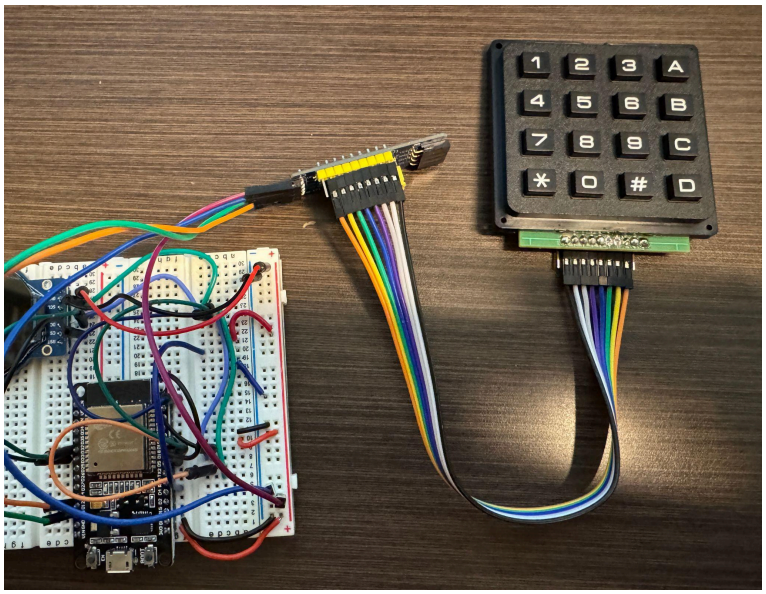


Figure 9.5 Breadboard Testing Setup for Keypad Testing with IO Expander

A similar procedure was followed to test the keypad using the PCF8574 IO bus expander module, which is interfaced using I2C. From software, we configured the proper I2C address of 0x20 and tested reading keypad presses over the bus by printing out an 8-bit value corresponding to the address of the pressed key. For example, the key in position (0,0) would register as 0b11111110. After successful testing with I2C addressing, the keypad was ready to be integrated into the check-in process.

9.1.7 WiFi Interface Testing

Developing Wi-Fi based applications was made very simple with the ESP32 and publicly available Wi-Fi libraries. To initially verify the Wi-Fi functionality on our ESP32 development boards, we performed a functional test using an example project where we simply had to enter a WiFi SSID and password and received serial feedback on connection status and details like our development board's assigned IP address as shown in Figure 9.6.

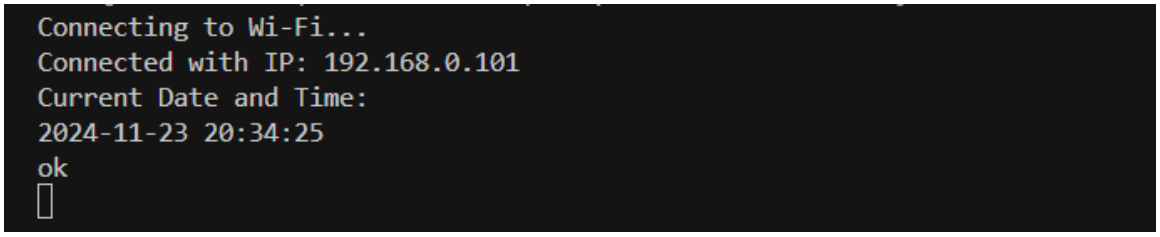


Figure 9.6 Wi-Fi Connection Test

With connectivity capabilities verified, we moved on to developing a test plan for ensuring Wi-Fi functionality in our final design. For this, we developed an on-board unit test that verified Wi-Fi connectivity using a ping request to a known server. We used pool.ntp.org for this, as it returns local date and time info that our system needs regardless. This test could be run on start-up or periodically throughout SCAN's up-time. On top of this test, we also designed a software test to ping our database server. These tests in combination provided a strong methodology for debugging connectivity issues.

9.1.8 Database Software Testing

Database testing focuses on the communication between the ESP32 and Firebase Realtime Database. Using custom API functions in a dedicated PlatformIO test project, we verified the database's ability to accurately log user check-ins/check-outs, update statuses, and retrieve data when needed. Future testing involved more in-depth testing, including unit and integration tests.

To begin manually testing the Firebase Realtime Database, we set up a custom PlatformIO project, including our custom-written driver API. We customized main.cpp to poll the serial terminal for user input and mapped key presses of "1", "2", and "3" to checkInUser, checkOut, and getUserData functions, respectively. We also set up a software-based timer to record response times for these database API functions. Using the serial terminal output window and a live view of the database via a web browser, we successfully verified our API's functionality and that authentication time requirements were met. Figure 9.7 shows these delay times, all of which meet our responsiveness requirements, while Figure 9.8 shows an example of the view from the database.

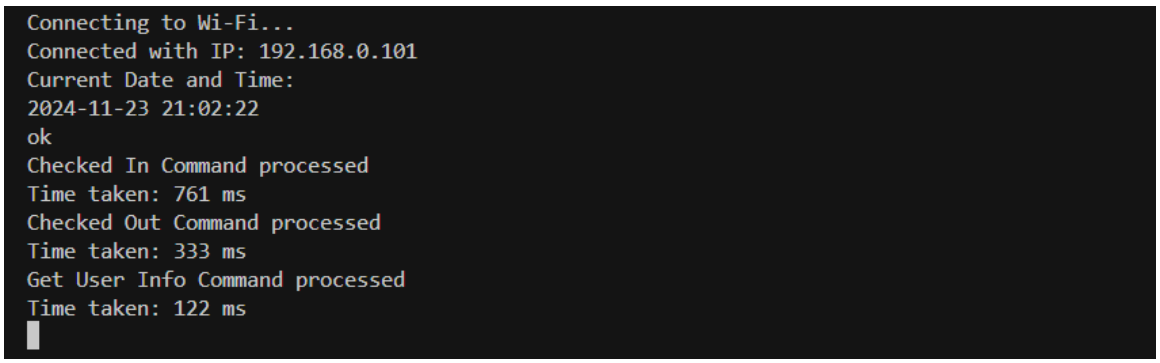


Figure 9.7 Database Interfacing Test - Terminal Output

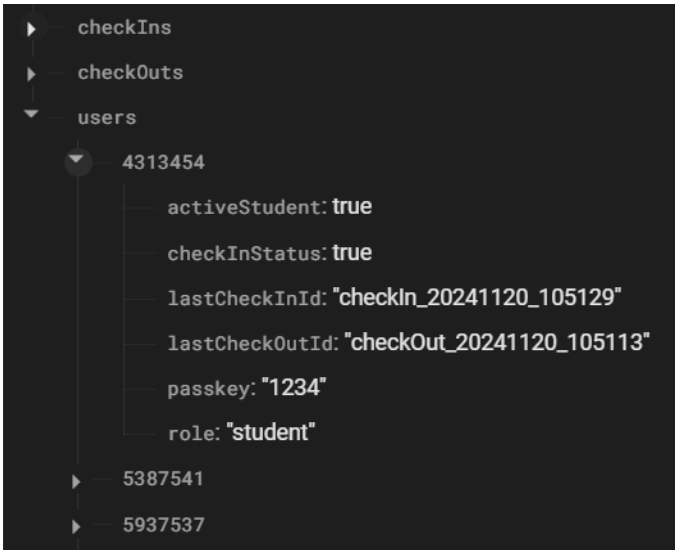


Figure 9.8 Database Interfacing Testing - Firebase Web-Viewer

We further verified the functionality and responsiveness of the database using Python test scripts. First, unit tests were written to verify the proper functionality of database API functions like checking in, checking out, and collecting user data. We also wrote functional tests, simulating real-time user interaction and confirming the capability of the database to handle high-frequency interaction accurately.

9.1.9 Web Interface Testing

The web interface software testing process ensured that the Data Analytics Dashboard operates reliably, accurately, and efficiently across different scenarios. Testing focused on verifying the functionality, performance, and compatibility of the web application. The testing process used both manual and automated techniques to validate real-time data handling, user interactions, and visualizations.

Regarding testing, there were a few primary objectives we had when it came to SCAN's web interface. First, we wanted to ensure real-time data updates from the Firebase Realtime Database were correctly reflected in the dashboard. We then wanted to verify the accuracy of computed analytics using these statistics, like number of attendees currently checked in, historical attendance records, average stay durations, and peak hours. We also wanted to verify the proper functionality of admin interaction with this endpoint for things like logging in and adding, editing, and deleting user profiles. Finally, we wanted to test a few smaller things like responsiveness on different screen sizes and platforms, and our application's response to fault conditions like losing connection. Proper testing methods for all of these aspects of our application ultimately brought about a more full-proof design that was easier to debug during development.

Functional Testing: With functional testing, we sought to verify proper execution of various features of our application, including ensuring that our analytics dashboard, activity log, and use management pages function as intended. This also included testing

of real-time database interface in which we wrote unit tests for the various API calls, ensuring proper feedback.

Performance Testing: Performance tests sought to verify various metrics about our application like how quickly data is fetched and displayed, and the responsiveness of our application during peak traffic. Response time tests were administered via scripting to verify response times met project requirements. Scripts were written to simulate high volumes of user traffic and use timer to ensure these requirements were still met.

Compatibility Testing: Compatibility testing was a bit easier to perform. Compatibility testing for this project consisted of accessing our web application via a multitude of browser and device combinations. We also used built in browser tools for simulating different screen sizes. Manual tests on these device and platform combinations ensured the application worked consistently across domains.

Error and Edge Case Testing: For added durability, we performed some simulated scenario edge case tests such as simulating a network failure, database access denial, and receiving corrupted user data to validate error handling mechanisms. Some other scenarios we tested for were very large data sets, empty database records, and unexpected user inputs.

Table 9.1 Web Interface Test Cases

Description	Expected Result	Status
Fetch real-time check-in data	Display correct number of attendees	Tested
Render peak attendance histogram	Correctly display data for a given day	Tested
Add a new user to the system	User profile successfully created	Tested
Access the web app on a mobile device	Responsive layout and full functionality	Tested
Simulate database disconnection	Show error message and retry option	Tested

To test the functionality of our web application, we used following frameworks and tools:

- **React Developer Tools:** Debugging and inspecting React component state and behavior.
- **Firebase Realtime Database Web-Viewer:** Simulating Firebase Realtime Database for controlled testing scenarios.
- **Browser Developer Tools:** Testing compatibility and performance across browsers.
- **Manual Testing Scripts and GitHub Actions:** Predefined GitHub Actions and unit tests to test specific features and edge cases.

9.1.10 PCB Testing

To support the testing of our PCB assembly, we incorporated several design features that enabled more efficient testing and debugging. Firstly, we broke out many of our unused GPIO pins to header pins. This allowed us to more easily debug logic signals by interfacing our PCB with DMMs and oscilloscopes. This also gave the added benefit of being able to expand our design. Secondly, we placed probe points in key areas around our PCB for easily verifying voltages like supply rails and grounds. Probe points were primarily placed around the supply inputs of various inputs for ensuring each received proper input voltages during operation. Additionally, probe points gave us an easy way for checking for things like shorts, continuity, and proper grounding.

For testing of our final design, we first verified all supply voltage rails output the expected voltages. With all supply voltages verified, we then tested connecting the USB C port to a computer to see if the USB device was recognized. We then ran individual component tests using our ESP-IDF project. If any were unsuccessful, we went into more in-depth probing and debugging. In some cases, such as our inability to share the SPI bus between the NFC transceiver and display, the extra GPIOs we broke out came in great assistance.

9.1.11 Requirements Testing

To prove that the final SCAN kiosk meets the requirements set out in our initial Divide and Conquer proposal, we evaluated our design against the requirement specifications set out in Chapter 2. In Table 8.2, a test plan is proposed for evaluating each requirement. By meeting or exceeding the requirements of each test, we demonstrated the SCAN kiosk's adherence to these specifications and the ability of SCAN's members to plan and execute a successful senior design project.

Table 9.2 SCAN Requirement Specifications and Test Plan

Requirement	**Specification**	**Test Plan**	**Status**
Maximum Enclosure Size	The maximum size of the enclosure and stand for each SCAN system will be **1'x1'x4'**	Design the enclosure such that the enclosure is within the desired dimensions.	Requirement met
Number of authentication methods	SCAN will support **two methods** of authentication: NFC tags and user ID entry through a keypad.	Test the functionality of NFC tag reader and keypad before and after system integration.	Requirement met
Minimum Battery Runtime	SCAN will have a minimum operating time of **2 hours** on battery	Measure charge levels of the battery using I2C commands between the	Requirement met

	power under typical operating conditions.	ESP32 and power management controller.	
Minimum Charging Power	SCAN will have a minimum supported charging power of **10W**.	Insert a USB C power meter between the kiosk and charger to measure the negotiated USB Power Delivery contract. The connected input device must support USB Power Delivery.	Requirement met
Time to Authenticate	SCAN will authenticate the credentials of a user ID in **less than 10 seconds**.	Test script will measure time to complete CheckInUser() database API function call.	Requirement met
Authentication Accuracy	SCAN will accurately authenticate users with an error rate of **5% or less.**	Testing of both authentications methods and simulated database interaction to determine error rate.	Requirement met
Minimum Proximity Sensing Range	SCAN will switch to active mode once a user is within **1.5 meters** of the system's range.	Test the detection range of the integrated PIR sensor in the final kiosk assembly.	Requirement met
Number of Check-in Statistics	SCAN will display **three forms of analytics** based on check-in data: occupancy, peak hours throughout the week, and the average duration of a user's stay.	Configure Python test script to simulate weekly user activity and verify against precomputed statistics.	Requirement met
Analytics Refresh Rate	The analysis of SCAN's collected data will be updated on the web page once **every minute**.	Manual tests will be conducted on design prototype, simulating user interaction and ensuring web-interface timing refresh requirement is met.	Requirement met

9.2 System Integration

Following the testing of each sub-component and sub-system, the hardware and software interfaces of each had to be integrated. The goal of integration was to maintain the functionality and reliability of each subsystem. Great care was taken when integrating components, as they may have shared hardware resources, bus interfaces, software dependencies, and physical space. The integration plan for the SCAN system can be seen in Figure 9.10. Each phase of integration is explored in this section.

Component Integration: In this phase, following component testing, we integrated the NFC reader module, TFT display, and PIR sensor to create a partially-functional check-in system. User IDs can be stored locally in the ESP32's volatile memory for processing of check-in and check-out information, but timestamps and analytics tracking were not available. To integrate codebases for each component, their PlatformIO projects must be merged and any dependencies must be included. This integration phase was completed in the last half of Senior Design 1.

Kiosk-Database Integration: After successful testing of a local check-in system, the SCAN prototype was integrated with the cloud database to authenticate users on a greater scale. The codebases for the ESP32-Firebase interface and NFC reader were merged to integrate the functionality of both applications. At this stage of development, user data can be stored entirely in the cloud, and the embedded kiosk application is able to query and retrieve authentication results in real time. This integration phase was completed at the end of Senior Design 1.

Power Integration: The power supply unit is a necessary part of all electronics designs. As such, the integration of our own power supply and power management circuitry were integrated directly into the SCAN PCB. For our Senior Design I prototype, the ESP32 development board includes its own power electronics to provide power. The circuitry for our power supply was integrated in the first iteration of the SCAN PCB. With the added complexity of battery charging, USB Power Delivery, and uninterrupted power to the kiosk, integrating the power system was not feasible to do within Senior Design 1, but it was successfully integrated during Senior Design 2.

Database-Web Interface Integration: Integration of the Firebase Realtime Database with the SCAN web app was a substantial step in ensuring proper functionality of the SCAN system. The Firebase Realtime Database acts as the central data repository for the SCAN system. The web application, built using React, communicates directly with the database through the Firebase Realtime Database SDK. This setup eliminates the need for an intermediary backend server, simplifying the architecture and reducing latency for real-time updates. The Firebase Realtime SDK uses event listeners to subscribe to specific datapaths, allowing the web application to react to changes in the remote database. This real-time update feature is primarily used to update the occupancy metric displayed on the analytics dashboard. To ensure efficient usage of the network, all other queries, like user management CRUD operations, are processed intermittently, either when a page is loaded, a button is pressed, or when a refresh timer has elapsed. Together,

integration of the Firebase Realtime Database provided SCAN's web application with the necessary data for real-time analytics display, user management operations, and activity logs.

Multi-Kiosk Integration: In the final phase of integration, we expanded SCAN from a single kiosk, single database system to a multi-kiosk, single-database system. To accomplish this, both kiosks needed to be in-sync and able to query the database in real-time. This phase required thorough testing of both kiosks individually to ensure functionality. We completed this integration phase in the latter half of Senior Design 2.

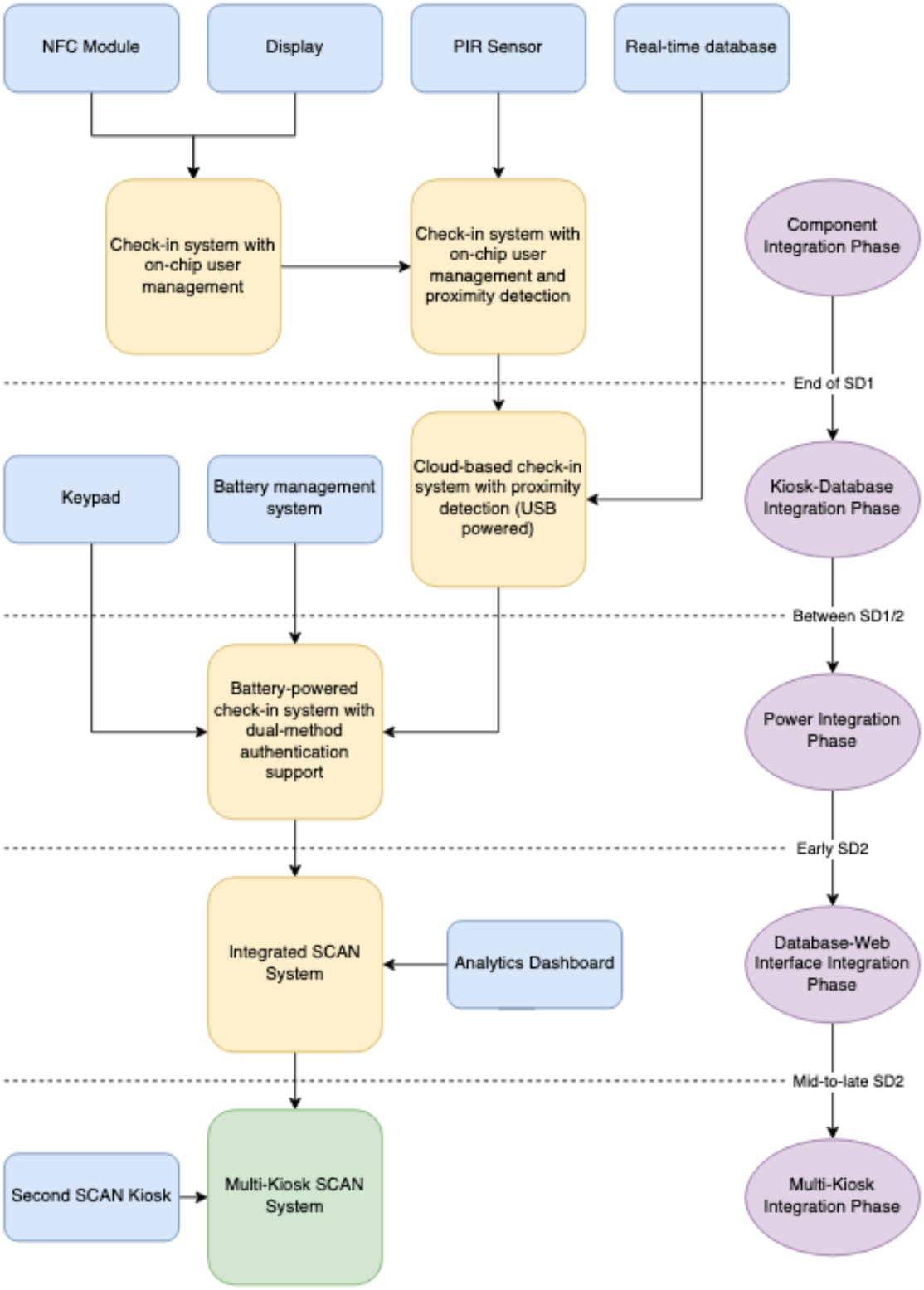


Figure 9.10 System Integration Timeline

10. ADMINISTRATIVE CONTENT

This section details our project's financial aspects, providing a budget and bill of materials for each individual SCAN kiosk. This section also details the project's timeline, using milestones and checkpoints to keep track of progress. Along with our project timeline, we have a work distribution section, where we assigned members to each subsystem of the project based on their technical expertise.

10.1 Project Budget and Financing

The budget for our project came from an even split of funding provided by each of the three group members. Table 10.1 breaks down the anticipated costs for each section of the project, including the cost for prototyping and end-product components. The goal was to keep the budget for each kiosk low to provide a cheaper alternative to more advanced smart check-in products. For this reason, each kiosk had a total budget of $120.00. Kiosk peripherals involve components such as the keypad, display, and NFC transceiver. The microcontroller is the cost of the ESP32-C6 itself. PCB, interconnects, and enclosure involve the costs associated with ordering the PCB assembly and fabricating the enclosure for the kiosk. Finally, power involves items such as the battery and USB-C cable.

Table 10.1 Per-Kiosk Budget Breakdown

Subsystem	**Budget**
Kiosk peripherals	$25.00
Microcontroller	$15.00
PCB, interconnects, and enclosure	$50.00
Power	$30.00
Total	$120.00

10.2 Bill of Materials

The bill of materials consists of all of the components we have purchased for the prototyping, testing, and development of this project. Multiple components were purchased in order to ensure enough parts were available for each group member to have a complete prototype. In the following bill of materials presented in Table 10.2, parts are divided into prototyping, PCB assembly, and enclosure costs. This BOM also does not take into account any redundant parts ordered in the event a component becomes damaged or inoperable. Finally, this BOM corresponds to the cost of developing **a single kiosk**. For the final presentation in Senior Design II, we demonstrated the functionality of two SCAN kiosks.

Table 10.2 Project Bill of Materials

Component	Description	Qty	Unit Cost	Total Cost	Status	Link
Major/Prototyping Components						
ESP32-C6	ESP32-C6 Development Board	1	$9.00	$9.00	Acquired	Link
NFC Tag Reader	PN532 NFC NXP RFID Module V3 Kit NFC Reader Module	1	$8.99	$8.99	Acquired	Link
NFC Tags	10 pcs Mifare Classic 1k Smart Cards	1	$9.99	$9.99	Acquired	Link
TFT Display	1.28 Inch TFT 3-pack LCD Display Module	1	$18.99	$18.99	Acquired	Link
Keypad	4x4 Matrix Membrane Keypad 16 Key Module	1	$7.89	$7.89	Acquired	Link
IO Expander	8PCS PCF8574 IO Expansion Board Module	1	$9.99	$9.99	Acquired	Link
Proximity Sensor	HiLetgo 3pcs AM312 Mini Pyroelectric PIR	1	$8.69	$8.69	Acquired	Link
Buzzer	Passive Electronic Alarm Buzzer 10pc	1	$5.99	$5.99	Acquired	Link
LEDs	Red, green, and blue status indicator LEDs	3	N/A	N/A	Acquired	
$LiFePO_4$ Battery	Talentree 6V $LiFePO_4$ Battery	1	$21.99	$21.99	Acquired	Link

Interconnects	Breadboards, jumpers, and pin headers	1	N/A	N/A	Already Owned	
Prototyping Cost				$108.41		
PCB Assembly Components (Assembly costs, ICs, etc)						
PCB Fabrication and Assembly	Estimated costs, JLCPCB (Not including components)	1	$15	$15	Acquired	Link
ESP32-C6 IC	ESP32-C6-WROOM-1-N8 SoC	1	$10.56	$10.56	Acquired	Link
IO Expander	PCF8574P	2	$0.65	$1.30	Acquired	Link
USB-C	16-pin USB Type C Connector	1	$0.06	$0.06	Acquired	Link
GC9A01 Display	1.28” Round Display (Display only)	1	$6.04	$6.04	Acquired	Link
Buck Converter	LM62460Q3RPHRQ1	1	$3.57	$3.57	Acquired	Link
Power Management Controller	BQ25798RQMR	1	$5.27	$5.27	Acquired	Link
Power Sink Controller	CYPD3177-24LQXQ (USB PD)	1	$1.13	$1.13	Acquired	Link
Discharge Protection	ESD122	1	$0.04	$0.04	Acquired	Link
USB-to-UART Bridge	CP2102N-A02-xQFN28	1	$2.38	$2.38	Acquired	Link
PN532	PN5321A3HN/C106,55	1	$7.21	$7.21	Acquired	Link
Quartz Crystal	27.12 MHz Oscillator	1	$0.08	$0.08	Acquired	Link

Passive/ Active circuit elements	Resistors, capacitors, LEDs, etc	Var.	N/A	N/A	Acquired	
PCBA Estimated Total				$53.00		
Kiosk Enclosure Materials						
Enclosure Materials	OVERTURE PLA Filament PLA, 1Kg	1	$17.99	$17.99	Acquired	Link
PVC Pipe	PVC Pipe 1” White Custom Length 4ft	1	$14.99	$14.99	Acquired	Link
Screws and other fixtures	*Estimated cost*	1	$5.00	$5.00	Acquired	
Total Cost (Prototyping and Final)				$191.14		
- **Prototyping Cost**				$81.48		
Total Cost (Final Kiosk)				$109.66		

With the bill of materials listed above, the kiosk peripherals for the final design (display, keypad, and NFC transceiver) came to a cost of $21.14. The package for the ESP32-C6 was $10.56, and the sum total of the PCB. interconnects, and enclosure came out to roughly $46.00, not accounting for JLCPCB discounts on first-time manufacturing and assembly. Finally, for the power budget, which consists of the battery and power management ICs, the project cost came to $31.96, which was slightly above our budget of $30.00. However, the cost to source the power management ICs directly from JLCPCB is higher than if we chose to order from another electronics vendor, so it is possible to mitigate these costs. Overall, the costs for our final design were within our budget of $120.00 per kiosk, and we met our goal of offering an affordable kiosk solution.

10.2.1 Senior Design II Updated Budget

As we developed our project in Senior Design II, the SCAN team encountered some additional costs that we did not anticipate. This included pricey shipping and assembly costs for our PCBs from JLCPCB and some additional material costs for the kiosk enclosure. We also needed a higher capacity battery, which led to an increase in price for our power subsystem. Should this project ever be commercialized, there are steps we can take to reduce our bill of materials and bring it closer to the $120.00 budget we set in Senior Design 1.

10.3 Project Milestones

The SCAN team was initially formed over the summer to begin preparation and planning for Senior Design I. At the beginning of the fall semester, the team brainstormed and finalized a project idea, and a set of milestones and objectives was planned for the Fall and Spring semesters to ensure that the project remains on track. In addition to **Tables 10.4-10.5**, the team used a Gantt chart that was automatically updated with newly created issues on the project's GitHub repository.

Table 10.4 Senior Design 1 Milestones

Documentation Milestones			
Task	**Start Date**	**End/Due Date**	**Status**
Brainstorm projects and ideas	8/18/2024	8/25/2024	Completed
Select the project and define the scope	8/30/2024	9/6/2024	Completed
Submit the initial divide-and-conquer 10-page document (Project proposal, objectives, administrative content)	8/30/2024	9/6/2024	Completed
Attend Divide and Conquer revision meeting with Dr. Wei	9/9/2024	9/11/2024	Completed
Upload revised D&C report onto the group website	9/20/2024	9/27/2024	Completed
Reach 45-page document milestone (Technology research, component selection, system requirements)	8/30/2024	10/12/2024	Completed
Reach 45-page document milestone and submit draft (Design constraints, ChatGPT comparison)	8/30/2024	10/25/2024	Completed
Attend the 45-page document meeting and receive feedback from reviewers	10/28/2024	10/30/2024	Completed
Upload the revised 45-page report to the project website	11/1/2024	11/8/2024	Completed
Upload draft SD1 final report and mini demo video to website	11/18/20243	11/26/2024	Completed
Submit SD1 final report on webcourses	11/26/2024	11/26/2024	Completed

Project Milestones			
Task	**Start Date**	**Expected End Date**	**Status**
Initial system design and breadboarding	10/21/2024	10/28/2024	Completed
Initial system testing and revision	10/28/2024	11/4/2024	Completed
First integration of prototyped hardware and software	11/4/2024	11/15/2024	Completed
Schematic capture and preliminary PCB design	11/08/2024	11/26/204	Completed

Table 10.5 Senior Design 2 Project Milestones

Task	**Start Date**	**Expected End Date**	**Status**
Ordering and testing of PCB assembly	1/6/2025	1/13/2025	Completed
Software/hardware integration with server application (Database and frontend)	1/13/2025	2/10/2025	Completed
Design revisions based on testing (if necessary)	2/10/2025	2/17/2025	Completed
Mock project demonstration and presentation	3/24/2025	3/31/2025	Completed
Final testing and documentation	3/31/2025	4/7/2025	Completed
Final presentation and demo submission	4/15/2025	4/15/2025	Completed
Final report submission	4/22/2025	4/22/2025	Completed

10.4 Work Distribution

Table 10.6 presents the workload distribution between the three group members. Each group member had primary responsibility, which are roles and subsystems that they were directly responsible for, and secondary responsibilities, where they provided assistance in designing and testing each subsystem. While each table lists the specific responsibilities of each member, there was communication and cooperation at each stage of the design process to ensure that each subsystem was seamlessly integrated into the SCAN system.

Table 10.6 Work Distribution

Cory Brynds | Computer Engineering	
Primary Responsibilities	**Secondary Responsibilities**
Embedded software development for NFC transceiver, SPI, display, keypad, buzzer, RGB LEDs, state control, and admin mode	Project manager and administrative documentation
Kiosk enclosure design	Senior Design website development and maintenance
DeAndre Hendrix | Computer Engineering	
Primary Responsibilities	**Secondary Responsibilities**
Web app design: analytics dashboard, user management page, and activity log	Embedded software development: database check-in logic, proximity sensor
Database structure and web app integration	Video editing
Jonah Sprandel | Electrical Engineering	
Primary Responsibilities	**Secondary Responsibilities**
Schematic design and PCB layout	Web app: kiosk management page
Embedded software development: HTTP, Wi-Fi, MQTT, OTA updates, I2C, USB PD, power management	Framework/DevOps: Docker, ESP-IDF, CI/CD pipeline

11. CONCLUSION

Over the course of Senior Design I, we witnessed the SCAN project transition from a vague idea to a concrete specification of requirements, hardware technologies, and software infrastructure. Through our technology review, we were exposed to a number of devices, standards, and protocols, which helped to inform our component selection and the design of the hardware/software subsystems for the project.

All major components for prototyping the SCAN system were then acquired, including the ESP32-C6 microcontroller, PIR sensor, NFC reader/writer module, TFT display, indicator LEDs and buzzer, keypad, battery, and enclosure materials. We developed a successful breadboard prototype capable of the core check-in functionalities layed out in our objectives, goals, and requirement specifications. Each major component was tested, and the software interfaces were written and validated on our microcontroller and database.

Our primary objective following the conclusion of Senior Design I was to translate our full schematic design to a printed circuit board to house our microcontroller, power circuit, and peripheral interconnects. We intended to order this PCB over winter break to identify any potential design flaws prior to the start of Senior Design II. We also continued development on the web interface design for our analytics dashboard. With our work in Senior Design I, we laid the infrastructure to successfully complete our project in Senior Design II. We also learned many valuable lessons about project management and time constraints, which we took into account as we developed the project in Senior Design II.

In Senior Design II, we followed the infrastructure we set up the previous semester and saw the project through to completion. Although some roadblocks presented themselves, such as a complex PCB design and the migration of our software to ESP-IDF, we persevered and met all of our requirement specifications. We met and exceeded each of our basic and advanced goals, and we were able to complete ⅔ of our stretch goals as well. After demoing our project to our reviewers and judges at the Senior Design Showcase, we received multiple comments about the potential commercialization of the SCAN system. This is a testament to the significance of the project we set out to solve and the quality of our implementation.

On a final note, we would like to thank Dr. Wei and Dr. Chan for their guidance on our project and in the Senior Design lecture series. We would also like to thank our reviewers, Dr. Sonali Das, Dr. Mike Borowczak, and Dr. Piotr Kulik, for taking the time to provide us with valuable feedback on the project and ultimately evaluate our final report and design.

APPENDIX A - REFERENCES

[1] "Envoy Visitors," Envoy. Accessed: Oct. 24, 2024. [Online]. Available: https://envoy.com/products/visitors

[2] "Check in attendees at your event with the Eventbrite app for organizers," Eventbrite Help Center. Accessed: Oct. 24, 2024. [Online]. Available: https://www.eventbrite.com/help/en-us/articles/741083/how-to-check-in-attendees-at-the-event-with-eventbrite-organizer/

[3] "Gym Kiosk," Empower Your Members with Self-Service Solutions. Accessed: Oct. 24, 2024. [Online]. Available: https://www.perfectgym.com/en/features/gym-kiosk

[4] J. Schneider and I. Smalley, "Microcontroller," IBM. Accessed: Oct. 25, 2024. [Online]. Available: https://www.ibm.com/think/topics/microcontroller

[5] K. Obuszewski, "When To Use FPGA vs Microcontroller," VORAGO Technologies. Accessed: Oct. 25, 2024. [Online]. Available: https://www.voragotech.com/blog/fpga-vs-microcontroller

[6] "AMD Technical Information Portal." Accessed: Oct. 24, 2024. [Online]. Available: https://docs.amd.com/r/en-US/ug901-vivado-synthesis/Choosing-Between -Distributed-RAM-and-Dedicated-Block-RAM

[7] "System on a Chip (SoC) Explained - Revolutionizing Electronics," Lenovo US. Accessed: Oct. 25, 2024. [Online]. Available: https://www.lenovo.com/us/en/glossary/system-on-a-chip/?orgRef=https%253A%252F%252Fwww.google.com%252F&srsltid=AfmBOor2t6th2AD-xIbMrs7y9tqA_tof-o6n24YxeCm_paF3VvosWbOT

[8] "CC3235SF," TI.com. Accessed: Oct. 25, 2024. [Online]. Available: https://www.ti.com/product/CC3235SF

[9] ESPRESSIF, "ESP32-C6 Series," ESPRESSIF. Accessed: Oct. 24, 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-c6_datasheet_en.pdf

[10] I. T. AG, "CYW55912 - Infineon Technologies," Infineon Technologies AG. Accessed: Oct. 25, 2024. [Online]. Available: https://www.infineon.com/cms/en/product/ wireless-connectivity/airoc-connected-mcu/cyw55912/

[11] MaxBotix, "What Is an Ultrasonic Sensor?," MaxBotix, Mar. 01, 2023. Accessed: Oct. 25, 2024. [Online]. Available: https://maxbotix.com/blogs/blog/how-ultrasonic- sensors-work

[12] T. I. [SLAA907,D] Incorporated, “Ultrasonic Sensing Basics (Rev. D)”.

[13] J. S. Cook, “The Right Tool for the Job: Active and Passive Infrared Sensors,” Arrow.com, Sep. 11, 2018. Accessed: Oct. 25, 2024. [Online]. Available: https://www.arrow.com/en/research-and-events/articles/understanding-active-and-passive-infrared-sensors

[14] T. I. [SLAA907,D] Incorporated, “Ultrasonic Sensing Basics (Rev. D)”.

[15] S. Tzur-David, “Software Vs Hardware Tokens - The Complete Guide,” Secret Double Octopus. Accessed: Oct. 24, 2024. [Online]. Available: https://doubleoctopus .com/blog/access-management/tokens-hard-soft-and -whats-in-between/

[16] J. Moermond, “What Is a Capacitive Sensor?,” Balluff. Accessed: Oct. 25, 2024. [Online]. Available: https://www.balluff.com/en-us/blog/what-is-a-capacitive -sensor

[17] MPJAuser, “http://www.aliexpress.com/store/product/wholesale -HC-SR501 -HCSR501-Human-body-sensor-module-and-pyroelectric -infrared-sensor-fr”.

[18] “PIR Motion Sensor Module AM312,” TechMaze. Accessed: Oct. 25, 2024. [Online]. Available: https://techmaze.romman.store/product/99187464

[19] “Grove - PIR Motion Sensor,” Arduino Online Shop. Accessed: Oct. 25, 2024. [Online]. Available: https://store-usa.arduino.cc/products/grove-pir-motion-sensor

[20] “What is Password-Based Authentication,” Descope. Accessed: Oct. 24, 2024. [Online]. Available: https://www.descope.com/learn/post/password-authentication

[21] Descope, “Biometric Fingerprint Authentication Explained,” Descope. Accessed: Oct. 23, 2024. [Online]. Available: https://www.descope.com/learn/post/ fingerprint -authentication

[22] “How Much Does a Fingerprint Scanner Cost?,” What is the Price? Accessed: Oct. 24, 2024. [Online]. Available: https://www.howmuchisit.org/fingerprint -scanner -cost/

[23] “The Ultimate Guide to Hardware Authentication Devices,” Kitsap Networking Services. Accessed: Oct. 24, 2024. [Online]. Available: https://www.kitsapnet working .com/post/the-ultimate-guide-to-hardware- authentication-devices

[24] M. Brain and T. Homer, “What is WiFi and How Does it Work?,” HowStuffWorks, Apr. 30, 2001. Accessed: Oct. 25, 2024. [Online]. Available: https://computer.howstuffworks.com/wireless-network.htm

[25] “What are LoRa and LoRaWAN?,” The Things Network. Accessed: Oct. 25, 2024. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

[26] Onomondo, “Onomondo,” Onomondo. Accessed: Oct. 25, 2024. [Online]. Available: https://onomondo.com/blog/lte-m-iot-guide/

[27] Center for Devices and R. Health, “Radio Frequency Identification (RFID),” U.S. Food and Drug Administration. Accessed: Oct. 24, 2024. [Online]. Available: https://www.fda.gov/radiation-emitting-products/electromagnetic-compatibility-emc/radio-frequency-identification-rfid

[28] C. Wong, “Nexqo - RFID & NFC Solution Provider,” Nexqo - RFID & NFC Solution Provider. Accessed: Oct. 06, 2024. [Online]. Available: https://nexqo.com/2023/11/rfid-communication-protocols/

[29] M. Afaneh, “Bluetooth Low Energy (BLE): A Complete Guide,” Novel Bits. Accessed: Oct. 24, 2024. [Online]. Available: https://novelbits.io/bluetooth-low-energy -ble-complete-guide/

[30] NXP Semiconductors, “PN532/C1,” Product Data Sheet. NXP, pp. 1–5, Nov. 28, 2017. Accessed: Oct. 24, 2024. [Online]. Available: https://www.nxp.com/docs/en/nxp /data-sheets/PN532_C1.pdf

[31] STMicroelectronics, “ST25R3911B,” Datasheet. STM, pp. 1–5, Apr. 01, 2022. Accessed: Oct. 24, 2024. [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/datasheet/group3/ba/60/79/5c/bb/a9/41/76/DM00321525/files/DM00321525.pdf/jcr:content/translations/en.DM00321525.pdf

[32] NXP Semiconductors, “CLRC663,” Product Data Sheet. NXP, pp. 1–5, Jan. 03, 2024. Accessed: Oct. 24, 2023. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/CLRC663.pdf

[33] “What is an LCD? LCD technology & Types of Display,” Orient Display. Accessed: Oct. 06, 2024. [Online]. Available: https://www.orientdisplay.com/knowledge -base/lcd-basics/what-is-lcd-liquid-crystal-display/

[34] “LED, IPS, OLED or TFT - Differences & Comparison,” Orient Display. Accessed: Oct. 24, 2024. [Online]. Available: https://www.orientdisplay.com/knowledge-base/lcd-basics/lcd-vs-tft-ips-led-oled-display/

[35] Leaktek Display, “What is difference between TFT and LCD?,” Leadtek Display. Accessed: Oct. 24, 2024. [Online]. Available: https://www.leadtekdisplay.com/what -is-difference-between-tft-and-lcd-a-1071.html

[36] “OLED-Info,” OLED-Info. Accessed: Oct. 24, 2024. [Online]. Available: https://www.oled-info.com/oled-technology

[37] “What Is A Relational Database (RDBMS)?,” Google Cloud. Accessed: Oct. 25, 2024. [Online]. Available: https://cloud.google.com/learn/what-is-a-relationa l-database

[38] “Which NoSQL database is right for you?,” Amazon Web Services, Inc. Accessed: Oct. 25, 2024. [Online]. Available: https://aws.amazon.com/nosql/

[39] “Google Cloud Databases,” Google Cloud. Accessed: Oct. 25, 2024. [Online]. Available: https://cloud.google.com/products/databases

[40] “Firestore documentation ,” Google Cloud. Accessed: Oct. 25, 2024. [Online]. Available: https://cloud.google.com/firestore/docs

[41] C. Studio, “What Is MongoDB?,” Pure Storage, Oct. 19, 2021. Accessed: Oct. 25, 2024. [Online]. Available: https://www.purestorage.com/uk/knowledge/what-is-mongodb .html

[42] martinekuan, “How does Azure work? - Cloud Adoption Framework,” Microsoft Learn. Accessed: Oct. 25, 2024. [Online]. Available: https://learn.microsoft .com/en-us/ azure/cloud-adoption-framework/get-started/what-is-azure

[43] NFC Forum, “Specifications,” NFC Forum. Accessed: Oct. 24, 2024. [Online]. Available: https://nfc-forum.org/build/specifications

[44] Lady Ada, “Adafruit PN532 RFID/NFC Breakout and Shield,” Adafruit Learning System. Accessed: Oct. 25, 2024. [Online]. Available: https://learn.adafruit.com/adafruit- pn532-rfid-nfc/ndef

[45] C. Ruth, “The Evolution of Wi-Fi Technology and Standards,” IEEE Standards Association. Accessed: Oct. 25, 2024. [Online]. Available: https://standards .ieee.org/ beyond-standards/the-evolution-of-wi-fi-technology-and-standards/

[46] NXP, “I2C-bus specification and user manual.” Accessed: Oct. 25, 2024. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[47] Electrical and Computer Engineering Department, “Embedded Systems Lab 7: Inter-Integrated Circuits (I2C) Communication,” Electrical and Computer Engineering Department, University of Central Florida.

[48] F. Duarte, “Number of ChatGPT Users (Oct 2024),” Exploding Topics. Accessed: Oct. 24, 2024. [Online]. Available: https://explodingtopics.com/blog/chatgpt-users

[49] “ChatGPT.” Accessed: Oct. 24, 2024. [Online]. Available: https://chatgpt.com

[50] “Perplexity,” Perplexity AI. Accessed: Oct. 24, 2024. [Online]. Available: https://www.perplexity.ai/

[51] “Understanding the Basics of Infrared Communications,” DigiKey. Accessed: Oct. 24, 2024. [Online]. Available: https://www.digikey.com/en/maker/tutorials/2021/understanding-the-basics-of-infrared-communications

[52] P. Dhaker, “Introduction to SPI Interface,” Analog Devices, Inc. Accessed: Oct. 25, 2024. [Online]. Available: https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html

[53] “USB Charger (USB Power Delivery),” USB-IF. Accessed: Oct. 25, 2024. [Online]. Available: https://www.usb.org/usb-charger-pd

[54] Texas Instruments, “USB PD Power Negotiations.” Accessed: Oct. 25, 2024. [Online]. Available: https://www.ti.com/lit/an/slva842/slva842.pdf

[55] T. C. Frankel, “The Cobalt Pipeline,” Washington Post. Accessed: Oct. 25, 2024. [Online]. Available: https://www.washingtonpost.com/graphics/business/batteries/congo-cobalt-mining-for-lithium-ion-battery/

[56] “Lead Acid vs Lithium Batteries: Understanding the Differences - GME Recycling,” GME Recycling -. Accessed: Oct. 25, 2024. [Online]. Available: https://www.gme-recycling.com/what-is-the-difference-between-lead-acid-and-lithium-batteries/

[57] “Lithium-Ion Battery,” Clean Energy Institute. Accessed: Oct. 25, 2024. [Online]. Available: https://www.cei.washington.edu/research/energy-storage/lithium-ion-battery/

[58] Jauch, “Introduction to Lithium Polymer Battery Technology,” Jauch. Accessed: Oct. 25, 2024. [Online]. Available: https://www.jauch.com/downloadfile/5c5050fa5b6510e9a8ad76299baae4e53/white_paper_introduction_to_lipo_battery_technology_11-2018_en.pdf

[59] “Lithium Iron Phosphate - an overview,” ScienceDirect Topics. Accessed: Oct. 25, 2024. [Online]. Available: https://www.sciencedirect.com/topics/engineering/lithium-iron-phosphate

[60] “BU-107: Comparison Table of Secondary Batteries,” Battery University. Accessed: Oct. 25, 2024. [Online]. Available: https://batteryuniversity.com/article/bu-107-comparison-table-of-secondary-batteries

[61] P. Neilson, “Unpacking Toyota’s Continued Use of NiMH Batteries,” Torque News. Accessed: Oct. 25, 2024. [Online]. Available: https://www.torquenews .com/8113/unpacking-toyotas-continued-use-nimh-batteries

[62] M. Levy, “Understanding the Real Energy Consumption of Embedded Microcontrollers,” *Convergence Promotions LLC*, Jun. 13, 2012. Accessed: Oct. 25, 2024. [Online]. Available: https://www.digikey.com/en/articles/understanding -the-real-energy-consumption-of-embedded-microcontrollers

[63] B. Patel, “Best Front End Frameworks You Must Try in 2024 [+Comparison],” Space-O Technologies. Accessed: Nov. 24, 2024. [Online]. Available: https://www.spaceotechnologies.com/blog/front-end-frameworks-comparison/.

[64] A. K, “What is ReactJs and use cases of ReactJs?,” DevOpsSchool.com. Accessed: Nov. 24, 2024. [Online]. Available: https://www.devopsschool.com /blog/what-is-reactjs-and-use-cases-of-reactjs/

[65] J. Smith, “NFC vs Mifare: The Battle of Contactless Technologies,” BAS-IP. Accessed: Nov. 24, 2024. [Online]. Available: https://bas-ip.com/articles /nfc-vs-Mifare-battle/

[66] R. Habraken, "PN532 Antenna Design," TechnoCentrum, Radboud University Nijmegen, Document Version 1.0, May 17, 2011.

[67] NXP Semiconductors, "PN532 Near Field Communication (NFC) controller," Product data sheet, Rev. 3.6, Nov. 28, 2017.

[68] STMicroelectronics, "ST25R3911B - High performance HF reader / NFC initiator with 1.4 W supporting VHBR and AAT," Datasheet, DS11793, Rev. 7, Apr. 2022. [Online]. Available: https://www.st.com/en/product/st25r3911b.

[69] NXP Semiconductors, "CLRC663 - High Performance Multi-Protocol NFC Frontend," Product data sheet, Rev. 5.3, Jan. 3, 2024. [Online]. Available: https://www.nxp.com.

[70] Espressif Systems, *OTA Workflow Diagram*, GitHub repository, 2024. [Online]. Available: https://github.com/espressif/esp-idf/blob/v5.2.3/examples/system/ota/ ota_workflow.png. [Accessed: Nov. 26, 2024].

[71] Espressif Systems, *ESP32-C6-WROOM-1 & ESP32-C6-WROOM-1U Datasheet*, Version 1.1, 2024. [Online]. Available: https://espressif.com/documentation /esp32-c6 -wroom-1_wroom-1u_datasheet_en.pdf. [Accessed: Nov. 26, 2024].

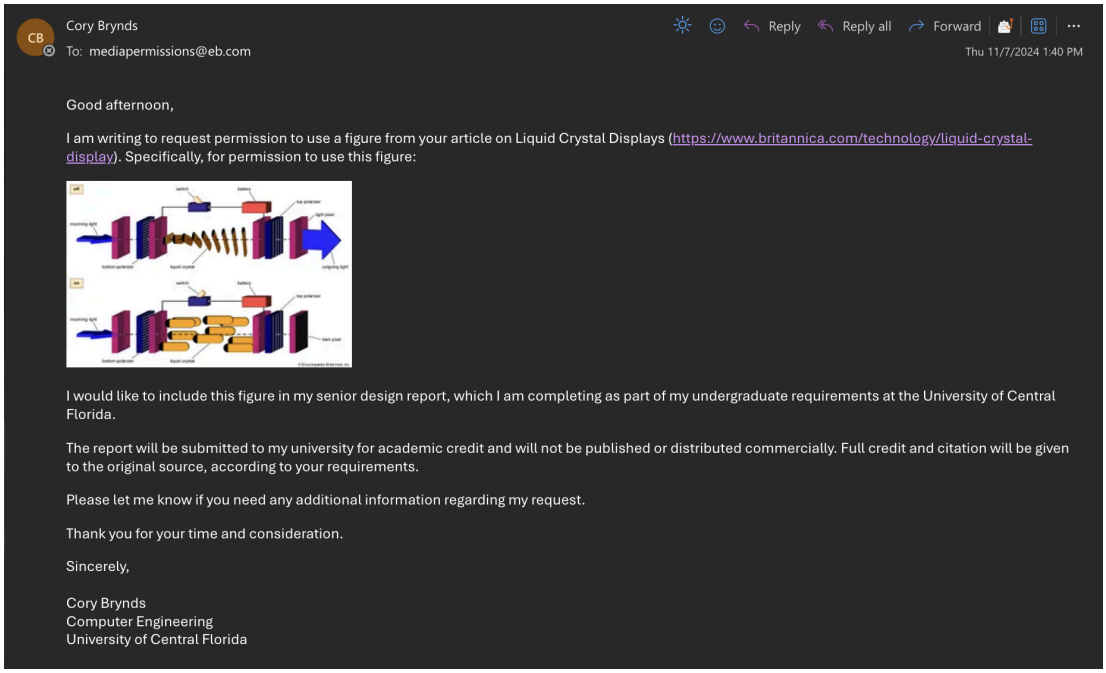
[72] Infineon, *EZ-PD BCR Datasheet*, 2024. [Online]. Available: https://www.mouser.com/datasheet/2/196/Infineon_EZ_PD_BCR_Datasheet_US B_Type_C_Port_Contr-3361317.pdf [Accessed: Nov. 18, 2024].

[73] Texas Instruments, *BQ25798 Datasheet*, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/bq25798.pdf?ts=1731924735062&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ25798 [Accessed: Nov. 20, 2024].

[74] Texas Instruments, *LM62460-Q1 Datasheet*, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm62460-q1.pdf?ts=1732533455472 [Accessed: Nov. 20, 2024].

[75] Espressif, *ESP32-C6-DevKitC-1 Schematics,* V1.4, 2024. [Online]. Available: https://docs.espressif.com/projects/esp-dev-kits/en/latest/_static/esp32-c6-devkitc-1/schematics/esp32-c6-devkitc-1-schematics_v1.4.pdf [Accessed: Nov. 20, 2024].

APPENDIX B - COPYRIGHT PERMISSIONS



Cory Brynds
To: mediapermissions@eb.com

Reply | Reply all | Forward

Thu 11/7/2024 1:40 PM

Good afternoon,

I am writing to request permission to use a figure from your article on Liquid Crystal Displays (https://www.britannica.com/technology/liquid-crystal-display). Specifically, for permission to use this figure:

I would like to include this figure in my senior design report, which I am completing as part of my undergraduate requirements at the University of Central Florida.

The report will be submitted to my university for academic credit and will not be published or distributed commercially. Full credit and citation will be given to the original source, according to your requirements.

Please let me know if you need any additional information regarding my request.

Thank you for your time and consideration.

Sincerely,

Cory Brynds
Computer Engineering
University of Central Florida

Request for image of Liquid Crystal Display Composition (Figure 3.2)

Email address*

Please provide the email address that you registered with RS DesignSpark.

co224015@ucf.edu

Topic*

Please select the topic that your request concerns. For software, please choose which program and its version number (available in your software Help -> About menu).

Website

Website*

Please choose the DesignSpark website issue you need support for.

Other

Subject*

A short summary of your request.

Permission to Use Figure in Senior Design Report

Description*

Describe your issue or question. This field is a text only field, please add images as attachments.

I am writing to request permission to use a figure from your article "All you need to know about TFT displays" (https://www.rs-online.com/designspark/all-you-need-to-know-about-tft-displays). Specifically, for permission to use the figure depicting the TFT display composition. I would like to include this figure in my senior design report, which I am completing as part of my undergraduate requirements at the University of Central Florida.

The report will be submitted to my university for academic credit and will not be published or distributed commercially. Full credit and citation will be given to the original source, according to your requirements.

Please let me know if you need any additional information regarding my request.

Thank you for your time and consideration.

Sincerely,

Cory Brynds
Computer Engineering
University of Central Florida

Request for image of Thin-Film Transistor display composition (Figure 3.3)

Your name *

Cory Brynds

Your email address *

co224015@ucf.edu

Subject *

Permission to Use Figure in Senior Design Report

Message *

Good afternoon,

I am writing to request permission to use a figure from your article on OLED Displays (https://www.oled-info.com/oled-technology).

I would like to include the figure detailing the composition of an OLED display in my senior design report, which I am completing as part of my undergraduate requirements at the University of Central Florida.

The report will be submitted to my university for academic credit and will not be published or distributed commercially. Full credit and citation will be given to the original source, according to your requirements.

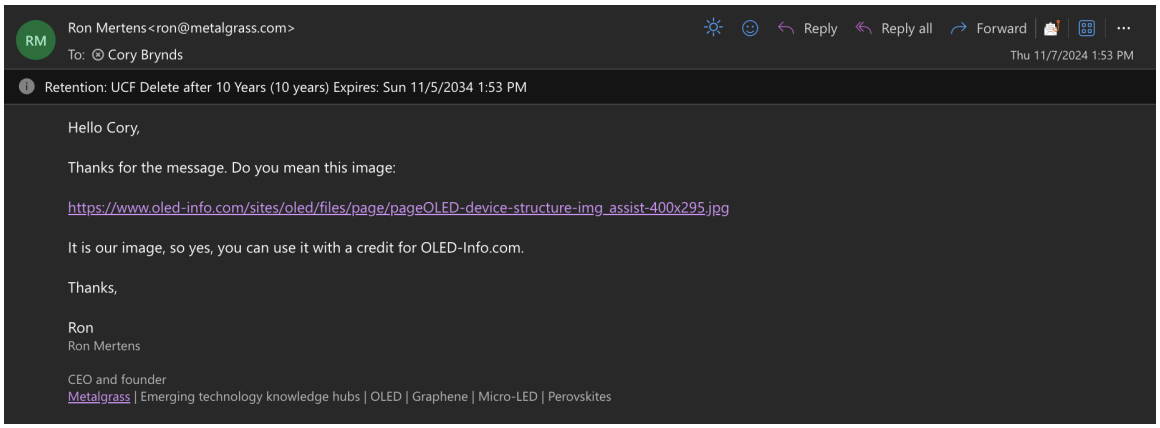
Please let me know if you need any additional information regarding my request.

Thank you for your time and consideration.

Sincerely,

Cory Brynds
Computer Engineering
University of Central Florida

Request for image of OLED display composition (Figure 3.4)



Approval for use of OLED display image (Figure 3.4)

Contact Us

NFC Forum
401 Edgewater Place, Suite 600
Wakefield, MA 01880, USA
Phone: +1 781-876-8955, **Fax:** +1 781-610-9864

HOW CAN WE HELP YOU?

Please complete the form below, choosing the subject and inquiry type in order to ensure that your question is directed to the appropriate subject matter expert.

EMAIL *

co224015@ucf.edu

ORGANIZATION

University of Central Florida

HOW CAN WE HELP YOU? *

General Inquiry

ADDITIONAL DETAILS

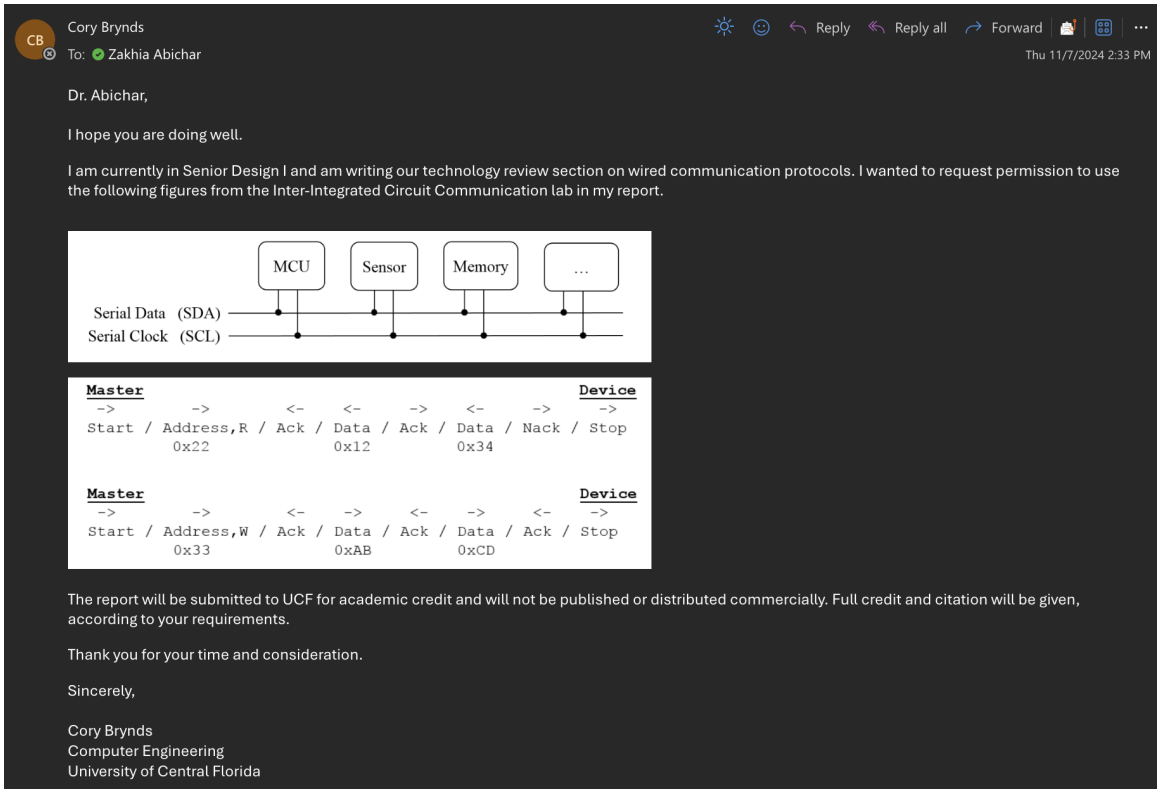
I wanted to request permission to use the Specifications Architecture figure from the NFC Specifications webpage in my senior design report, which I am completing as part of my undergraduate requirements at UCF.

The report will be submitted to my university for academic credit and will not be published or distributed commercially. Full credit will be given, according to your requirements.

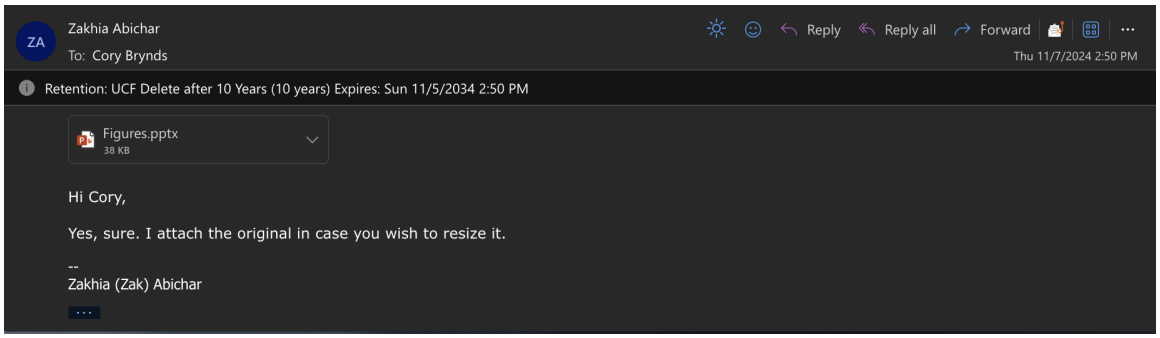
Please let me know if you need any additional information regarding my request.

Sincerely,
Cory Brynds

Request for use of image of NFC Specifications Architecture (Figure 4.1)



Request for use of images of I2C communication interface (Figures 4.3-4.5)



Approval for use of images of I2C communication interface (Figures 4.3-4.5)

APPENDIX C - LLM PROMPTS AND RESPONSES

1.0 Perplexity Prompts and Responses

1.1 Microcontroller Selection

Prompt: "Which WiFi-capable microcontroller should I select to interface with peripherals such as an NFC reader and TFT display for my senior design project?"

Response: "When selecting a WiFi-capable microcontroller to interface with peripherals like an NFC reader and a TFT display for a senior design project, consider the following options:

ESP32

- Features: The ESP32 is a popular choice for IoT projects due to its built-in WiFi and Bluetooth capabilities. It has dual-core processing power, which is beneficial for handling multiple tasks such as reading NFC data and updating a TFT display simultaneously[2][4].
- Compatibility: There is a wide range of libraries available for interfacing with NFC readers and TFT displays, making it easier to develop your project. However, some users have reported challenges with pin conflicts or library compatibility, which may require careful management of resources[2].
- Cost: The ESP32 is cost-effective and widely supported by a large community, providing plenty of resources and examples[5].

STM32F407

- Features: The STM32F407 offers more I/O pins and better analog-to-digital conversion compared to the ESP32. It is suitable for projects requiring extensive peripheral connections and high-performance processing[4].
- Performance: This microcontroller can handle more complex tasks and offers better documentation for low-level programming, which might be beneficial if your project demands precise control over peripherals[4].
- Considerations: While it does not have built-in WiFi, you can add this capability using external modules. This might be necessary if your project requires robust network connectivity.

RA6M3

- Features: The RA6M3 from Renesas includes a dedicated TFT controller and is designed for IoT applications. It supports high-speed USB and Ethernet, which could be advantageous if your project involves high data throughput[3].
- Ecosystem: Supported by the Flexible Software Package (FSP) and FreeRTOS, it offers a flexible development environment. This might be beneficial if your project requires advanced features like security or large RAM usage[3].

Recommendation

For most student projects focusing on IoT with moderate complexity, the ESP32 is likely the best choice due to its integrated WiFi, ease of use, and strong community support. If

your project demands more I/O flexibility or specific performance characteristics, consider the STM32F407 or RA6M3 depending on your specific needs."

1.2 Table Comparison of Microcontrollers

Prompt: "Using the same microcontrollers as before, generate a table comparing features such as the number of IOs, memory sizes, and operating voltage of each MCU"

Response:

Microcontroller	Number of I/O Pins	Flash Memory (MB)	SRAM (KB)	Operating Voltage (V)
ESP32	36[42]	4[43]	520[44]	2.2 - 3.6 V[45]
STM32F407VGT6	100[46]	1[46]	192[46]	2.0 - 3.6V[46]
RA6M3	-	2[47]	640[47]	2.7 - 3.6 V[48]

"I have generated the table comparing the features of the microcontrollers you are interested in. Please refer to the table for details on the number of I/Os, memory sizes, and operating voltage for each MCU."

1.3 Extended Table Comparison of Microcontrollers

Prompt: "Can you regenerate the table using three additional key features of each MCU? Also, compare the ICs of each microcontroller instead of a dev board, and cite the information from each respective datasheet."

Response:

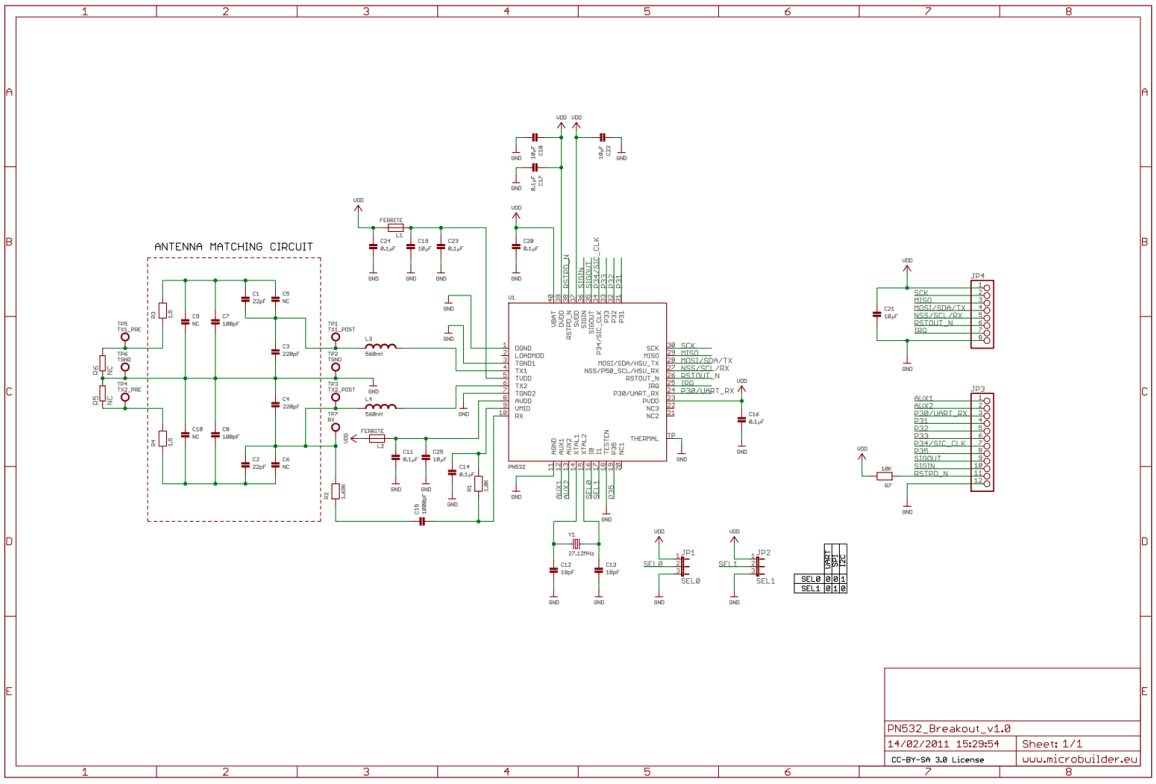
MCU	# of I/Os	Memory Sizes	Operating Voltage (V)	Processor Speed (MHz)	Power Consumption	Supported Comm. Protocols
ESP32	34[7]	Flash: 4096 KB, SRAM: 520 KB[8]	2.7V - 3.6V[7]	240 MHz[9]	Active: 160 mA; Sleep: varies[10]	UART, I2C, SPI, Bluetooth, Wi-Fi[11]

STM32 F407	100[13]	Flash: 1024 KB, SRAM: 192 KB[14]	1.8 V to 3.6 V[14]	168 MHz[13]	Active: 238 µA/MHz, Sleep: unknown[14]	UART, SPI, I2C, USB, CAN[13]
RA6M3	176[15]	Flash: 2048 KB, SRAM: 640 KB[15]	2.7V to 5.5V[15]	120 MHz[15]	Active: 66 mA, Sleep: 1.5 uA[15]	USB, Ethernet, CAN, UART, I2C[15]

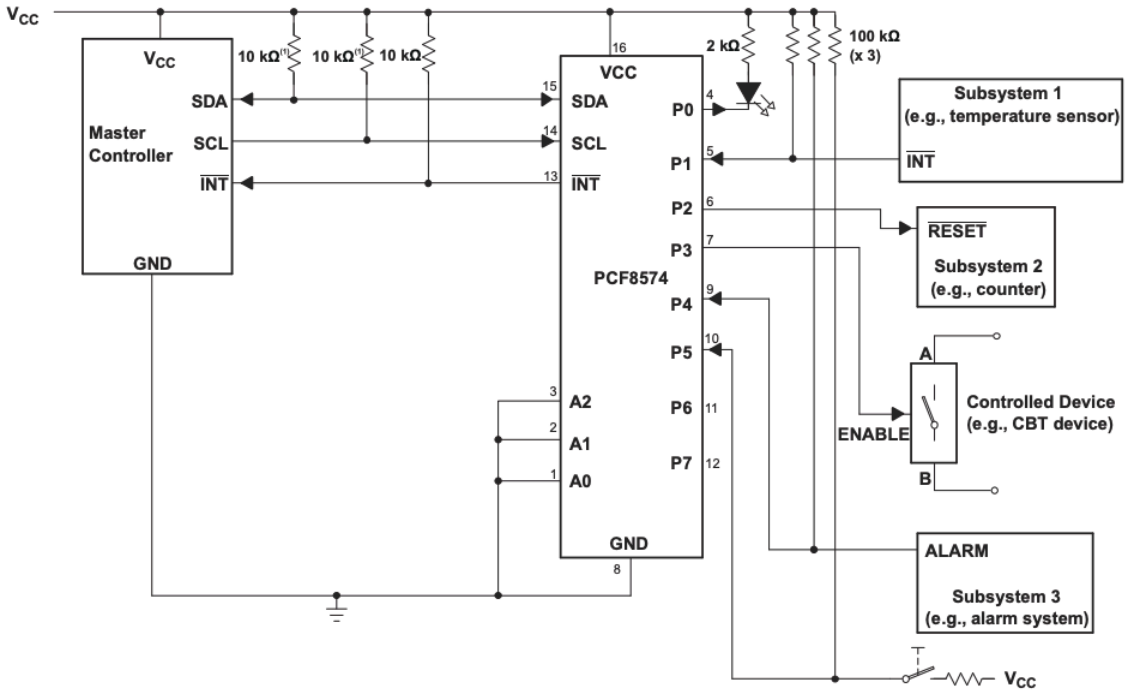
“I have generated the table comparing the features of the microcontrollers, focusing on their ICs. Please refer to the table for details on the number of I/Os, memory sizes, operating voltage, and additional key features for each MCU.”

APPENDIX D - REFERENCE SCHEMATICS

1.0 PN532 Breakout Schematic (Adafruit Industries)

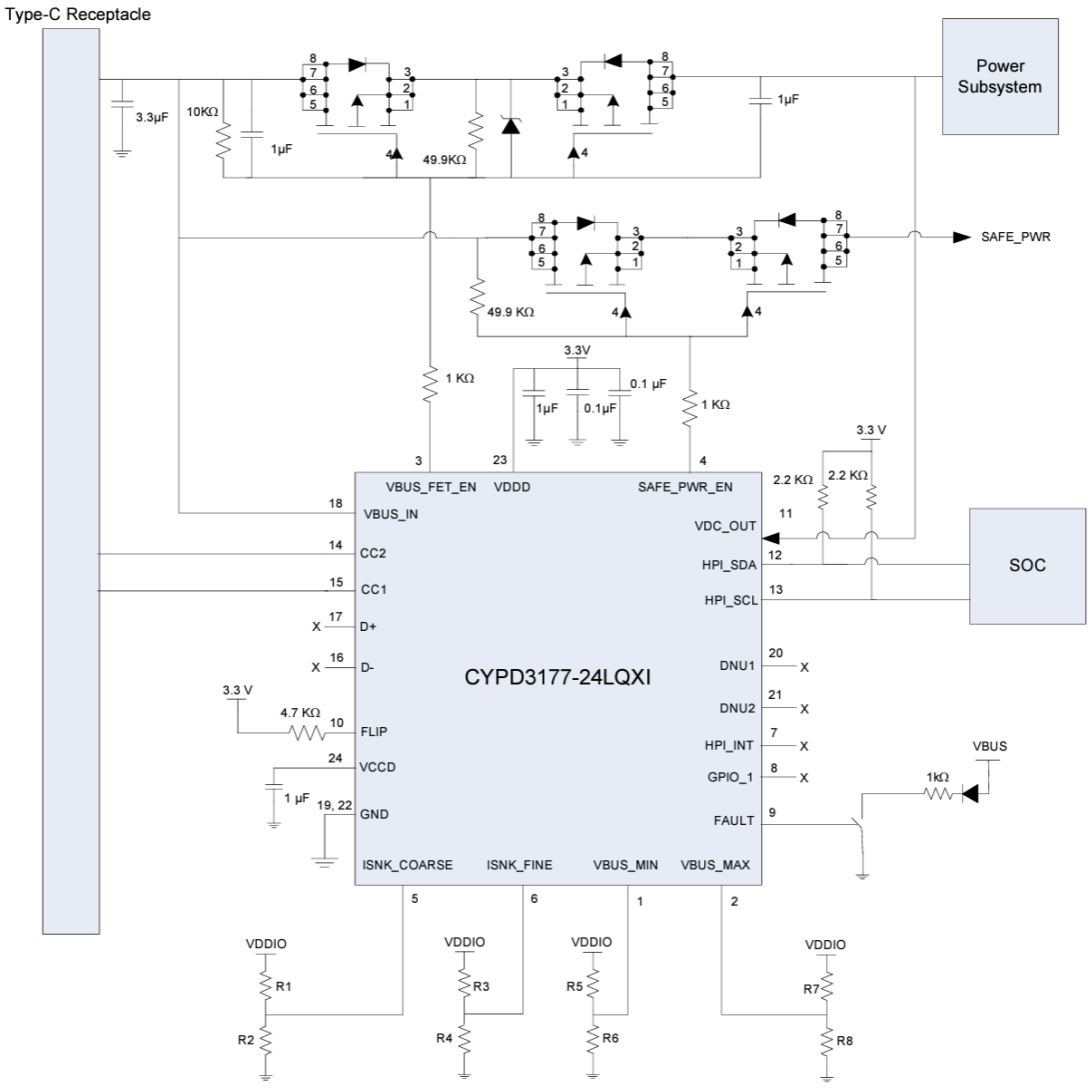


2.0 PCF8574 Application Circuit (From Texas Instruments Datasheet)



3.0 CYPD3177 Datasheet Reference Schematic [72]

Figure 3. EZ-PD BCR based Application Diagram (for Electronic Systems Requiring 12 V to 15 V Input at 2 A)



Notes
1. Refer to Table 2, Table 3, and Table 4 for values of these resistor divider networks.
2. FLIP pin is in LOW state when TypeC Plug is upside-up, and in Hi-Z state when upside-down.
3. Use a 50-kΩ resistor on the FLIP pin to set the UFP VDO data capability bit to 0.

4.0 BQ25798 Datasheet Reference Schematic [73]

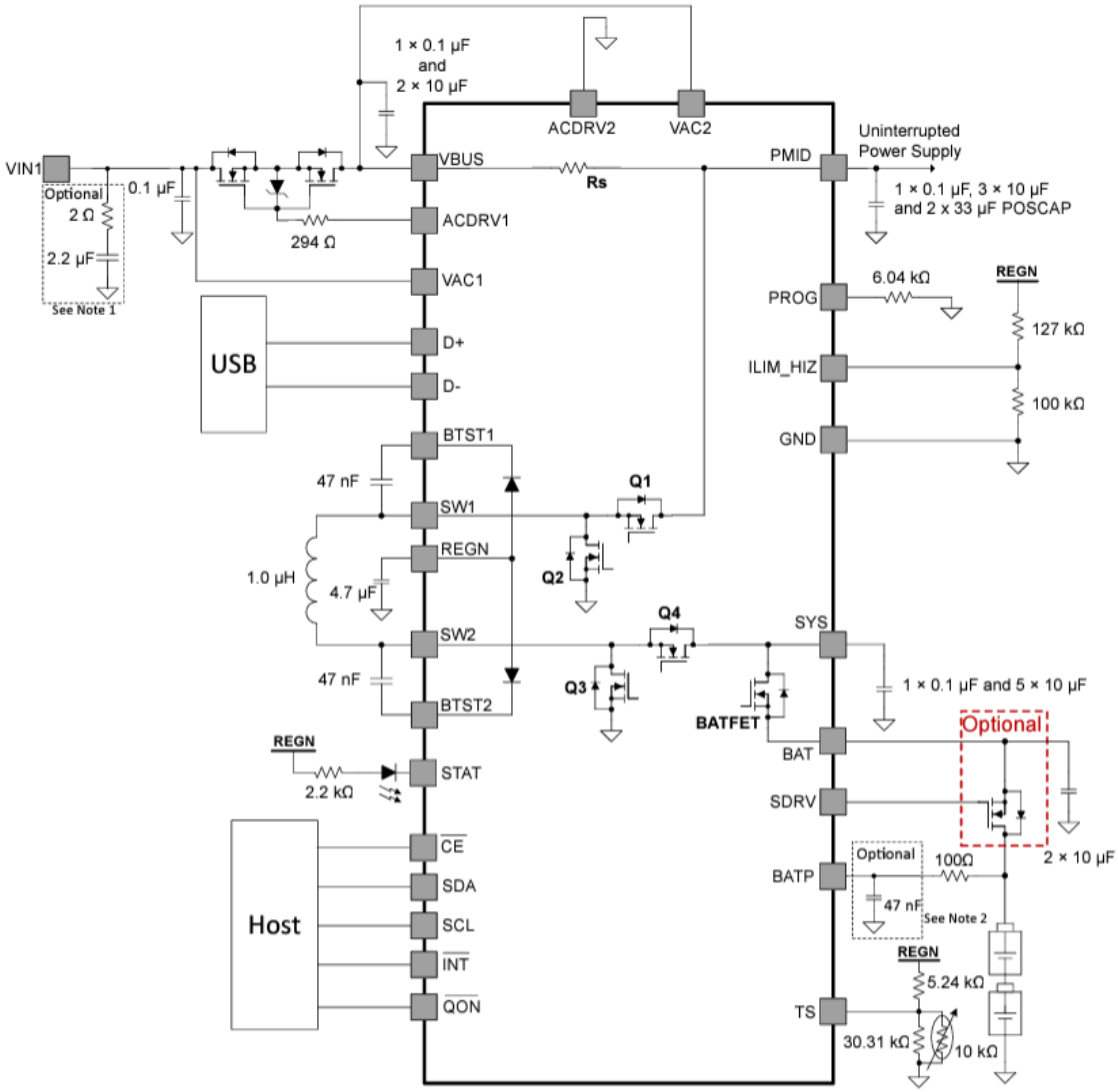


Figure 10-2. BQ25798 Application Diagram with Single Input Source, Back Up Mode and Ship FET

5.0 LM62460-Q1 Datasheet Reference Schematic [74]

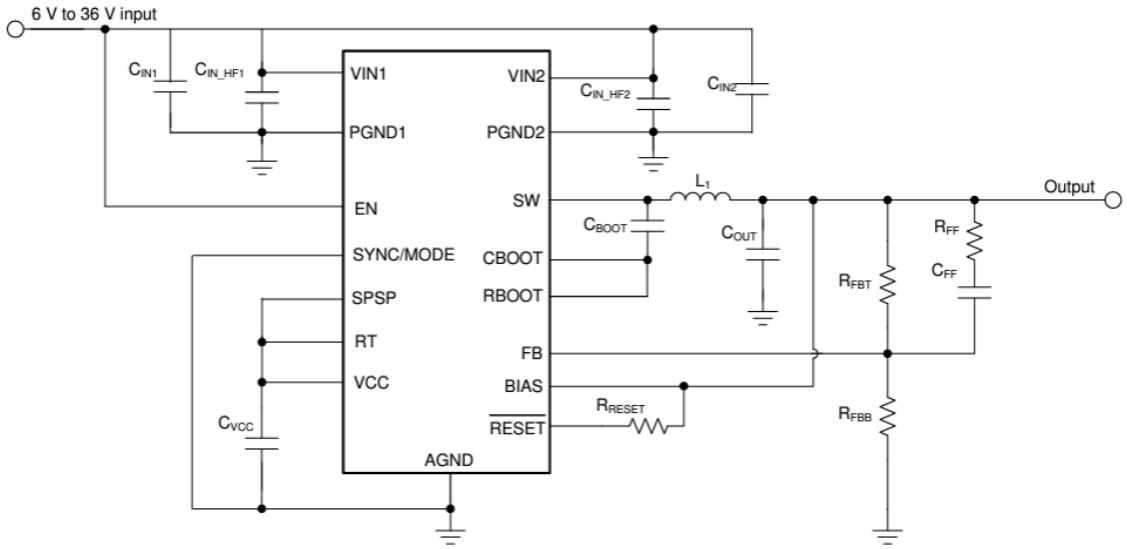
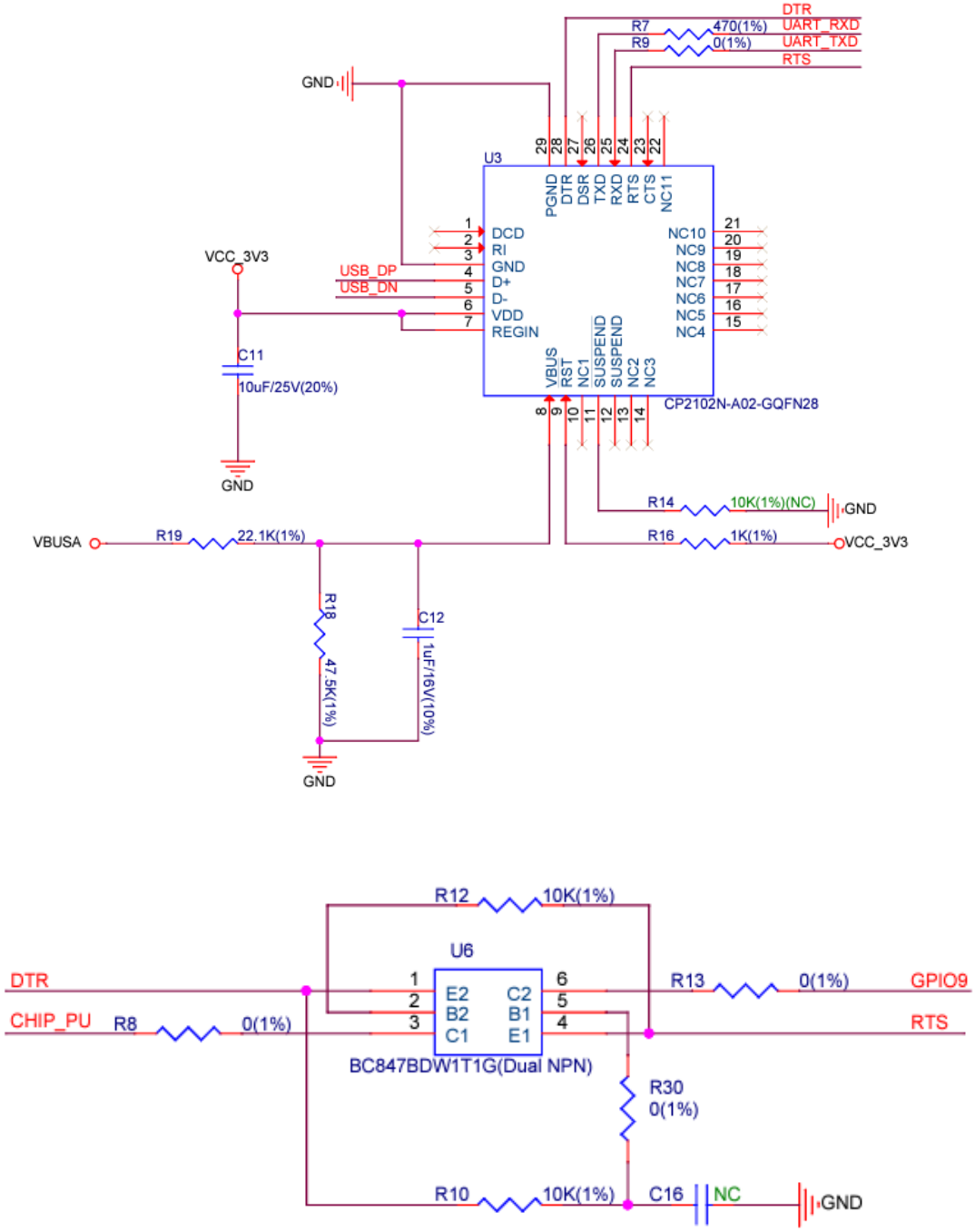


Figure 9-1. Example Application Circuit - 400-kHz Adjustable Output

UART:


DTR
UART_RXD
UART_TXD
RTS
R7
470(1%)
R9
0(1%)
GND
U3
PGND
DTR
DSR
TXD
RXD
RTS
CTS
NC11
DCD
RI
GND
D+
D-
VDD
REGIN
NC10
NC9
NC8
NC7
NC6
NC5
NC4
VBUS
RST
NC1
SUSPEND
SUSPEND
NC2
NC3
CP2102N-A02-GQFN28
VCC_3V3
USB_DP
USB_DN
C11
10uF/25V(20%)
GND
R14
10K(1%)(NC)
GND
VBUSA
R19
22.1K(1%)
R16
1K(1%)
VCC_3V3
R18
47.5K(1%)
C12
1uF/16V(10%)
GND
R12
10K(1%)
U6
DTR
CHIP_PU
R8
0(1%)
E2
B2
C1
C2
B1
E1
BC847BDW1T1G(Dual NPN)
R13
0(1%)
GPIO9
RTS
R30
0(1%)
R10
10K(1%)
C16
NC
GND