**MAKERERE**  **UNIVERSITY**



**COLLEGE OF ENGINEERING DESIGN ART AND TECHNOLOGY**

**SCHOOL OF ENGINEERING**

**DEPARTMENT OF MECHANICAL ENGINEERING**
**FINAL YEAR PROJECT REPORT**

# DESIGN AND CONSTRUCTION OF A 5-CODE KEY BOX FOR MULTIPLE OCCUPANCY OFFICES OF THE MECHANICAL ENGINEERING DEPARTMENT

**STUDENT NAME**: MUGERWA RICHARD JACKSON

**STUDENT NUMBER:** 2200706331

**REGISTRATION NUMBER**: 22/U/6331

**MAIN SUPERVISOR:** MR. EDMOND MPAGI

SIGNATURE: ……

DATE: 19/6/2026

**CORE SUPERVISOR:** Dr. NOBERT MUKASA

SIGNATURE: ……

DATE: ……...

A project report submitted to the Department of Mechanical Engineering at CEDAT in partial fulfilment of the requirements for the award of Bachelor of Science in Mechanical Engineering of Makerere University.

# DECLARATION

I, **Mugerwa Richard Jackson**, declare that the information presented in this report for my final year project entitled "***Design and Construction of a 5-code key box for multiple occupancy offices of the Mechanical Engineering Department***" is my original work and is in no way a duplication of any other document and has never been submitted or published before by another person or organisation.

Signature: .....

Date: 19/06/2026

## Digitisation and Self-Archiving Consent Agreement: Theses

**Agreement between Makerere University & Students (Authors of Theses / Dissertations / Reports)**

1. The author is a student of Makerere University and author of the thesis / dissertation entitled:

I am a student at Makerere University and the author of this dissertation with Reg No: 22/u/6331.

The title is DESIGN AND CONSTRUCTION OF A 5-CODE KEY BOX FOR MULTIPLE OFFICES OF THE MECHANICAL ENGINEERING DEPARTMENT.

2. The author grants to the University:
   a. The right to deposit the electronic version of the Thesis / Dissertation into Makerere University Institutional Repositories (Mak IR) or (Mak UD); and
   b. The right to store the thesis / dissertation in Mak IR / Mak UD and make it permanently available to the general public via the Internet at no cost to the general public after a grace period (if any is specified). Choose one of the two options below:
   c. The Author may opt for immediate open access to the public ……Allow access immediately……
   d. Or Restrict access indefinitely or for the specified number of years: ……

3. The author warrants that to the best of the authors knowledge and belief:
   a. The thesis / dissertation is an original work;
   b. The author is the owner of all the intellectual property in the thesis / dissertation;
      or
   c. The Author is entitled to deal with the intellectual property in the thesis / dissertation by publishing it on the Internet
   d. The Author has the right, power and authority to enter into this Agreement and to grant the University the rights contained in this Agreement; and
   e. The University's use of the thesis / dissertation pursuant to this Agreement will not infringe the intellectual property rights of any third party.

4. The Author acknowledges and agrees that the University is not responsible or liable for any breach of the intellectual property rights in the thesis / dissertation, in particular any breach of copyright, as a result of the use of the thesis / dissertation pursuant to this Agreement.

5. The University acknowledges that the rights granted by the Creator in clause 2 of this Agreement, do not cause any transfer or assignment of any proprietary rights in the intellectual property in the article to the University.

Signed by the Author as confirmation that the Author has read and accepted the terms of this Agreement:

Name: **MUGERWA RICHARD JACKSON**

College/School: CEDAT Department: MECHANICAL ENGINEERING

(Tick) Type of Degree: (Undergraduate / PGD / Masters / PhD), Reg. No.: _____

Tel No.: 0755866166 E-Mail: richard0755866166@gmaill.com

Signature: _____ Date: 20/06/2026

Supervisor's endorsement: _____ 22/06/2026

# ABSTRACT

The increased need for secure and efficient management of shared resources in institutional environments has prompted the development of automated systems such as electronic key boxes that eliminate the inefficiencies of manual key handling. In the Mechanical Engineering Department, shared offices and laboratories faced challenges of key loss, unauthorised access, and accountability, leading to reduced operational efficiency and security concerns. This project focused on the design and construction of a 5-code electronic key box that utilised a microcontroller-based control circuit integrated with a keypad, display unit, servo motor and solenoid lock to regulate access to the office key through personalised codes. The system enhanced safety by logging user activities, triggering alarms during unauthorised attempts, and allowing only verified users to retrieve or return the key. The study also emphasised cost-effectiveness, modularity, and future scalability to accommodate IoT integration for remote monitoring and data access. Through the implementation of this system, the project contributed to digital transformation in educational institutions, promoted responsible access management, and supported sustainable technological advancement aligned with the goals of innovation and institutional efficiency.

# ACKNOWLEDGEMENT

First and foremost, I give all praise and worship to Almighty God, whose guidance, strength, and grace have sustained me throughout this journey and brought my plans to fruition according to His will.

I extend my deepest gratitude to the Makerere University administration, and in particular, the Department of Mechanical Engineering, for establishing the Final Year Project program. This initiative has provided an invaluable platform for students like myself to experience the handling of a real-life engineering project, serving as a critical foundation for our future careers. Through this work, I have gained a profound appreciation for how diverse course units interconnect and complement each other to achieve a unified goal.

My special and sincere thanks go to my project supervisor, Mr. Edmond Mpagi, for his steadfast guidance, insightful feedback, and the considerable time he devoted to reviewing this report. May God bless him abundantly.

I am eternally grateful to my parents and siblings for their unwavering love, prayers, care, and sacrifices. Their financial and emotional support has been my cornerstone at every step, preparing me for the future.

I also express my thanks to MAK UNIPOD and Mr. Magembe Charles for the resources, support, and mentorship that have been instrumental in helping me shape and achieve my goals.

To my friends, I offer heartfelt thanks for your genuine camaraderie and encouragement throughout this four-year academic journey. Your motivation and reassurance during challenging moments kept me focused and resilient.

Finally, I acknowledge all those whom I have not mentioned individually but who have supported me, directly or indirectly, on this educational path. Thank you for walking with me to this point.

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

| Acronyms. | | Full Form |
|---|---|---|
| CEDAT | - | College of Engineering, Design, Art and Technology |
| CAGR | - | Compound Annual Growth Rate |
| USD | - | United States dollar |
| PIN | - | Personal Identification Number |
| RFID | - | Radio Frequency Identification |
| SDG | - | Sustainable Development Goals |
| LCD | - | Liquid-Crystal Display |
| GSM | - | Global System for Mobile Communications |
| AES | - | Advanced Encryption Standard |
| GCM | - | Galois/Counter Mode |
| CCM | - | Counter with Cipher Block Chaining Message Authentication |
| MCU | - | Machine Control Unit |
| BOM | - | Bill OF Materials |
| UPS | - | Uninterruptible Power Supply |
| I/O | - | Input and Output |
| USB | - | Universal Serial Bus |
| UI | - | User Interface |
| GPIO | - | General Purpose Input /Output |
| RSSI | - | Received Signal Strength Indicator |
| BLE | - | Bluetooth Low Energy |
| GRP | - | Glass fibre reinforced polyester |
| IEC | - | International Electrotechnical Commission |
| AT | - | Attention Command |

# CHAPTER 1: INTRODUCTION

## 1.1 Background

Office buildings are commonly rented or leased by companies, organisations, and institutions. Some organisations own the buildings but still have limited space due to their large company structures. Due to limited space, many of these organisations assign multiple employees to the same office, resulting in shared workspaces with a high number of occupants per room, which leads to multiple occupancy in offices. However, each office room typically has a maximum of three physical keys, which must be shared among all employees using that space. This arrangement creates several challenges. Many organisations struggle to duplicate keys or replace locks because these processes can be expensive, time-consuming, and often require external service providers. In some workplaces, the available keys are kept at the front desk, reception, or security office. While convenient, this practice exposes the organisation to significant risks such as key misplacement, unauthorised access, or misuse by individuals with ill intent. Such vulnerabilities can lead to serious consequences, including the loss of organisational property, unauthorised entry, exposure of confidential documents, and potential leakage of sensitive information, all of which may result in substantial financial losses and reputational damage. Additionally, when keys are assigned to specific employees within a shared office, other risks arise. Employees may lose the key, forget it at home, misplace it, or be absent unexpectedly due to emergencies or personal commitments. This can disrupt access to the office, delay work operations, and compromise overall security and efficiency.

Security in both physical and digital infrastructures across the globe remains a foundational concern. As organisations increasingly prioritise data protection and cyber defence, physical security, particularly in key management and access control, continues to play a critical role in safeguarding valuable equipment, intellectual property, and sensitive facilities. Globally, the physical security market was valued at USD 112.5 billion in 2022 and is expected to reach USD 158.5 billion by 2027, growing at a compound annual growth rate (CAGR) of 7.1% according to Markets and Markets, 2023 (Physical Security Market by Systems, 2025) This steady growth highlights the urgency and relevance of securing

physical assets, especially in environments where multiple users share common access points. Educational institutions, particularly universities, are complex environments where secure access control is essential yet often overlooked. Many academic departments, especially those with practical and research-intensive components like mechanical engineering, must manage shared access to laboratories, equipment rooms, and academic offices. These spaces typically house expensive machines, tools, prototypes, and sometimes hazardous materials. A 2021 report by (Video Surveillance, 2021) found out that over 60% of security professionals rank access control, including physical key management, as a top concern due to risks related to unauthorised access, asset theft, and loss of institutional property.

At Makerere University, such concerns are increasingly real. The Department of Mechanical Engineering, within the College of Engineering, Design, Art and Technology (CEDAT), hosts several staff, students, and researchers who require access to shared offices, machine labs, and research facilities. With a growing population of users and physical infrastructure, the department often relies on traditional and manual key management systems, such as personally keeping the keys or leaving them at the custodian's office or at the front desk.

Local reports and internal departmental reviews have noted recurrent issues, including missing keys, untracked borrowing, and unauthorised office access, especially during off-peak hours or semester breaks (Auditor General Report, 2024). In some instances, misplaced keys have delayed access to vital resources, causing interruptions in teaching schedules and research activities. These issues are not isolated. These challenges are not unique. A study by Okello and Nabwire (Lawal-solarin et al., 2021) On operational security in Ugandan universities reported that 64% of engineering departments surveyed had experienced security lapses directly related to poor key management practices.

As digital access control systems remain expensive and technologically intensive for many departments to implement, there is a clear need for a cost-effective, secure, and user-friendly solution for managing physical keys. For Makerere University's Mechanical Engineering Department, the development of a tamper-resistant, multi-user physical key

box offers a practical opportunity to enhance departmental efficiency, reduce risks, and improve accountability in access control.

Despite all, this project focuses on the design and construction of a physical key box tailored to the unique operational needs of Makerere University's Mechanical Engineering Department. The solution aims to blend mechanical reliability with controlled accessibility, ensuring that shared office and laboratory spaces remain secure while also being conveniently accessible to authorised users.

## 1.2 Problem Statement

At Makerere University, the Department of Mechanical Engineering has a growing number of staff, researchers, graduate students, and technicians sharing a limited number of workspaces, resulting in multiple occupancy in offices. Since each workspace lock has fewer keys than occupants in the office, access can be disrupted when the keys are lost, misplaced, or stolen, leading to delays in work, missed classes, postponed activities, and reduced productivity. If the keys fall into unauthorised hands, valuable departmental and personal property such as laptops, printers, laboratory equipment, tools, and sensitive documents may be stolen or compromised, resulting in security risks and unplanned expenses. These challenges are further exacerbated in other multiple occupancy environments, where several individuals depend on the same set of keys. Without a structured and secure system for managing access, it becomes difficult to identify who took which key, at what time, and whether it was returned, leading to confusion, inefficiencies, and strained administrative oversight. While advanced digital locking systems or biometric access tools offer possible solutions, they remain financially impractical for most university departments operating under budgetary constraints.

## 1.3 Objectives

### 1.3.1 General Objective

To develop a secure, multi-user electronic key box system that utilises a 5-code authentication mechanism to efficiently manage, monitor, and control access to keys of shared offices and other areas in the Mechanical Engineering Department.

### 1.3.2 Specific Objectives

1. To determine the design requirements of the lock system.
2. To develop concept sketches, the flow algorithm, and 3D models of preliminary and final designs.
3. To construct and carry out a performance evaluation of the prototype, fully assembled and functional.

## 1.4 Significance

The project has a significant contribution to SDG 8 Decent Work and Economic Growth and SDG 9 Industry, Innovation, and Infrastructure by promoting technological advancements, operational efficiency, and sustainable development within educational and institutional environments. Automating key management has reduced human errors, creating safer and more efficient working conditions, which is in line with SDG 8's goal. In relation to SDG 9, the project encouraged innovation and modernisation of institutional infrastructure through the adoption of smart access control technologies. It demonstrated how locally engineered solutions could be developed to improve security management and resource utilisation while reducing reliance on costly imported systems.

## 1.5 Justification

This project has addressed the objectives by providing an alternative way of storing a single key in an electronic storage box that provides access to authorised office space users through a 5-code PIN or RFID tag/ card to remove and press back the key in the storage box, high number of individuals in an office can now access the workspace at any time of the day without concern for theft and property loss since individual details of key access were logged and stored in the system memory. In any case, the access time and office used can

be retrieved. This, therefore, has reduced the challenges that come with multiple occupancy in offices.

## 1.6 Scope of the Work

The primary goal of this project was to focus on the design, construction, and testing of an electronic key box system intended for managing multiple occupancy in offices within the Mechanical Engineering Department. My design covered a 5-code PIN and an RFID tag/ card used to access the system for the key. The door of the box opened and closed automatically once a valid 5-code PIN was used. The system code was programmed using C++ programming language using the Arduino IDE, used to compile and upload the code to the Arduino Mega, which was the MCU of the system. The MCU has 20 usable GPIO pins in total, where 14 are digital I/O, and 6 are analog inputs pins, used to communicate with the different electronic and mechanical components. The system has a power backup which lasts for a maximum of five days of continuous use. It operated in a distance radius of 5m to determine whether the key is in the box or the return of the key in the box, and also reminded the individual to return the key to the box if the set limit distance was exceeded through a long-sounding alarm made by the buzzer. There was 3D modelling of some of the internal components and exterior design in SolidWorks. The performance evaluation was carried out, and the data were compared to the available system on the market in the same operational range. The system does not utilise or integrate biometric security means due to the expense and time frame of the project.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Introduction

Key storage systems have evolved from simple mechanical locks and cabinets to electronic and IoT-enabled smart key boxes. In difficult or high-occupancy environments, such as universities, shared laboratories, and office complexes, managing access to shared keys presents significant challenges. Traditional systems, Conventional mechanical key racks and paper sign-out sheets lack accountability, are vulnerable to unauthorized duplication, and do not provide real-time alerts or audit trails, and do not provide real-time monitoring (Electronic key and asset management, n.d.). Electronic key-box systems, on the other hand, integrate microcontrollers, user authentication, actuators, and logging mechanisms to control and monitor key borrowing and returning. Recent advancements extend these designs with IoT connectivity, enabling real-time notifications, remote audits, and cloud-based databases (David Odu et al., 2017). Electronic key-box systems combine controlled release mechanisms, authenticated access (PIN/RFID/biometrics), logging, and remote alerts to address these gaps.

In environments with high personnel turnover (like military bases, shared offices, and laboratories), early solutions relied on sign-in/out logs or mechanical key cabinets. The lack of digital records created accountability issues. With the creation of affordable microcontrollers in the 2000s, research shifted to keypad and password-protected systems, which allowed dynamic reconfiguration of access codes (Rana et al., 2023) More recently, IoT-based smart boxes, such as the Thai laboratory key system (Nonthaputha et al., 2021), have integrated barcode scanning, touchscreen panels, and mobile alerts, improving usability and accountability in multi-user contexts. Mechanical locks and keys were the dominant solution for centuries. The late 20th century saw the adoption of PIN keypads, magnetic stripe cards, and RFID tokens for access control. These technologies brought re-programmability and user accountability. By the 2000s and 2010s, cheap microcontrollers enabled compact, reprogrammable electronic locks that could implement PIN entry, local logging, serial/GSM alerts, and simple remote control (David Odu et al., 2017).

Modern commercial key cabinets provide robust management features, role-based access, centralised audits, per-user assignment of keys, reporting, and physical tamper detection, making them the enterprise benchmark for office key management. These systems illustrate the expectations for an office 4/5-code solution when scaled: auditability, secure user provisioning, and integration with facility management (Smart Key Storage Systems, 2025). Recent advances include secure cloud logging (IoT), smartphone-based authentication, biometric augmentation (fingerprint/face), and encrypted wireless protocols. These features improve auditability and flexibility but introduce new attack surfaces (software, network) that require strong device hardening and cryptography. Best practice recommends authenticated encryption (e.g., AES-GCM/CCM) and secure firmware update paths for fielded devices (IoT SIM Card, 2025). IoT, cloud, and encrypted radio protocols introduced remote monitoring, SMS/alerting, and centralised audit storage in the 2010s–2020s. Industry solutions (electronic key cabinets) now offer role-based access, audit trails and integration with workforce systems.

## 2.2 Some of the previously used modelling equations and Formulae to create systems similar to the key storage box model

Table 1: Modelling equations and Formulae

| **Model / Formula** | **Description** | **Relevance to Key-Box with PIN Codes** |
|---|---|---|
| **Key space/ Number of possible combinations.** | If you have a PIN system with length $n$ digits and $b$ possible symbols per digit (e.g. 0 - 9, so $b$ = 10), then total possible PIN combinations = ( $b^n$ ). | For 5-code: (10^5 = 100,000). This is fundamental for estimating brute-force risk(Colby Gutierrez-Kraybill, 2016). |
| **Probability of guessing the PIN in one attempt** | ($P = \frac{1}{b^n}$) | If the attacker picks one random 5-digit code (with b=10, n=5), ($P = \frac{1}{10^5}$ = 0.00001). Useful for |

| | | |
|---|---|---|
| | | designing lockout thresholds(Colby Gutierrez-Kraybill, 2016). |
| **Probability of failing vs success over multiple attempts** | If you allow *t* attempts, the probability of *at least one success* = (1 - (1 - P) $^{t}$) assuming independent guesses. | Helps to define how many wrong attempts before lockout; risk assessment for letting 3, 5, etc. tries. |
| **Entropy estimation of PIN/password** | Entropy (=n\cdot \log2(b)) bits | For 5-digit numeric PIN: entropy = (5\cdot \log_2(10) ≈ 5 × 3.3219 = 16.61) bits. Indicates strength roughly. |
| **Security/usability trade-off in lockout systems** | Some papers (in general PIN/password literature) model the trade-off between the number of allowed wrong tries (k) vs inconvenience vs security risk; sometimes via probability models of attacker success and expected false lockouts. | Useful for your system's "3 wrong try lockout" decision. Although the prototype key box papers did not fully formalise this, related works . |

All the above together provide hardware, firmware, user management, and communication aspects, the four pillars necessary to design a storage key-box. However, they fall short in the implementation of independent key storage.

## 2.3 Electronic Actuation and Locking Mechanisms

The transition from mechanical to electronic key storage systems fundamentally depends on reliable actuation mechanisms that translate authentication decisions into physical actions. Contemporary smart key boxes employ several categories of electronic locking systems, each presenting distinct advantages and engineering trade-offs.

### 2.3.1 Solenoid-Based Actuation Systems

Solenoids represent the most widely adopted actuation technology in electronic key storage systems due to their simplicity, rapid response, and low cost (David Odu et al., 2017). A typical solenoid actuator consists of a ferromagnetic core (plunger) surrounded by an electromagnetic coil; when energised, the coil generates a magnetic field that displaces the plunger, thereby releasing a locking mechanism (Rana et al., 2023). The power requirements for solenoid actuation present a significant engineering consideration for battery-operated key boxes, with typical door open/close operations requiring a 5.0V supply, as documented in finger vein authentication key box implementations (Nonthaputha et al., 2021). However, inrush currents during solenoid activation can substantially exceed steady-state consumption, necessitating careful power supply design and capacitor buffering (Electronic key and asset management, n.d.).

The mechanical latching configuration significantly influences the security characteristics of solenoid-based systems. Fail-secure designs, where the lock remains engaged during power loss, are generally preferred for key storage applications where preventing unauthorised access during power outages outweighs accessibility considerations. This contrasts with fail-safe configurations typically employed on emergency egress doors, which unlock during power failure to facilitate evacuation (Avigilon, 2023). For key storage boxes intended to protect valuable assets, fail-secure actuation represents the appropriate design choice.

### 2.3.2 Motor-Driven Mechanisms

Alternative actuation approaches employ electric motors rather than solenoids, offering distinct advantages for applications requiring controlled, gradual movement rather than impulsive actuation. The key holding element, housing, power assembly, and sensing units work cooperatively, with the power assembly configured to drive the key holding element to leave the housing or enter the housing along an access direction (Smart Key Storage Systems, 2025). When the key holding element moves, it is configured to trigger sensing units, providing position feedback essential for state determination (Rana et al., 2023).

Motor-driven mechanisms typically consume less peak power than solenoids but require extended actuation periods, creating different power management challenges. The sensing units disposed in the housing enable precise control of the motor's operation, allowing the control module to stop the motor upon detection of fully open or fully closed states (Nonthaputha et al., 2021). This feedback capability distinguishes motorised systems from simple solenoid actuators, enabling more sophisticated user interfaces and reducing mechanical stress on components (Electronic key and asset management, n.d.).

### 2.3.3 Electromagnetic Locking Systems

Electromagnetic locks present an alternative locking paradigm that eliminates moving mechanical parts entirely. These systems operate on the principle of electromagnetism, where an electromagnet mounted on the enclosure frame attracts an armature plate attached to the door when energised (Avigilon, 2023). Holding forces typically range from 270 kg to 680 kg for commercial applications, providing substantial resistance to forced entry (Smart Key Storage Systems, 2025).

The fundamental distinction between fail-safe and fail-secure configurations applies equally to electromagnetic locks. A fail-safe electromagnetic lock requires continuous power to remain locked; upon power loss, the magnetic field collapses and the door unlocks (Avigilon, 2023). Conversely, a fail-secure electromagnetic lock requires power to unlock, remaining locked during power interruptions (Avigilon, 2023). For key storage applications, fail-secure electromagnetic locks are preferred, though their continuous power consumption during locked states presents greater energy demands than solenoid-based mechanisms that only consume power during actuation(Electronic key and asset management, n.d.).

### 2.3.4 Position Sensing and State Verification

Reliable determination of lock state (locked/unlocked, open/closed, key present/absent) requires redundant sensing approaches to prevent state confusion that could compromise security or usability. Magnetic reed switches represent the most common position-sensing technology in electronic lock systems, offering contactless operation, high reliability, and minimal power consumption (Rana et al., 2023). When a magnet mounted on the

moving component approaches the reed switch, the switch contacts close, signaling position confirmation to the control microcontroller (Nonthaputha et al., 2021).

Optical position sensing provides an alternative approach with enhanced tamper resistance. A light emitter and photodetector pair, positioned such that the moving component interrupts or permits light transmission, offers non-contact sensing without the mechanical wear concerns of physical switches (David Odu et al., 2017). Recent implementations combine multiple sensing technologies to achieve redundant verification; for instance, hall effect sensors provide magnetic field detection while microswitches offer mechanical contact confirmation (Rana et al., 2023). This multi-modal approach prevents single-point sensing failures from compromising the system state

### 2.3.5 Tamper-Evident Locking Features

Recent developments in lock security have introduced tamper-evident mechanisms that detect and report physical interference attempts. A lock apparatus incorporating tamper detection includes a lock body, lock backplate, lock mechanism, and a sensor that raises an event when contact between sensor ends and lock body surfaces is disrupted (EP4438837, 2024). Such systems report lock tampering events to central monitoring infrastructure, and notably, if the safe loses power and subsequently restores, it reports any tampering event that occurred during the power interruption (EP4438837, 2024). This capability is particularly valuable for key storage applications where attackers might attempt power cycling to disable alarm systems.

## 2.4 Power Supply Architectures

Power management represents a fundamental engineering challenge for electronic key storage systems, particularly those deployed in locations where continuous mains power cannot be guaranteed. Contemporary systems employ hierarchical power architectures that balance reliability, cost, and operational longevity.

### 2.4.1 Primary Power Sources

Mains-powered systems offer unlimited operational capacity but require professional installation and remain vulnerable to grid outages. Typical commercial electronic key

cabinets operate from 5V, 12V DC or 24V DC power supplies derived from mains transformers or switched-mode power supplies (Electronic key and asset management, n.d.). The inclusion of filtering capacitors and voltage regulation circuitry ensures stable operation despite mains fluctuations; such filtering smooths supply variations and provides ride-through capability during brief interruptions (Smart Key Storage Systems, 2025).

Battery-primary systems eliminate installation complexity and enable deployment in locations without power infrastructure. Alkaline battery configurations using four to eight AA cells provide 6V to 12V operation suitable for low-power microcontrollers and solenoid actuation (Nonthaputha et al., 2021). Lithium-ion rechargeable batteries offer higher energy density and rechargeability, with typical capacities of 3500 mAh supporting approximately 50 to 100 key transactions on a full charge (Rana et al., 2023). However, battery-primary systems require regular monitoring and replacement cycles to prevent operational interruptions from battery depletion.

### 2.4.2 Battery Backup Architectures

Hybrid power architectures combine mains power with battery backup to provide resilience against power interruptions while enabling rich functionality during normal operation. When mains power is present, the system operates from the primary supply while maintaining battery charge through intelligent charging circuitry (David Odu et al., 2017). Upon power loss, the system transparently switches to battery backup, with Schottky diode isolation preventing back-feeding into the disabled mains supply (Electronic key and asset management, n.d.).

### 2.4.3 Power Consumption Optimization

Low-power design techniques significantly extend battery life in portable and backup-powered deployments. Microcontroller selection critically influences system autonomy; modern ARM Cortex-M series processors offer deep-sleep modes drawing less than 1 µA while retaining random access memory contents and real-time clock functionality (Nonthaputha et al., 2021). Periodic wake-ups for housekeeping functions such as battery

voltage monitoring and real-time clock maintenance typically occur at 1-second to 1-minute intervals, consuming negligible average current (Nonthaputha et al., 2021).

### 2.4.4 Power Supply Security Considerations

Power supply manipulation represents a potential attack vector against electronic key storage systems. Attackers may attempt power cycling to reset system state, bypass lockout timers, or force fallback to default configurations (Allen et al., 2024). Research on smart lock security has demonstrated that power interruption attacks can defeat certain authentication mechanisms, though robust implementations employ non-volatile memory to preserve lockout state across power cycles (Allen et al., 2024). Capacitive energy storage provides ride-through capability during brief power interruptions, preventing state loss from transient disturbances. The capacitor serving to supply power to circuitry during pulse width modulation on the DC power supply, coincidentally provides resilience against power cycling attacks (Smart Key Storage Systems, 2025).

## 2.5 External Enclosures and Physical Security

The enclosure housing electronic key storage systems must satisfy competing requirements for physical security, environmental protection, user accessibility, and aesthetic integration. Material selection and mechanical design directly influence both security level and deployment flexibility.

### 2.5.1 Polymeric Enclosures

Glass fibre reinforced polyester (GRP) enclosures offer corrosion resistance, electrical insulation, and cost-effectiveness for indoor and protected outdoor deployments. Industrial GRP enclosures such as the Thalassa PLM series are manufactured as single-molded pieces, eliminating seams that could admit water or provide attack points. These enclosures incorporate built-in canopies that shield user interface components from direct rainfall and provide thermal protection against solar loading (Schneider Electric, 2025).

The material properties of fibre-reinforced polyester provide significant practical advantages for key storage applications. The material is self-extinguishing and halogen-free, producing no toxic fumes in the event of fire. Anti-corrosion properties ensure longevity in coastal or industrial environments where metallic enclosures would degrade

(Schneider Electric, 2025). For key storage systems deployed in chemical storage areas or marine environments, GRP enclosures often outlast metallic alternatives by substantial margins.

Ingress protection ratings for GRP enclosures typically reach IP66, indicating complete protection against dust ingress and protection against powerful water jets. Impact resistance rated IK10 according to EN 62262 confirms survival of 20 joule impacts, equivalent to a 5 kg mass dropped from 400 mm (Schneider Electric, 2025). These ratings qualify GRP enclosures for outdoor installation in exposed locations, including building exteriors and industrial yards.

### 2.5.2 Metallic Enclosures

Stainless steel enclosures provide superior mechanical security and electromagnetic shielding for high-value installations. Type 304 stainless steel construction offers an optimal balance of corrosion resistance, strength, and workability. Industrial-grade stainless enclosures typically achieve IP66 ingress protection and IK10 impact resistance, with material thickness of 1.2 mm to 2.0 mm depending on size and security requirements (Schneider Electric, 2025).

The 120-degree door-opening angle of commercial stainless-steel enclosures facilitates user access to stored keys while maintaining structural integrity under load. Double-bar door lock systems with 3 mm throw bolts provide substantial resistance to prying attacks, distributing retention forces across multiple engagement points. Surface-mounted brackets enable secure wall attachment, preventing enclosure removal without tool access to mounting fasteners concealed within the locked volume (Schneider Electric, 2025).

### 2.5.3 Tamper Resistance and Mechanical Attack Mitigation

Physical security features protect against forced entry and covert tampering attempts. Enclosures intended for high-security key storage typically incorporate multiple tamper-evident and tamper-resistant design elements. The use of stainless-steel lid screws (material 1.4567) with tamper-resistant drives prevents unauthorized disassembly using commonly available tools. Casting sleeves made of stainless steel A2 provide corrosion-resistant threaded inserts that resist pull-out attacks (Schneider Electric, 2025).

The integration of tamper detection sensors with enclosure mechanical design creates comprehensive attack detection. Recent patent developments describe lock tampering detection where a sensor raises an event when a sensor end loses contact with a lock body surface, reporting the event through the system's communication infrastructure (EP4438837, 2024) For key storage systems, such tamper detection can trigger immediate alerts to security personnel while logging the event in permanent audit records.

### 2.5.4 User Interface Integration

Enclosure design must accommodate user interface components while maintaining environmental sealing. Recesses for membrane keypad installation provide factory-integrated provisions for user input devices, with sealing achieved through the keypad's own gasket or enclosure-mounted seals. Flat lid variants accommodate behind-panel mounting of display and input components visible through transparent windows. Clip-on design covers made of aluminum finished in matt silver, cover the installation level, providing both aesthetic finishing and mechanical protection for interface components (Schneider Electric, 2025).

Captive lid screws with retained washers prevent screw loss during maintenance while ensuring that environmental sealing remains intact even after repeated access. Mounting domes with M5 thread on the enclosure bottom and in the lid support internal component installation without compromising external sealing, providing dedicated attachment points for mounting plates and electronic assemblies (Schneider Electric, 2025).

The 100-degree to 120-degree lid opening angle facilitates user access to stored keys while maintaining structural integrity and weather sealing (Schneider Electric, 2025). Door retention mechanisms, such as friction stays or gas struts, hold the door open during user interactions, preventing accidental closure that could damage hinge mechanisms or pinch user fingers (Smart Key Storage Systems, 2025).

## 2.6 Communication and IoT Integration

Contemporary key storage systems increasingly incorporate wireless communication capabilities to enable remote monitoring, real-time alerting, and centralised management. This connectivity introduces both operational benefits and additional security

### 2.6.1 Wireless Communication Technologies

Wi-Fi connectivity enables direct integration with building networks and cloud services. Typical implementations use 2.4 GHz 802.11 b/g/n protocols, providing sufficient bandwidth for transaction logging, real-time status updates, and firmware updates (Rana et al., 2023). Power consumption during Wi-Fi operation is substantial, with active transmission drawing 100-300 mA at 3.3V, though duty cycling and power-saving modes reduce average consumption for infrequent communication (Nonthaputha et al., 2021).

Bluetooth Low Energy (BLE) provides lower power consumption than Wi-Fi while enabling smartphone interaction for authentication and management. BLE's 10 100 metre range is suitable for building-level deployments, and its widespread smartphone support enables user authentication via mobile credentials (Allen et al., 2024). However, research has demonstrated that Bluetooth-based smart locks may be vulnerable to replay attacks and relay attacks if cryptographic protections are insufficient (Allen et al., 2024). For key storage applications, BLE should be implemented with secure pairing and encrypted communication channels.

### 2.6.2 Security Considerations for Connected Systems

The integration of wireless communication introduces additional attack surfaces that must be addressed through a robust security architecture. Evaluation of smart security devices has revealed that many commercial products remain vulnerable to wireless attacks, including replay attacks, brute-force authentication bypass, and radio frequency jamming. These vulnerabilities affect both well-known and lesser-known brands, suggesting systematic weaknesses in IoT security implementation (Allen et al., 2024).

Firmware update mechanisms represent a critical security control for deployed systems. Secure over-the-air (OTA) updates require cryptographic signing of firmware images, with devices verifying signatures before installation (IoT SIM Card, 2025). Rollback protection

prevents attackers from forcing reversion to vulnerable firmware versions, while update authentication ensures only authorized parties can initiate updates (Rana et al., 2023).

### 2.6.3 Relay Attacks and Proximity Verification

Proximity-based authentication systems face the threat of relay attacks, where an attacker extends the effective range of a legitimate credential using signal amplification and retransmission. Research on passive keyless entry systems has demonstrated that relay attacks can successfully unlock doors from distances exceeding 100 metres using off-the-shelf software-defined radio equipment (Park & Hong, 2023). For key storage systems using smartphone or RFID authentication, similar attack vectors exist.

Countermeasures against relay attacks include distance bounding protocols and physical layer security techniques. The BackProx system demonstrates relay-resilient proximity detection using pseudo-random frequency hopping with reference backscattering devices, achieving 98% true positive detection at close range with 0.3% false positives at long range (Park & Hong, 2023). While developed for automotive applications, similar principles could apply to key storage systems requiring proximity verification.

### 2.6.4 Audit Logging and Cloud Integration

Centralized audit logging represents a primary benefit of connected key storage systems. Each key access event—user identification, timestamp, key selected, and outcome—transmits to central servers for permanent storage and analysis (Electronic key and asset management, n.d.). Cloud-based logging provides geographic redundancy and accessibility from any network-connected device (Nonthaputha et al., 2021).

Real-time alerting enables immediate response to security events. Configured alerts may trigger on failed authentication attempts, tamper detection, unauthorized access times, or key retention beyond scheduled return periods. Alert delivery via SMS, email, or push notification ensures that responsible personnel receive time-sensitive information regardless of location (Rana et al., 2023).

## 2.7 Bluetooth Low Energy (BLE) Communication for Proximity Detection

### 2.7.1 Overview of Bluetooth Low Energy Technology

Bluetooth Low Energy (BLE), introduced as part of the Bluetooth 4.0 specification by the Bluetooth Special Interest Group (SIG) in 2010, is a wireless personal area network technology designed for low-power applications (Bluetooth SIG, 2010). Unlike classic Bluetooth, BLE remains in sleep mode for most of its operational life, waking briefly to transmit data packets, making it particularly suitable for battery-powered devices such as beacons, sensors, and tracking systems (Gomez et al., 2012). This characteristic aligns with the power management requirements of the key storage system, where prolonged battery backup operation is essential.

### 2.7.2 The HM-10 BLE Module

The HM-10 is a popular, low-cost BLE module based on the Texas Instruments CC2540 or CC2541 system-on-chip (Jin, 2016). Operating in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band, the module supports both master (central) and slave (peripheral) roles, enabling flexible configuration for various applications. Key features relevant to this project include:

- **AT command interface** for configuration and control (Jin, 2016)
- **Remote control mode (MODE2)** allowing direct pin control via BLE
- **Received Signal Strength Indicator (RSSI) reporting** for signal strength monitoring
- **Advertising and scanning capabilities** for device discovery
- **iBeacon compatibility** for proximity-based applications

According to Liu et al., (2018), The HM-10 module achieves a typical communication range of 30 to 50 metres in open space and consumes approximately 8 to 10 mA during active transmission, with standby current reduced to as low as 1.5 µA. These characteristics make the HM-10 suitable for proximity verification in the key storage system, where continuous monitoring of the key's location relative to the storage box is required.

### 2.7.3 AT Command Protocol

The AT (Attention) command protocol is a legacy serial communication standard originally developed for Hayes modems in the 1980s, which has been widely adopted for configuring Bluetooth and Wi-Fi modules due to its simplicity and universality (Siekkinen et al., 2012). For the HM-10 module, AT commands are transmitted via Universal Asynchronous Receiver-Transmitter (UART) at a 9600 baud rate and allow comprehensive configuration of module parameters. Table 2 summarises the essential AT commands employed in this project.

Table 2: Essential HM-10 AT Commands for Proximity Detection

| Command | Function | Response |
|---|---|---|
| AT | Verify communication | OK |
| AT+ROLE0 | Set as slave/peripheral (tag mode) | OK+Set:0 |
| AT+ROLE1 | Set as master/central (receiver mode) | OK+Set:1 |
| AT+RSSI? | Query the signal strength of the connected device | +RSSI: -59 |
| AT+DISC? | Discover nearby advertising devices | OK+DISC:MAC, RSSI |
| AT+ADVI5 | Set advertising interval (546 ms) | OK+Set:5 |
| AT+IMME1 | Set manual connection mode | OK+Set:1 |

### 2.7.4 RSSI-Based Distance Estimation

The Received Signal Strength Indicator (RSSI) is a measurement of the power present in a received radio signal, expressed in decibels relative to 1 milliwatt (dBm) (Aparicio et al., 2016). In BLE systems, RSSI values typically range from -30 dBm (very close) to -100 dBm (near the limit of detection), with more negative values indicating greater distance. The relationship between RSSI and physical distance is modelled using the log-distance path loss formula (Rappaport, 2002):

$$RSSI(d) = RSSI(d_0) - 10n\log_{10}\left(\frac{d}{d_0}\right) + X$$

Where:

- $d$ = distance from transmitter to receiver (metres)
- $d_0$ = reference distance (typically 1 metre)
- $n$ = path loss exponent (2.0 for free space, 2.5–3.5 for indoor environments)
- $X$ = Gaussian random variable representing shadowing effects

Solving for distance, the working formula employed in this project is:

$$d = 10^{\left(\frac{|RSSI|-|A|}{10n}\right)}$$

Where $A$ is the average RSSI measured at 1 metre distance from the transmitter (Farid et al., 2013). For the HM-10 modules used in this project, the value of $A$ was experimentally determined to be -59 dBm, representing the signal strength at precisely 1 metre separation in the laboratory environment. The environmental factor $n$ was set to 2.0, appropriate for free-space propagation within the office setting where line-of-sight between the key and the receiver is maintained.

### 2.7.5 Master-Slave Configuration for Proximity Detection

For proximity-based security applications, two HM-10 modules are typically configured in a master-slave topology (Liu et al., 2018). In this project, one HM-10 module attached to the key acts as a **slave (tag)**, continuously broadcasting advertising packets. The second HM-10 module integrated into the key storage box serves as the **master (receiver)**, scanning for these advertisements or maintaining an active connection to the key tag.

Table 3: HM-10 Configuration for Key Proximity Detection

| Role | Function | AT Command Sequence | Power Source |
|---|---|---|---|
| Slave (Key Tag) | Broadcasts advertising packets | AT+RENEW, AT+RESET, AT+ROLE0, AT+ADVI5, AT+NAMETag1 | CR2032 battery |

| | | | |
|---|---|---|---|
| Master (Receiver) | Scans for tag or maintains connection | AT+RENEW, AT+RESET, AT+ROLE1, AT+IMME1 | Box power supply |

According to Winstead & Xu, (2018), The master module can achieve RSSI update rates of 0.5 to 2 Hz in discovery (scan) mode and 10 to 20 Hz in connected mode. The connected mode provides more stable readings with reduced fluctuation at the cost of higher power consumption on the tag side. For this project, the discovery mode was selected to maximise battery life of the key-attached tag while maintaining adequate proximity detection accuracy for the 5-metre threshold.

### 2.7.6 Distance Threshold Implementation for Security

The proximity threshold of 5 metres was selected as the second authentication factor in the key storage system. This threshold ensures that the key holder is physically present at the storage box before access is granted, preventing relay attacks where an attacker could otherwise extend the effective range of a legitimate credential using signal amplification (Park & Hong, 2023). The threshold value was chosen based on empirical testing of RSSI fluctuations in the office environment, balancing security requirements against false rejection rates. Table 4 presents the relationship between RSSI values and estimated distances used in the system.

Table 4: RSSI-Distance Mapping for HM-10 Modules

| RSSI Value (dBm) | Estimated Distance (Metres) | System Action |
|---|---|---|
| -30 to -50 | < 1 metre | Optimal proximity |
| -51 to -65 | 1–5 metres | Access granted (within 5 m threshold) |
| -66 to -80 | 5–15 metres | Access denied, "KEY TOO FAR" alert |
| < -80 | > 15 metres | Access denied, continuous alarm triggered |

### 2.7.7 Accuracy Limitations and Mitigation Strategies

Raw RSSI measurements are known to fluctuate significantly due to multipath fading, environmental interference, and signal absorption by obstacles (Laitinen et al., 2015). Several studies have documented approaches to improve RSSI-based distance estimation accuracy:

- **Moving average filtering**: Smooths temporal variations by averaging recent RSSI readings, reducing short-term fluctuations (Liu et al., 2018)
- **Kalman filtering**: A recursive algorithm that estimates the true signal strength from noisy measurements, achieving approximately 76% noise reduction in indoor BLE applications (Luo et al., 2018)
- **Gaussian filtering**: Removes statistical outliers from RSSI measurements before distance calculation (Zhang et al., 2019)

For this project, a simple moving average filter was implemented in the Arduino code, averaging the last five RSSI readings before distance calculation. This approach balances computational simplicity with adequate noise reduction for the 5-metre proximity threshold application, where millimetre-level accuracy is not required.

### 2.7.8 Standalone HM-10 Operation

Research by Rahman et al., (2018) demonstrated that HM-10 modules can operate without an external microcontroller after initial configuration. The modules retain AT command settings in non-volatile memory, enabling standalone deployment as beacons or remote-controlled devices. In this project, the key-attached HM-10 tag was configured using the AT command sequence shown in Table 3 and then powered independently using a CR2032 coin cell battery, eliminating the need for an additional microcontroller on the key itself. This approach significantly reduced the size, weight, and cost of the key tag while maintaining adequate battery life for extended deployment.

However, for RSSI-to-distance conversion on the receiver side, external processing by the Arduino Mega microcontroller is required due to the HM-10's limited onboard computational capabilities (Liu et al., 2018). The Arduino performs the logarithmic distance

calculation, implements the moving average filter, and executes the access control decision based on the computed distance relative to the 5-metre threshold.

# CHAPTER 3: METHODOLOGY

## 3.1 Objective 1: To determine the design requirements of the lock system

Through the conducted detailed literature review, the design standards, operational requirements, and security features from previous studies and existing commercial systems were used. The analysis was focused on determining functionality, materials, and cost factors relevant to institutional key management needs. These requirements were to address safe physical storage and tamper resistance, robust access control (authentication and administration flows), reliable electro-mechanical actuation, secure firmware and communications, audit and logging, power resilience, usability and human factors, cost and scalability. The published prototypes demonstrated baseline functionality. IoT and commercial systems extend this baseline with remote audit, role management, and hardened security functions.

The system requirements spanned physical, electronic, software and operational considerations. Physically, the design called for a secure, tamper-resistant enclosure fabricated from 1.5 mm mild steel sheet with a 1.5″ hinge, mini door latches and a mechanical emergency-override button accessible only to administrators. Access control was based on a primary five-code PIN entered through a 4×4 matrix keypad, combined with RFID to give a dual-factor authentication concept, all managed by an Arduino Mega microcontroller and protected by an administrator/master code for adding, deleting and maintaining users, with supporting features such as password masking and user session timeouts.

## 3.2 Objective 2: To develop concept sketches, the flow algorithm, and 3D models of preliminary and final designs

To satisfactorily achieve this objective, the following were done and are detailly explained in chapter 4.

### 3.2.1 Concept design, generation, and selection

The sketches were hand-drawn, and the selection was achieved through a method of concept scoring of the five different interior design sketches and two exterior design

sketches of the system. The method utilised the different user need criteria to rank the concepts using ratings which ranged from 1 to 9, with 1 being the lowest and 9 the highest rating. And the concepts were ranked from 1 to 5 using a weighted score.

### 3.2.2 Configuration design

This entailed the dimensions, constraints and material selection of the system. All these were fully mentioned in Chapter 4.

### 3.2.3 Parametric design of the system

This section involved the calculations for the degree of freedom and torque of the rank and pinon that is used to drive the door to open and close, and the power supply voltage requirements and conversions.

### 3.2.4 Hardware architecture development

In this section, the schematic diagram and layout of the electronic components was developed using KiCAD software, the detailed connection of each component in the system to the MCU and the pin allocation of each component in the system.

Under this same section, the control system of the storage box was discussed in detail using control diagrams and how both operations of closing and opening of the door were carried out by the system.

### 3.2.5 Software program/code architecture

This section describes how the programming language, C++, was used in the Arduino IDE to create communication between the different electronic components of the system, application of the number configurations and mathematical models, the pseudo code used to generate the code program in C++, and the relationship between the number configuration and the code program of the system.

### 3.2.6 Detail design of the system

This section describes how SolidWorks was used to generate the different final sketches, drawings, specifications and 3D models of the final design of the system.

## 3.3 Objective 3: To construct and carry out a performance evaluation of the prototype, fully assembled and functional

The prototype of the system was fully assembled and constructed following the 3D model drawings; the design parameters were time-to-time adjusted to best achieve the general objectives, and the best measurable performance was obtained through continued tests with the use of the system in the world real scenario, and the data was gathered and evaluated.

# CHAPTER 4: DESIGN, CONSTRUCTION, AND EVALUTION

## 4.1 Design requirements of key access box system.

### 4.1.1 The design requirements, based on the literature review above

1. A ½ sheet of mild steel of 1.5mm
2. An emergency Button for Admin use only.
3. 5-code PINs and an admin/master code, retry limits (3 attempts and 3 minutes timed lockout) and optional second factor of an RFID.
4. RFID (MFRC-522 RC522 RFID + S50 CARD).
5. 4x4 matrix keyboard.
6. MG90S gear servo motor
7. A 5V DC min solenoid.
8. Battery backup (3.7V 7800mAh 18650 Polymer Lithium Ion Battery (LiPo)).
9. USB power bank charging module.
10. 16x2 I2C LCD Character Module Blue.
11. An Arduino Mega Board as MCU.
12. HDC 3-24V High Decibel Active Buzzer.
13. 5V-3A DC Adapter.
14. 2 in 1 Charger + Boost Converter (DD012CVSA).
15. LED lights for status.
16. 220ohms resistors.
17. 12x18cm Universal PCB DIY prototype board.
18. 5V relay.
19. Wielding.
20. 10mm, and 5mm Bolts and nuts.
21. Door Hinge.
22. Jumper Wires.
23. BLE Beacon Module.
24. A 3D printed Rack and pinion Mechanism.
25. C++ code or program to run the system.

### 4.1.2 Functional and performance requirements implemented

1. A single key that was place inside key box.
2. For Authentication, a 5-digit numeric PIN that is stored securely, an admin/master PIN for provisioning and an optional RFID tag reader as a secondary factor.
3. The Access time for the system is within ≤ 2 seconds of the correct PIN and a lockout of 3 incorrect attempts, with a temporary lockout time of maximum of 3 minutes in case of wrong PIN, which is configurable from 1 to 15 minutes.
4. For Auditing, the system has a local event log with a timestamp, user ID (PIN slot) and an action such as open/close/failure. Log will be stored in non-volatile memory. The admin was able to extract the database or login file to assess the information.
5. The tamper detection system that was able to detect lid removal or box movement and trigger an audible alarm and log event.
6. Power the box system used a mains input and an internal battery backup. The battery has an endurance of greater than or equal to 5 days under normal idle conditions and typical user activity, typically of 15 unlocks per day.
7. The box has a design that is reliable for continuous operation, low-power idle, and a mechanical emergency override accessible to the admin. The key storage box has physically small footprint, with lockable mounting to the wall of the office.

## 4.2 Development of the concept sketches, dimension, material section, and code or program creation

To fulfil this objective, design sketches and, 3D model were created. Since the key storage box is for an individual office, the box was of a relatively small size since it holds a single key, for that office it is assigned to. An improved flow algorithm of the design, the electronic circuit schematics, and control sequence of every system and mechanism in the design was done.

### 4.2.1 Concept design, Generation, and Selection

By determining the institutional needs, the available technology and materials locally, 5 concepts of the inner design and operation were developed, until the final working concept and two outer concepts.

#### 4.2.1.1 The outer/external concepts shown in figure 1 and 2

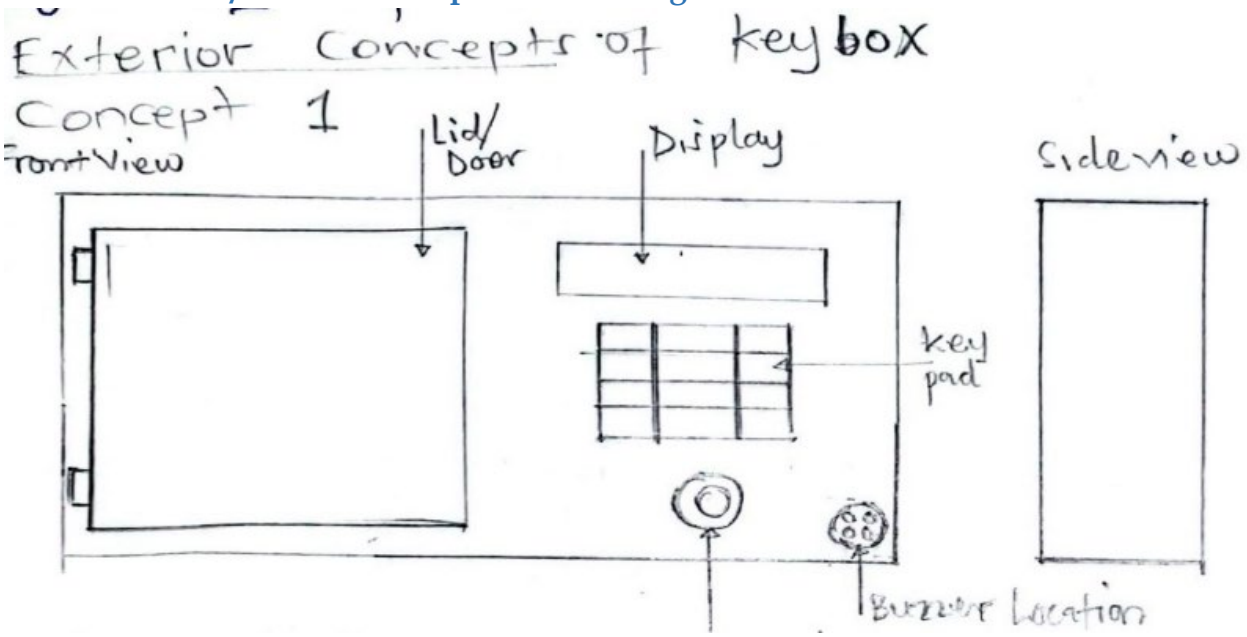


Figure 1: Exterior sketch concept 1

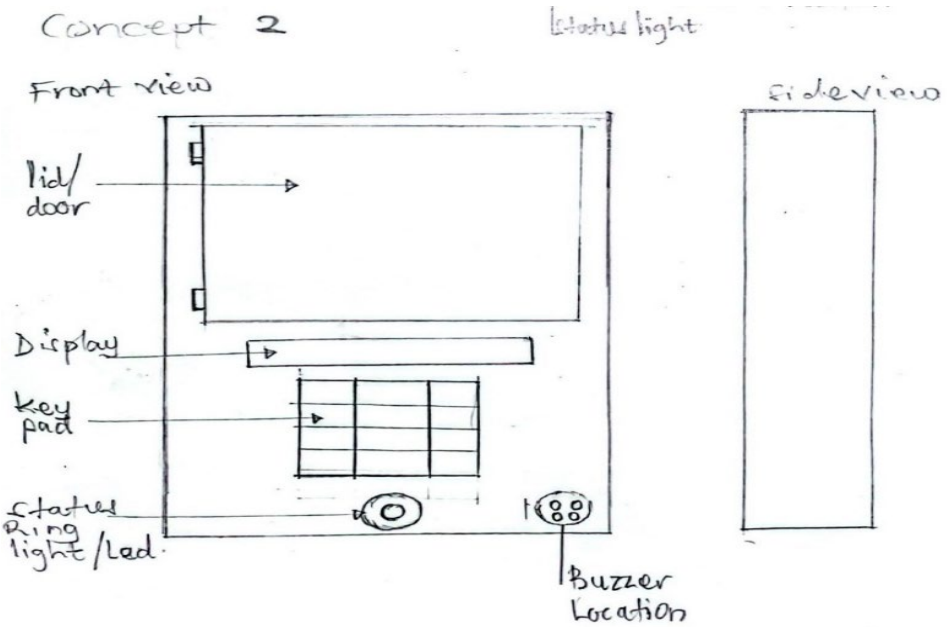


Figure 2: Exterior sketch concept 2

Looking at all the user needs and literature review, the needs in institutional segment were matched with those in the literature review and came up with this the final overall external/ outer concept as seen in figure 2. This final concept is the combination of the two first concept but improve in dimension and size to accommodate all the electronic used in the key storage box.

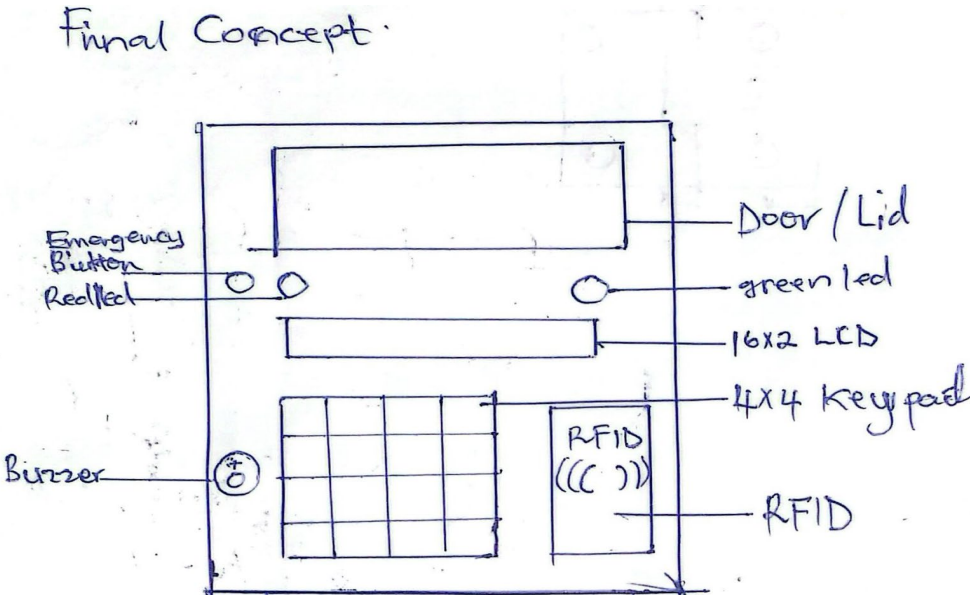


Figure 3: The final revised external concept

#### 4.2.1.2 The internal concepts shown in figure 4 to 6

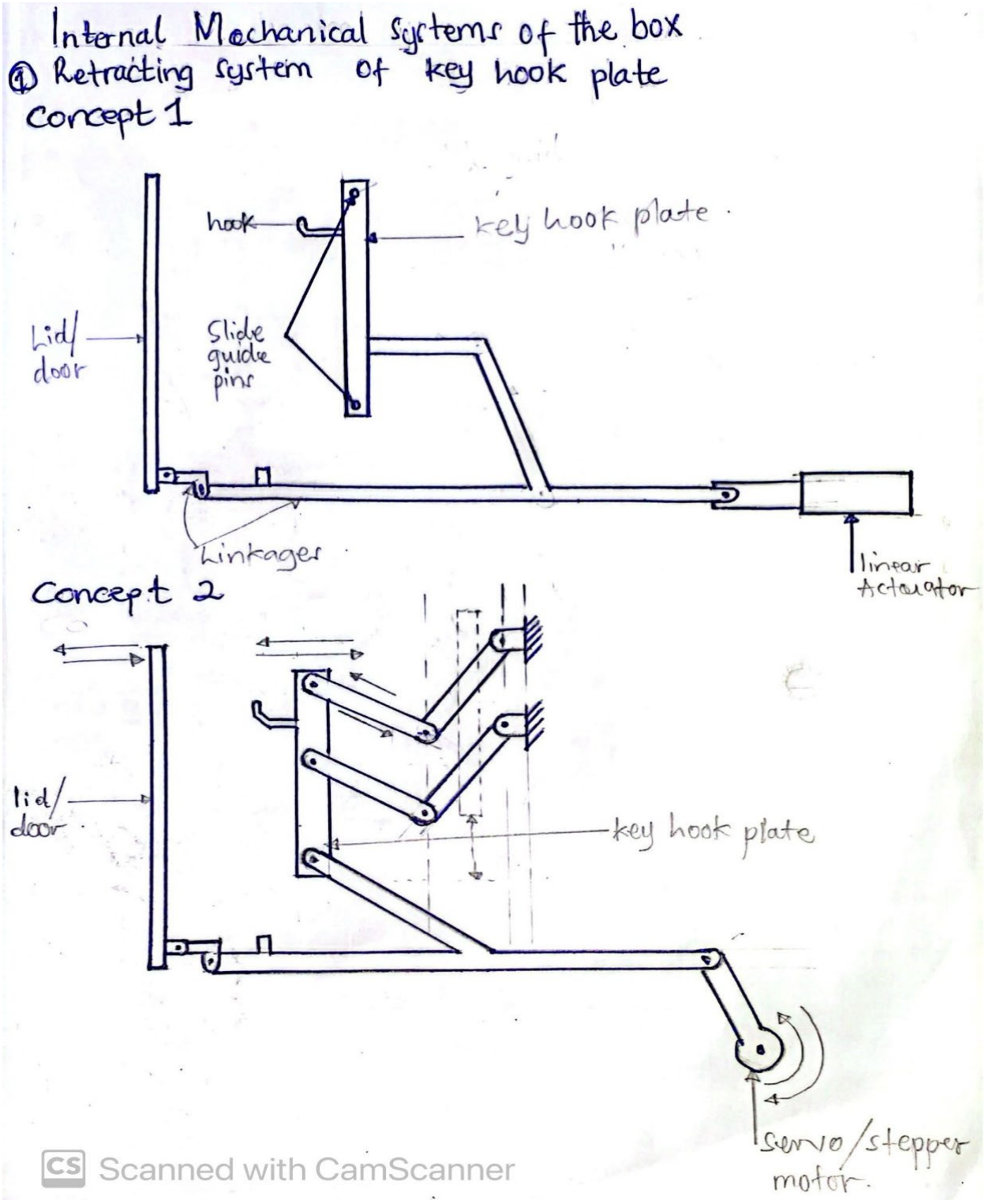


Figure 4: concept 1 and concept 2

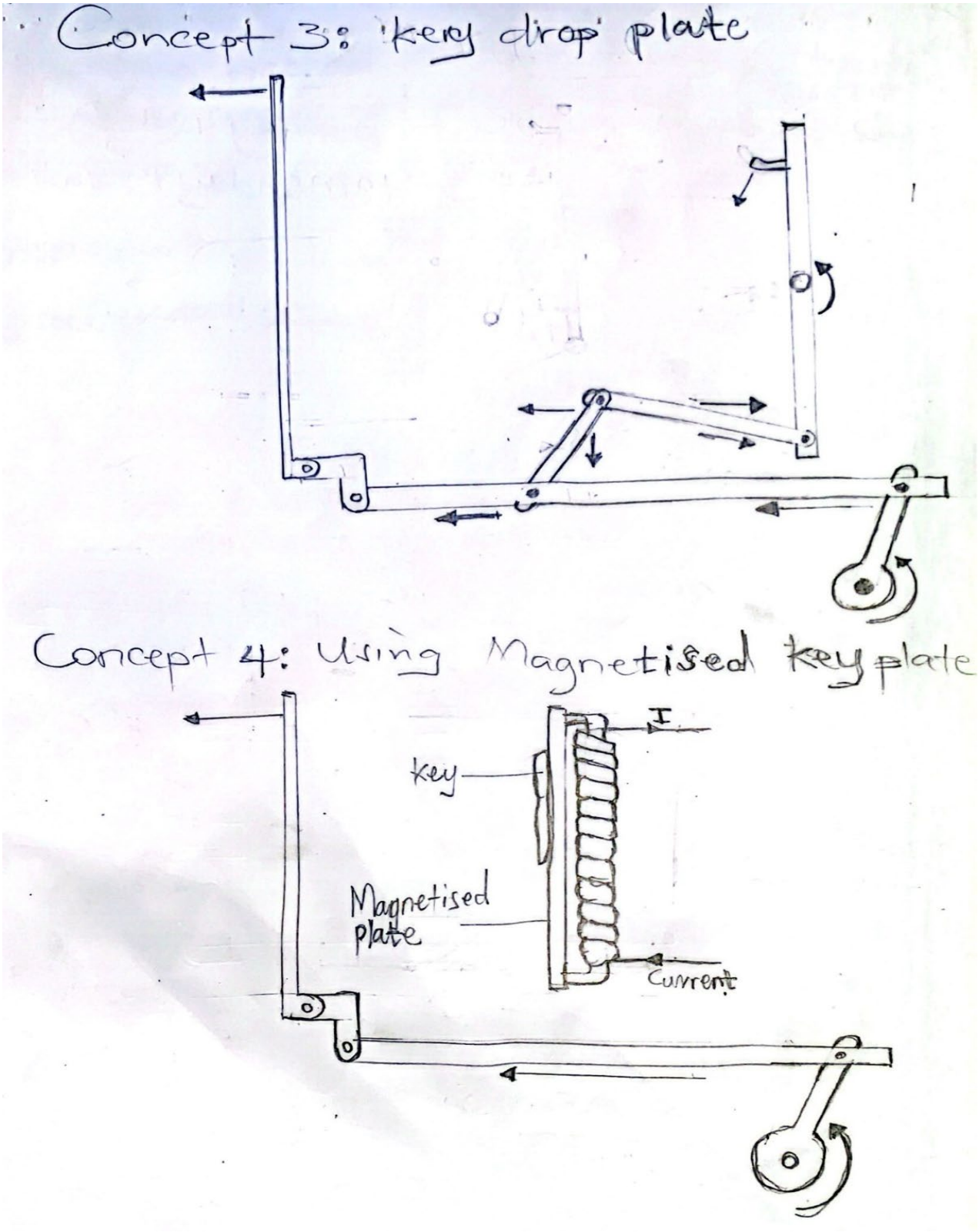


Figure 5: concept 3 and concept 4

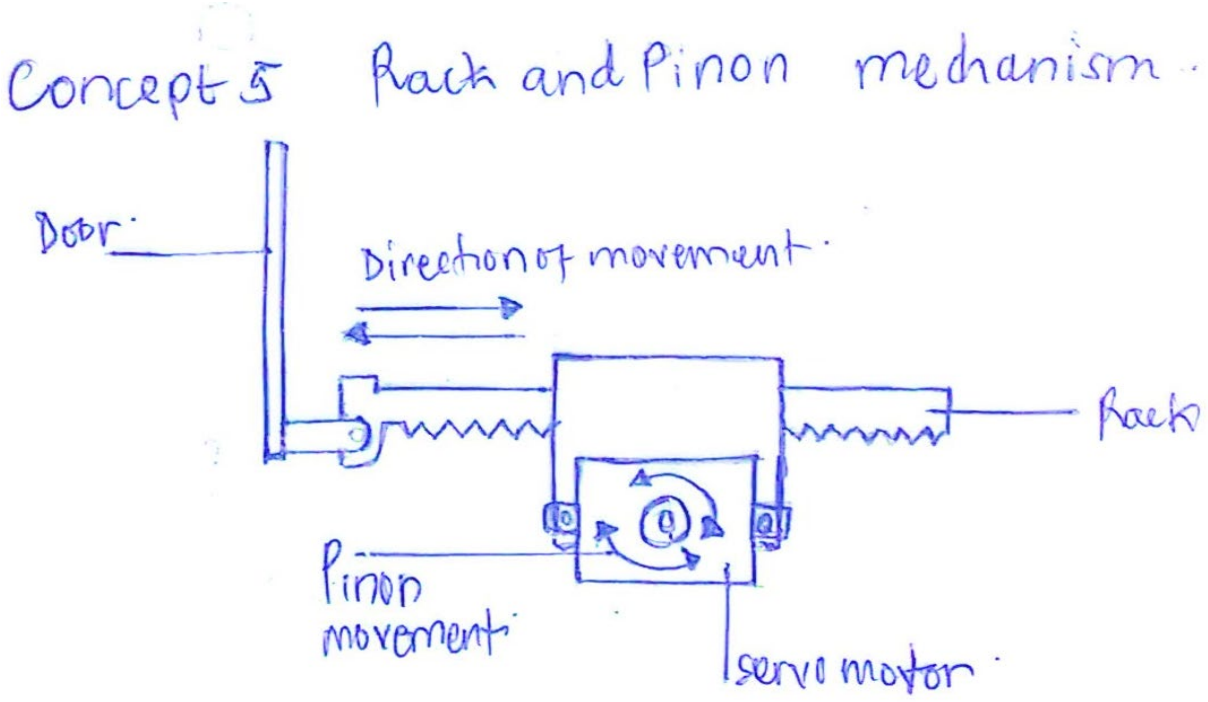


Figure 6: concept 5

#### 4.2.1.3 Internal Concept Selection for Key Box

The selection process here was influenced by both internal and external factors. These factors were cost, arrangement of the internal electronics, customer needs, ergonomics (removal), application, and environmental factors.

Using the method of concept scoring, one concept was chosen, and which was late improved to final internal concept. The scores are divided into two, ratings, weight, and weighted score. The rating ranges from 1 to 9, with 1 being the lowest and 9 the highest rating. The weight is percentage of the criteria selected and weighted score is the product of rating and weight.

A concept with highest weighted score was the best and had highest application in this project. Below were how the concepts, 1 to 5 were scored and ranked.

Table 5: Showing concept scoring of interior concepts

| | | CONCEPTS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CONCEPT 1 | | CONCEPT 2 | | CONCEPT 3 | | CONCEPT 4 | | CONCEPT 5 | |
| criteria | weight | rating | weighted score | rating | weighted score | rating | weighted score | rating | weighted score | rating | weighted score |
| A | 10% | 3 | 0.3 | 2 | 0.2 | 4 | 0.4 | 1 | 0.1 | 7 | 0.7 |
| B | 30% | 5 | 1.5 | 2 | 0.6 | 2 | 0.6 | 1 | 0.3 | 9 | 2.7 |
| C | 20% | 4 | 0.8 | 3 | 0.8 | 3 | 0.6 | 2 | 0.4 | 8 | 1.6 |
| D | 30% | 7 | 2.1 | 7 | 2.1 | 7 | 2.1 | 1 | 0.3 | 9 | 2.7 |
| E | 20% | 2 | 0.4 | 3 | 0.6 | 4 | 0.8 | 6 | 1.2 | 10 | 2.0 |
| Rank | ascending | | 3 | | 5 | | 4 | | 2 | | 1 |
| Total scores | 100% | | 5.1 | | 4.3 | | 4.5 | | 6.8 | | 9.7 |

The base criteria were as listed below,

A- Applicability (how easy is it to implement in the real world)

B- Cost (high-cost lower rating and low-cost higher rating)

C- Arrangement of the internal electronics

D- Ergonomics (easy to access or remove)

E- Security (tamper resistance)

The concept that had the highest weighted score was concept 5, followed by concept 4, and the lowest was concept 2.

### 4.2.2 Configuration design (dimensions, constraints & materials selection)

#### 4.2.2.1 Dimensions and Constraints of the System

The external enclosure was made of cast or mild steel or stainless steel, 1.5 mm thick and includes anti-tamper screws and a fixed mounting plate. The internal volume and dimensions are 300 mm × 200 mm × 150 mm for the height, width, and length, respectively. The door has a link connected to a rack and pinion drive mechanism, which is also used to open and close the door automatically, connected to 360-degree servo motors.

The door has a dimension of 160mm x 90mm, and has two 1.5" hinges, an electronic solenoid latch attached to it, and a closed micro limit switch on the lid to detect when opened or closed. Once the correct code was punched in, the door was automatically opened with the assistance of a servo motor and rack and pinion mechanism. Once the key was placed back or picked up, the door was closed automatically after a period of 20 seconds.

The dimensions of the rack and pinion mechanism considered were as follows in Figures 7 and 8;

**Rack retainer and servo motor attachment dimensions**

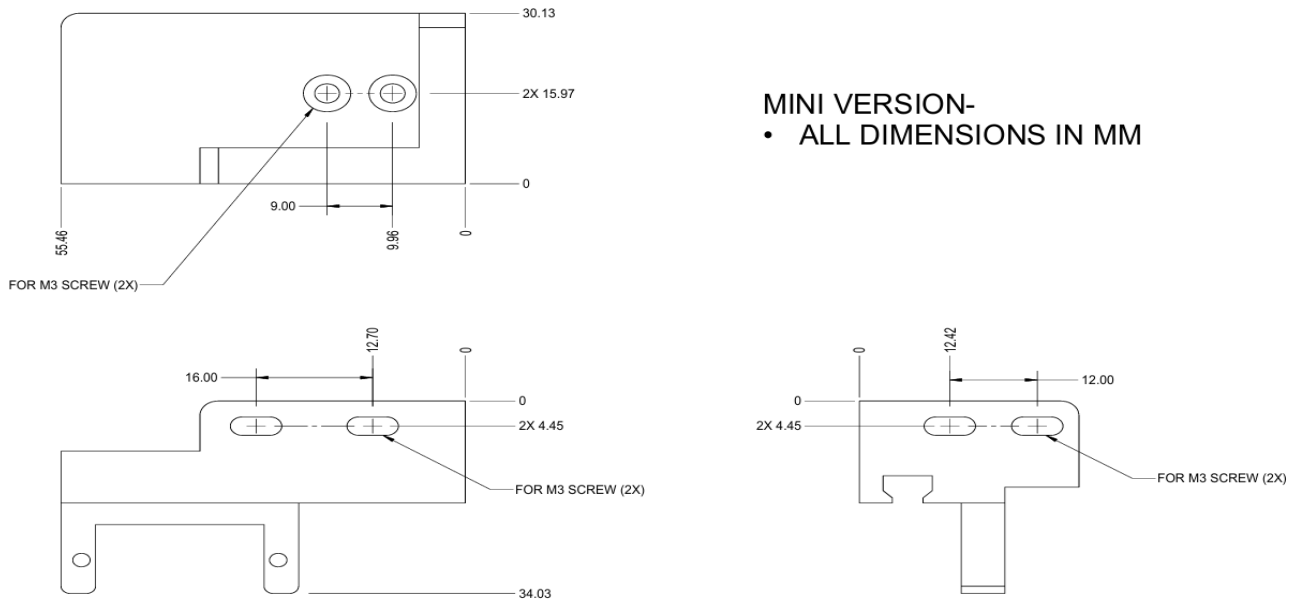


Figure 7: Servo Bracket dimensions

## Rack Pusher dimensions (The rack pusher length used was 100mm)

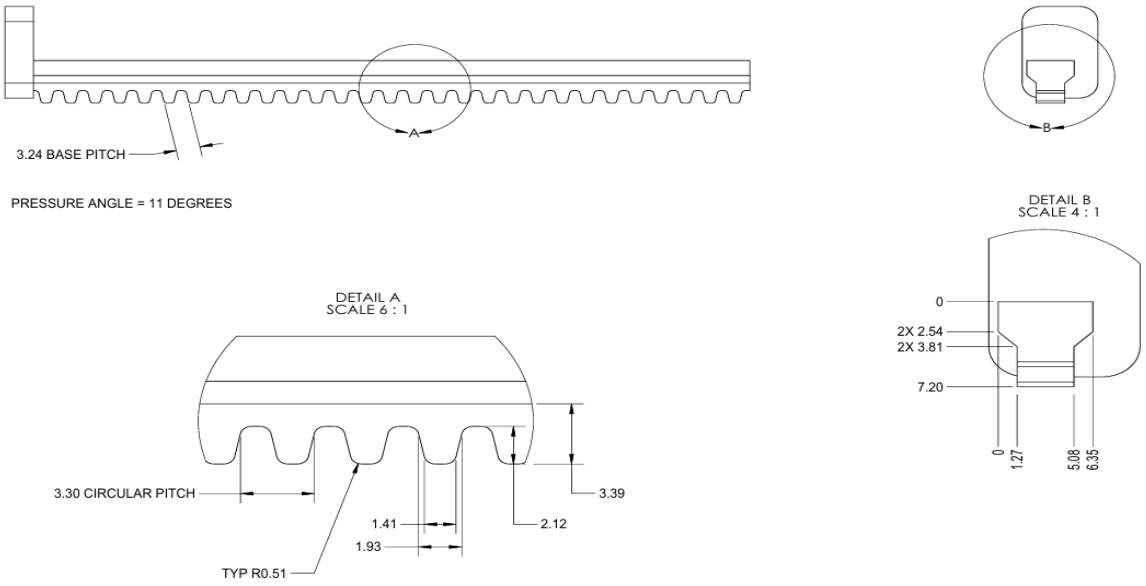


Figure 8: Rack Pusher Dimensions

## 3D model of rack and pinon system

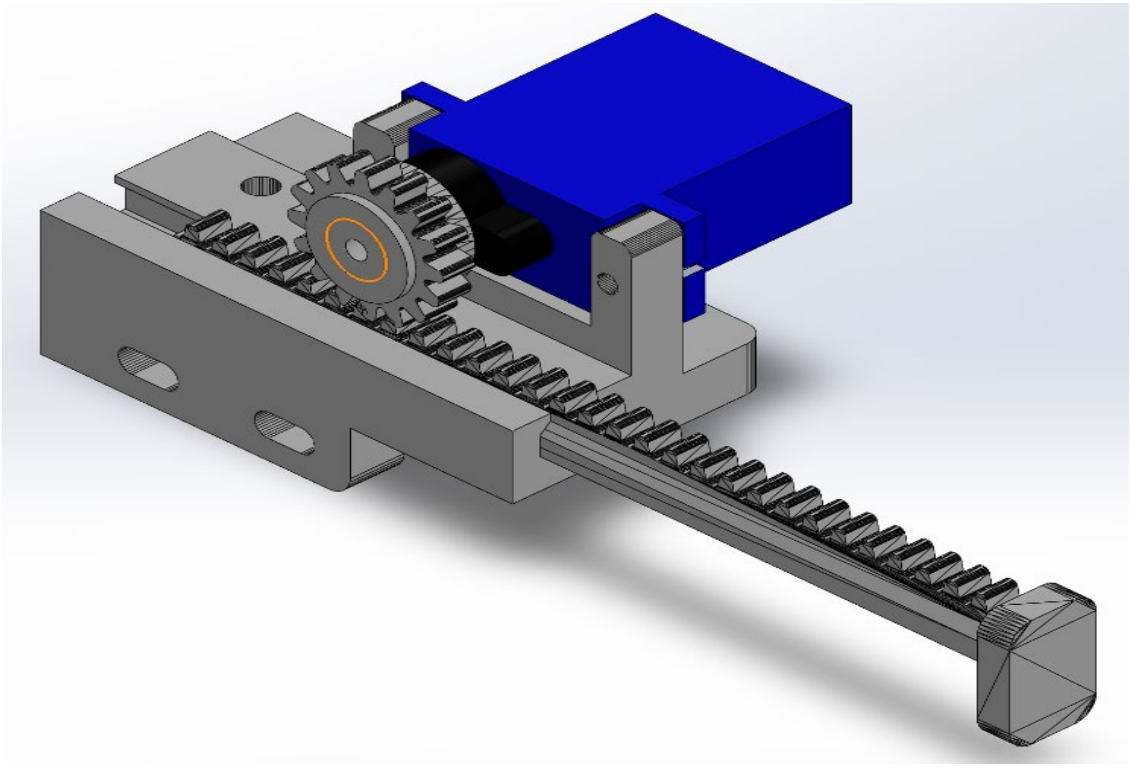

Figure 9: 3D model of the complete assembly of the rack and pinion

## Gear Calculations

Table 6: Gear Data

| Parameter | Value |
|---|---|
| Base pitch | 3.24 mm |
| Pressure angle (φ) | 11 degrees |

Using the base pitch and pressure angle, the circular pitch was calculated from the formula,

$$p_b = p \times \cos(\phi)$$

Where:

- $p_b$ = Base pitch = 3.24 mm
- $\phi$ = Pressure angle = 11°
- $p$ = Circular pitch (what we solve for)

Calculation

$$p = \frac{p_b}{\cos(11°)}$$

$$\cos(11°) = 0.981627$$

$$p = \frac{3.24}{0.981627} = 3.30 \text{ mm}$$

Circular pitch $p = 3.30$ mm

Module (m)

$$p = \pi \times m$$

$$m = \frac{p}{\pi} = \frac{3.30}{3.14159} = 1.05 \text{ mm}$$

Module $m = 1.05$ mm

### 4.2.2.2 Material selection

Based on the institutional needs, mild steel was the material best suited to craft the exterior casing of the key box because it had the necessary properties like its toughness(Resists bending and deformation), relatively cheap and widely available, making it ideal for

prototyping project where budget matter, easy to fabricate and machine using arch wielding method, can easily be coated for corrosion Resistance using paint and affordable.

Rack and pinion systems were 3D printed from PLA using 3D printers because it was easy to fabricate (quick to print parts), had low material cost, and were strong enough for low-load bearing mechanical systems.

### 4.2.3 Parametric design of the system

#### 4.2.3.1 Degree of freedom and torque of the system mechanism

The mechanism of the door was driven by servo motors that are connected to a rack and pinion mechanism, which possesses a certain degree of freedom. At first, the key drop plate was to be driven by the linkage system, which possessed 4 linkages. The degree of freedom (DoF) for that setup was as follows.

- DoF = $3(L-1)-2J_1 - J_2$
- $J_1 = 3$, $J_2 = 1$, and $L=4$
- DoF = $3(4-1) -(2x3) - 1$
- DoF = 2, for the linkage system.
- L – number of linkages
- $J_1$ – number of joints with 2 DoF
- $J_2$ - number of joints with 1 DoF

For the utilisation of 5V servo motor torque, the rack and pinion mechanism was the best option. The degree of freedom of this mechanism was 1. Since it could convert the rotation motion of the pinion to the linear motion of the rack.

The servo motor is connected to a rack and pinion system, then to the main link that either open and closes the door, as illustrated in the concept sketches. The initial concept of the servo motor system utilised linkages to drive the door. The $360^0$-servo motor has a torque of 0.18Nm at 4.8V or 0.23Nm at 6V

A 1.5mm sheet of steel has 11.8 $kg/m^2$ of weight/ mass. This implies that;

$1x10^6$ $mm^2$ area = 11.8kg,

$1mm^2$ area = $1.18x10^{-5}$ kg,

For the door alone,

Area = 100x90 mm = 9000 $mm^2$

Weight (w) = $1.18x10^{-5}$ x 9000 = 0.1062 kg

Torque; τ = force x distance, using a rack of 100mm length to drive the door,

force = weight (w) x acceleration due to gravity (g) = 0.1062 x 9.81 = 1.041822 N.

Therefore, Torque, τ = 1.041822 x 100/1000 = 0.1041822Nm of torque.

For the door and the solenoid,

Weight of solenoid = 0.130Kg

Force = 0.130 x 9.81 = 1.28 N

, Torque, τ = 1.28 x 0.1 = 0.128Nm of torque

Total torque = 0.128 + 0.1042 = 0.2322 Nm

#### 4.2.3.2 Power Supply Voltage Conversion Calculations

The system used a 5V DC adapter, which was suitable for Arduino and other modules.

DC-DC converter power balance

$$P_{in} = P_{out}$$
$$V_{in}I_{in} = V_{out}I_{out}$$

So, if 5V circuit consumed 2A:

$$I_{in} = \frac{V_{out}I_{out}}{V_{in}} = \frac{5 \times 2}{5} = 2A$$

This helped in selecting the correct 5V adapter rating.

#### 4.2.3.3 Total Current Consumption Calculation

Total current = (summing current drawn by all loads)

$$I_{total} = I_{Arduino} + I_{LCD} + I_{RFID} + I_{Servo} + I_{Solenoid} + I_{BLE} + I_{LEDs} + I_{Buzzer}$$

Example estimate:

- Arduino Mega = 50 mA
- LCD = 20 mA
- RFID RC522 = 30 mA
- HM-10 = 30 mA
- Buzzer = 30 mA
- LED (each) = 20 mA
- Servo MG90S (each) = 500 mA peak
- Solenoid (each) = 500 mA to 1A

#### 4.2.3.4 Battery Backup Capacity (Runtime) Calculation

A 3.7V, 7800mAh battery,

Battery energy:

$$Wh = V \times Ah$$
$$Wh = 3.7 \times 7.8 = 28.86Wh$$

Runtime estimation:

$$t = \frac{Battery\ Energy}{Load\ Power}$$

If the system consumed 2W average, then

$$t = \frac{28.86}{2} = 14.43 \text{ hours}$$

Resistor Calculations for LEDs (Current Limiting)

The design used 220Ω resistors for red and green LEDs.

From Ohm's law:

$$R = \frac{V_{supply} - V_f}{I}$$

For Arduino 5V output and LED forward voltage = 2V = $V_f$:

$$R = \frac{5-2}{0.02} = 150\Omega$$

Using 220Ω reduced current draw:

$$I = \frac{5-2}{220} = 0.0136A = 13.6mA$$

This is was safe for Arduino pins.

#### 4.2.3.5 5V relay Driver Calculations for Solenoid Control

The solenoid ran on 5V, but Arduino provides 5V logic, so a 5V relay is required.

Solenoid power:

$$P = VI$$

If solenoid current is 0.8A:

$$P = 5 \times 0.8 = 4W$$

This helped in relay and adapter selection.

#### 4.2.3.6 Flyback Diode Protection Calculation

Solenoids generated back EMF when switched OFF.

Energy stored in the solenoid:

$$E = \frac{1}{2}LI^2$$

A flyback diode provided a safe discharge path, preventing Arduino damage.

#### 4.2.3.7 Voltage Divider Calculation (HM-10 RX Protection)

Your report uses a voltage divider for the HM-10 RX input using 10kΩ and 20kΩ.

Voltage divider formula:

$$V_{out} = V_{in}\frac{R_2}{R_1 + R_2}$$

If Arduino TX = 5V, HM-10 RX needed = 3.3V, then:

$$V_{out} = 5 \cdot \frac{20}{10+20}$$

$$V_{out} = 5 \cdot \frac{20}{30} = 3.333V$$

This protected the HM-10 module.

#### 4.2.3.8 Fuse Rating Calculation (Battery Protection)

Fuse rating was above normal current but below dangerous current.

$$I_{fuse} = 1.25 \times I_{normal}$$

If the system were normally to draw 1A, then:

$$I_{fuse} = 1.25 \times 1 = 1.25A$$

So, the selected fuse was of 1.5A.

#### 4.2.3.9 Servo Motor Power Requirement Calculation

The MG90S servo typically uses 4.8–6V.

Power:

$$P = VI$$

If the servo was at peak current = 0.7A:

$$P = 5 \times 0.7 = 3.5W$$

$$P = 3.5W$$

This is why servo motors use external 5V, not the Arduino 5V pin.

#### 4.2.3.10 Power Loss in Resistors (LED resistor check)

$$P = I^2R$$

Using LED current 13.6mA:

$$P = (0.0136)^2 \times 220 = 0.0407W$$

So, a ¼W resistor was safe.

#### 4.2.3.11 Adapter Rating Calculation

The adapter supplied the maximum expected load.

$$I_{adapter} \geq I_{total\,peak}$$

If peak load was 2.5A:

$$Adapter = 5V,\ 3A$$

A 5V 3A adapter was selected.

#### 4.2.3.12 Distance Estimation Electrical Model (RSSI-Based)

THE BLE RSSI model was used:

$$Distance = 10^{\left(\frac{MeasuredPower - RSSI}{10N}\right)}$$

With measured power -59dBm and N=2:

$$Distance = 10^{\left(\frac{-59 - RSSI}{20}\right)}$$

This was used for proximity-based security.

### 4.2.4 Hardware architecture development

The microcontroller has a machine control unit (MCU), ATmega328P, on Arduino Mega to extend battery life. The Input interface has a 4×4 matrix keypad with an RFID reader MFRC522 for PIN input. The user feedback on the system has a small 16x2 I2C LCD, one buzzer and 2 status LEDs, both to indicate status, the green indicates whether the right password was used or the Door is open, and the Red one indicates if a wrong password is used or the alarm for forced entry or the key is at a far distance from the Storage box. For real-time clocking, an RTC: DS3231 real-time clock module is used for accurate timestamps in the system. Power management of the system utilised a 5V DC mains adapter that runs from the mains to charge the battery and power the system. A Low battery & charging detection and monitoring system was used to detect battery voltage and the charge controller, which can handle charge termination. A Fuse & protection was inline with the battery lead for reverse polarity protection, and transient suppression.

### 4.2.4.1 Schematic diagram showing the traces of the electronic Components in the System

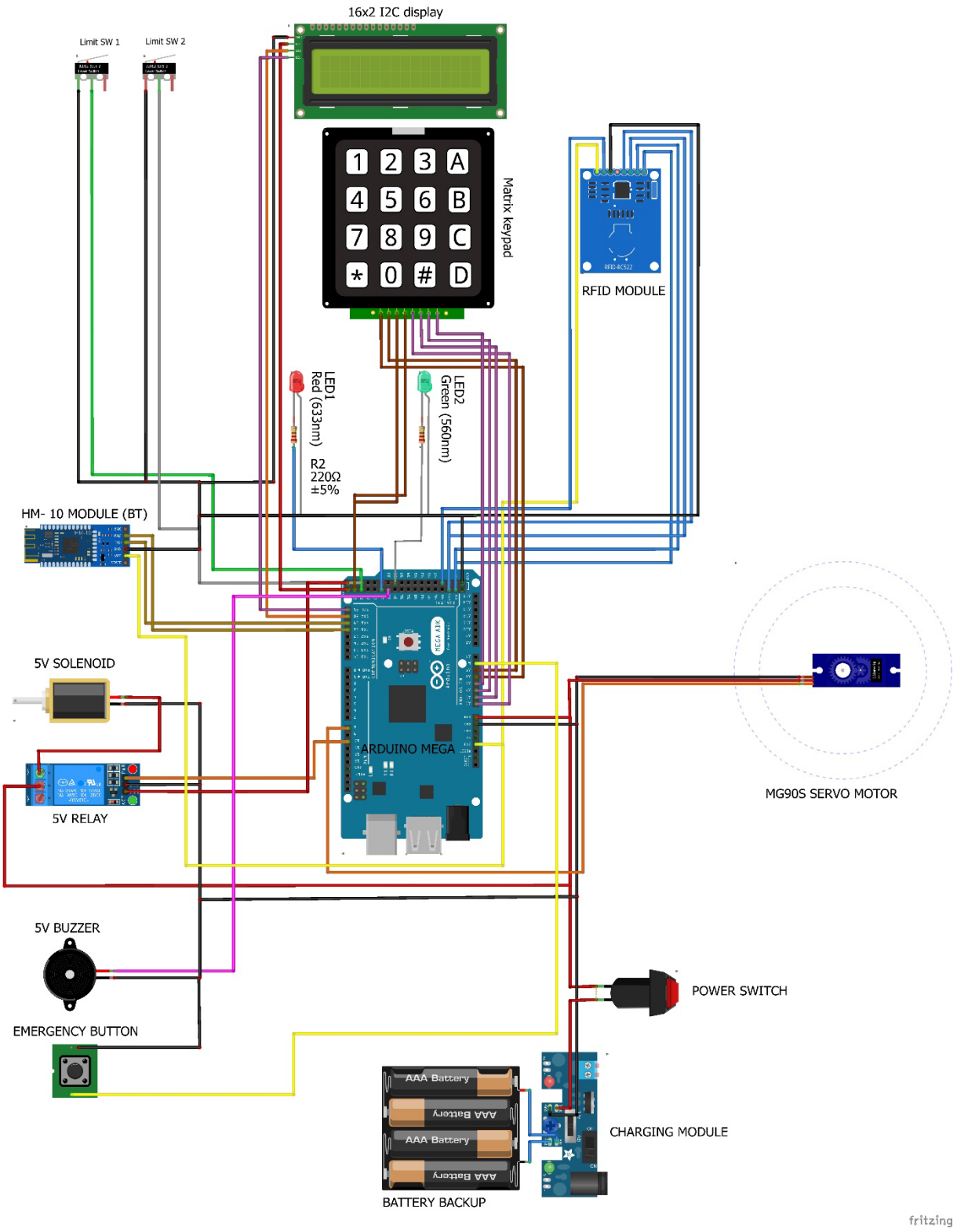


Figure 10: The system schematics

Using Ki CAD software, a system schematic showing all the traces of the different electronics involved in the system was drawn. Below was how the different components were connected to the MCU/ Arduino MEGA in Figure 10 above.

### 4.2.4.2 The control system of the key box

Figures below illustrate how the key storage box control systems operated; it has both a forward and a feedback path, which are as shown in Figure 16.

**The door control system diagram and operations**

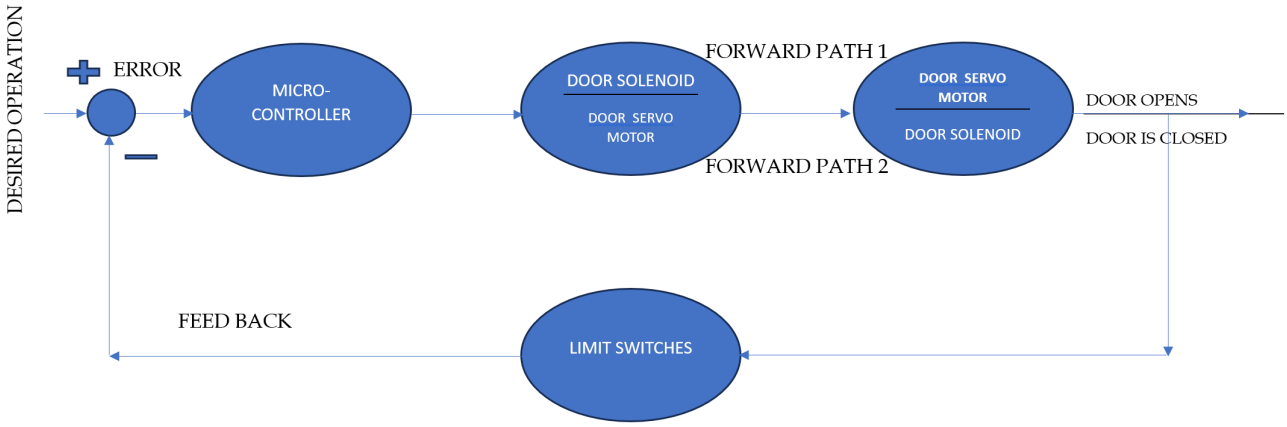


Figure 11: Door control system

**The desired operations included:**

1. PIN was input to open the door.
2. Automatic closing of the door after placing the key back.
1. Push to engage the closing of the door.

**There were two forward paths depending on the operation being engaged.**

1. Forward path 1 was for opening the door. In this path, if the PIN input was determined to be right by the microcontroller, the door solenoid was dispatched, followed by the door servo motor pushing the door outwards to open until the door limit switch 1 was triggered to stop the servo motor.
2. Forward path 2 was to engage the closing of the door if either pushed or automatically closing after 5s when left open. When the rack pusher end triggered the limit switch,

with precise positioning, the stopping of the servo motor movement was done by door limit switch 2, and the solenoid was requested to lock the door.

#### 4.2.4.3 Key Tag Hardware Configuration (Transmitter Side)

The key tag side of the proximity detection system consisted of a single HM-10 Bluetooth Low Energy (BLE) module configured as a standalone iBeacon transmitter. Unlike the receiver module integrated into the key storage box, the key tag module operated independently without an external microcontroller, significantly reducing its size, weight, and power consumption. This design choice enabled the tag to be physically attached to the office key.

Table 7: Key Tag Hardware Components

| **Component** | **Specification** | **Function** |
|---|---|---|
| HM-10 BLE Module | CC2541 chipset, 2.4 GHz | Broadcasts iBeacon advertising packets |
| CR2032 Coin Cell Battery | 3V, 220 mAh | Standalone power source |
| Voltage Regulator (internal) | 3.3V | Regulates battery voltage for CC2541 |

The HM-10 module was selected for the key tag application due to its ultra-low power consumption characteristics. The average current draw was minimised, extending the operational life of the CR2032 coin cell battery to approximately six days at a 760 ms advertising interval (AT+ADVI5 setting).

##### 4.2.4.3.1 Standalone Operation Wiring

After programming, the key tag HM-10 was disconnected from the Arduino and powered independently using a CR2032 coin cell battery. The standalone wiring was as in Table 8:

Table 8: Standalone Key Tag Wiring

| **HM-10 Pin** | **CR2032 Battery Connection** |
|---|---|

| | |
|---|---|
| VCC | Positive (+) terminal |
| GND | Negative (-) terminal |

The absence of connections on the TX and RX pins during standalone operation did not affect the module's ability to broadcast iBeacon advertisements, as the advertising function operates independently of the serial interface once configured.

### 4.2.5 Software and code Architecture

The programming language that was used was C++, which was written in the Arduino IDE environment, which was used to compile and upload the code to the MCU (Arduino Mega). The whole program/ code was placed in the appendix.

#### 4.2.5.1 The 5- code mathematical operations and modelling

The system utilised permutations of digit passwords for authentication. Below was the concept that the system employed during password authentication.

Table 9: Permutations and combinations of digits

| **Concept** | **Meaning** | **Formula** | **Usage in Password Systems** |
|---|---|---|---|
| Permutation (P) | The arrangement of items where order matters. | $P(n,r) = \frac{n!}{(n-r)!}$ | Used when digits cannot be repeated (e.g., 1234 ≠ 4321). |

The PINs depend on the order of digits; permutations are the correct model to use rather than combination models.

#### 4.2.5.2 Using permutations in a 5-code/ digit password system.

**For case 1: Digits were repeated**

If each of the 5 positions contains any of the 10 digits (0–9):

$$\text{Total passwords} = 10^5 = 100{,}000$$

**For case 2: Digits were not repeated**

$$P(10,5) = \frac{10!}{(10-5)!} = 10 \times 9 \times 8 \times 7 \times 6 = 30{,}240$$

So, 30,240 unique 5-digit passwords are possible.

#### 4.2.5.3 Password Strength and Probability of Guessing

If an unauthorised person attempted to guess a PIN at random:

$$P(\text{success}) = \frac{1}{N}$$

Where N is the total number of possible passwords.

### Probability of Guessing

Table 10: Probability of Guessing

| Password Type | Total Passwords | Probability of Correct Guess | Security Level |
|---|---|---|---|
| 5-digit (with repetition) | 100,000 | $\frac{1}{100,000} = 0.00001$ | High |
| 5-digit (no repetition) | 30,240 | $\frac{1}{30,240} = 0.000033$ | High |

#### 4.2.5.4 Multi-User Key Box Code Assignment

We Assumed:

- Each user was assigned one unique 5-digit code (no repetition).
- The office had a maximum of 10 members.

From 30,240 available codes:

$$\text{Percentage of codes used} = \frac{10}{30,240} \times 100 \approx 0.0331\%$$

Hence, there was ample room to expand the user base while keeping the codes unique and secure.

Implementing a 5-digit permutation-based code provided optimal security for multi-user key management systems while maintaining usability and low computational cost.

#### 4.2.5.5 The number configuration of the system using permutations and probability was as follows in tables 11 to 14:

Table 11: Password Space / Permutations

| Case | Formula | Total Possible 5-Digit Codes |
|---|---|---|
| Digits can repeat (default assumption for keypad entry) | $10^5$ | 100,000 |
| Digits cannot repeat (more restrictive) | P(10,5) =10×9×8×7×6 | 30,240 |

The system uses repetition allowed (typically for PIN keypads), giving 100,000 unique codes.

Table 12: Probability of Guessing a Correct Code (Single Attempt)

| Scenario | Probability |
|---|---|
| One random guess (repetition allowed) | $\frac{1}{100,000} = 0.00001$ |
| One random guess (no repetition) | $\frac{1}{30,240} = 0.000033$ |

Table 13: Multi-user Code Assignment (for up to 10 users)

| Calculation | Result |
|---|---|
| Percentage of code space used (10 out of 30,240 unique non-repeating codes) | $\frac{10}{30,240}$×100 = 0.0331% |
| Implication | Vastly more than enough capacity; no risk of collision. |

Table 14: Entropy (Security Strength)

| Formula | Value |
|---|---|
| Entropy= $n \times \log_2(b)$ where $n$=5, b=10 | 5 x 3.3219 = 16.61 **bits** |

16.6 bits was considered moderate security; combined with the 3-attempt lockout, it provided adequate protection for an office key box.

#### 4.2.5.6 Pseudo-code of the system

Figures 12 to 13 show the different flow stages and development of the code in pseudo-code

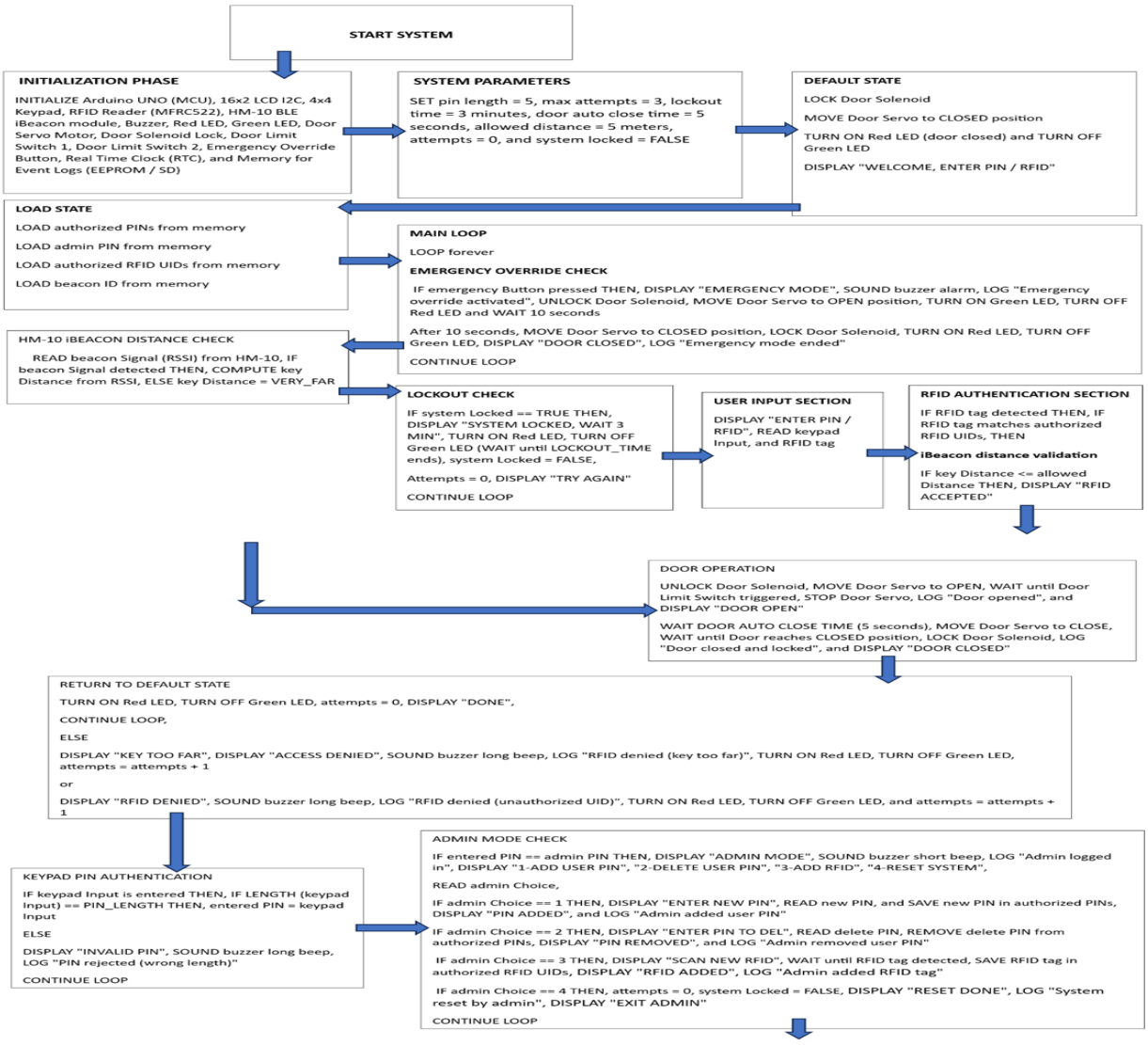


Figure 12: Pseudo code flow 1

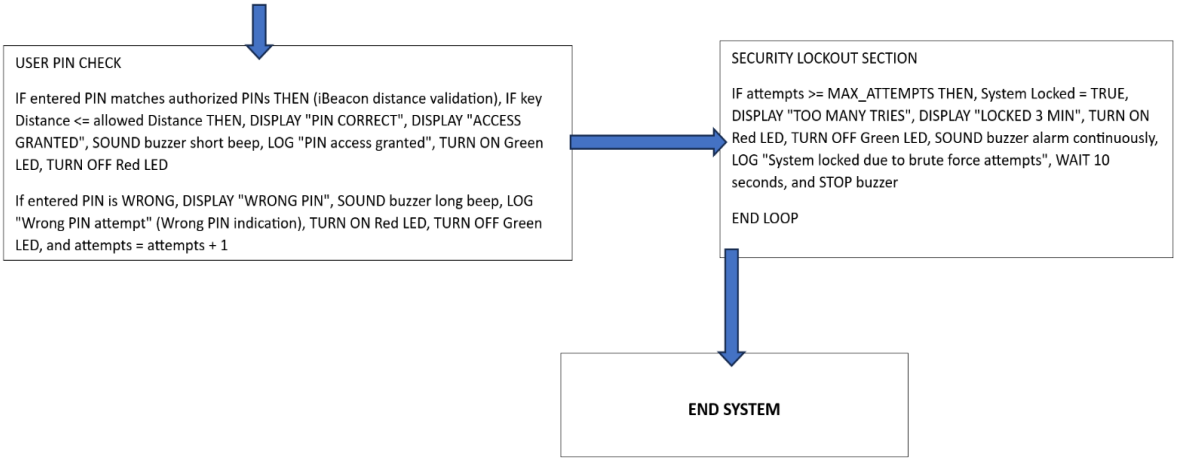


Figure 13: Pseudo code flow 2

#### 4.2.5.6.1 AT Command Configuration Sequence

After establishing serial communication via the pass-through code, the following AT command sequence was transmitted to configure the HM-10 as a standalone iBeacon tag. Each command was sent through the Serial Monitor (9600 baud, "Both NL & CR" line ending) and verified by the OK response. These commands are shown in Table 18 in the Appendix.

#### 4.2.5.6.2 Advertising Interval Selection and Battery Life Trade-off

The advertising interval parameter (AT+ADVIx) directly influenced both the responsiveness of the distance detection system and the operational battery life of the key tag. A shorter advertising interval resulted in more frequent broadcast packets, providing the receiver with more frequent RSSI updates and thus more responsive distance tracking, but at the cost of higher average current consumption and reduced battery life. Conversely, a longer advertising interval extended battery life but reduced the update rate of distance estimates, as shown in Table 19 in the appendix.

The AT+ADVI5 setting (760 ms interval) was selected as it provided a reasonable balance between distance detection responsiveness and battery life, allowing approximately one week of continuous operation before battery replacement was required.

#### 4.2.5.6.3 Integration with Receiver Side

The key tag's broadcast name "key Tag" was defined as the TARGET_TAG_NAME constant in the receiver firmware. The receiver's readBeaconDistance() function continuously scanned for devices advertising this name, extracted the RSSI from the received packets, applied the moving average filter, and calculated the distance using the log-distance path loss model. This design ensured that the receiver only responded to the designated key tag and ignored other BLE devices in the vicinity.

The complete two-way communication architecture is summarised in Table 17.

### 4.2.6 Modelling and drawings of the system in SolidWorks.

The system has different sections of the hardware components. The focus of modelling was on the external components, which were constructed from the steel plates and a few electronic components whose positioning was vital and aesthetic.

These included: -

- External Steel Casing
- Rack And Pinion Mechanism
- Servo Motor

#### 4.2.6.1 The modelling of the casing (exterior)

The exterior was divided into six parts: the front plate, the back plate, the left side plate, the right side plate, the top plate, and the bottom plate. The plates were assigned cast steel as the material during modelling. All these plates were modelled in SolidWorks with appropriate dimensions, as mentioned and shown in Figures 14 to 17.

Figure 17 shows 3D model of fully assembled system

## System drawings

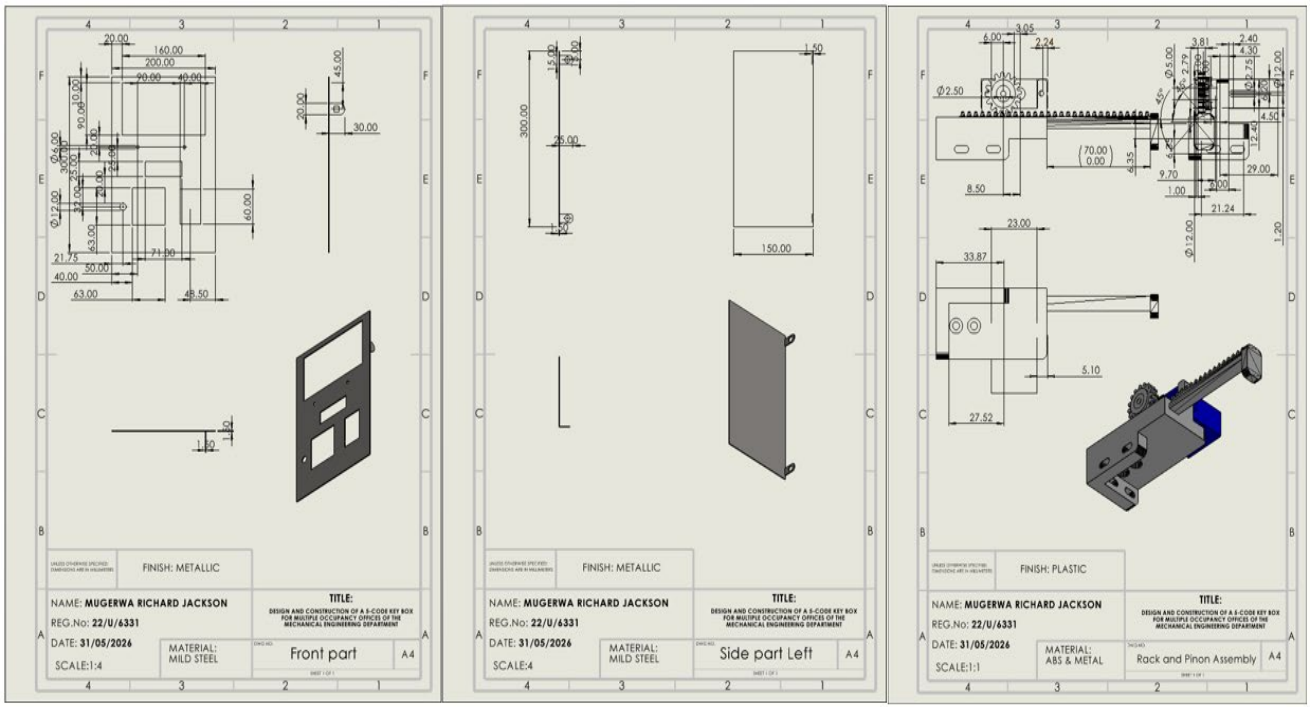


Figure 14: Drawings of some of the modelled Parts of the casing

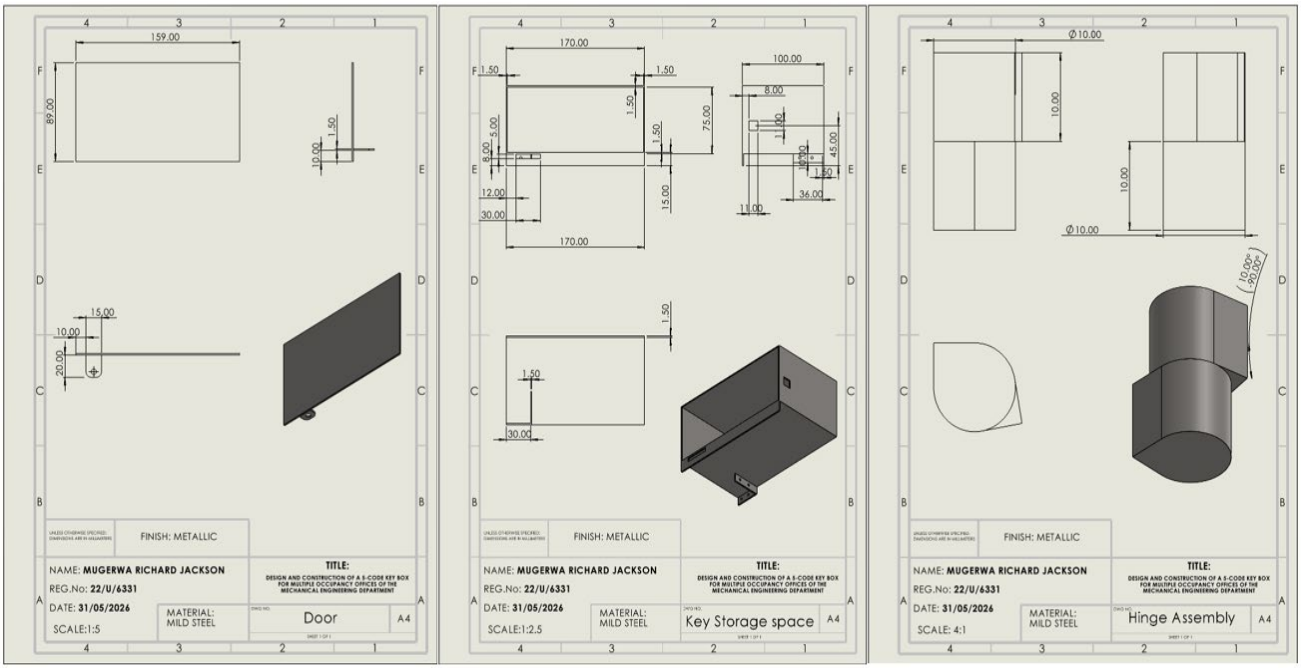


Figure 15: Drawings of some of the modelled Parts of the casing

## A drawing of a fully assembled system in SolidWorks

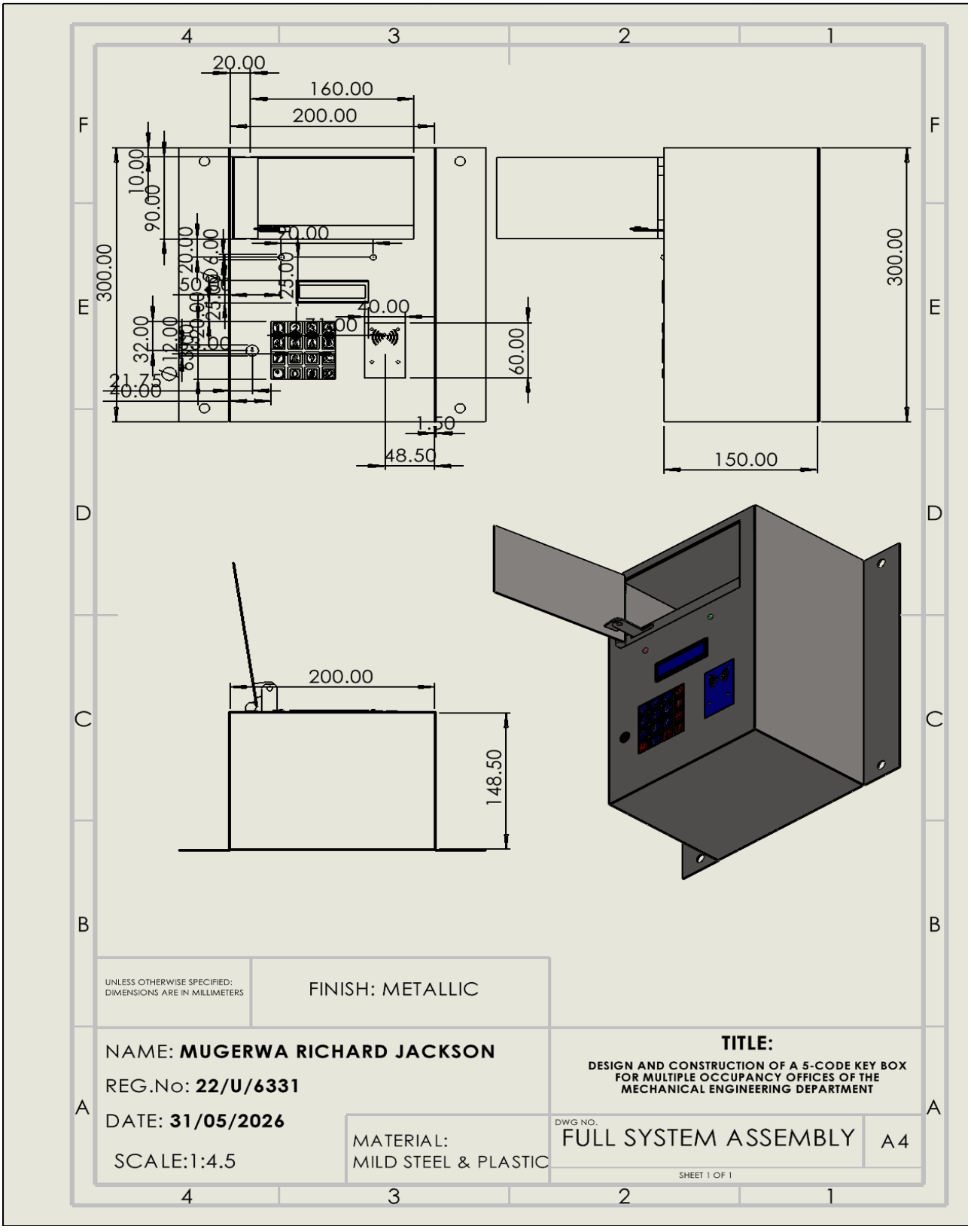


Figure 16: Full system assembly

## The 3D model of the Key access box

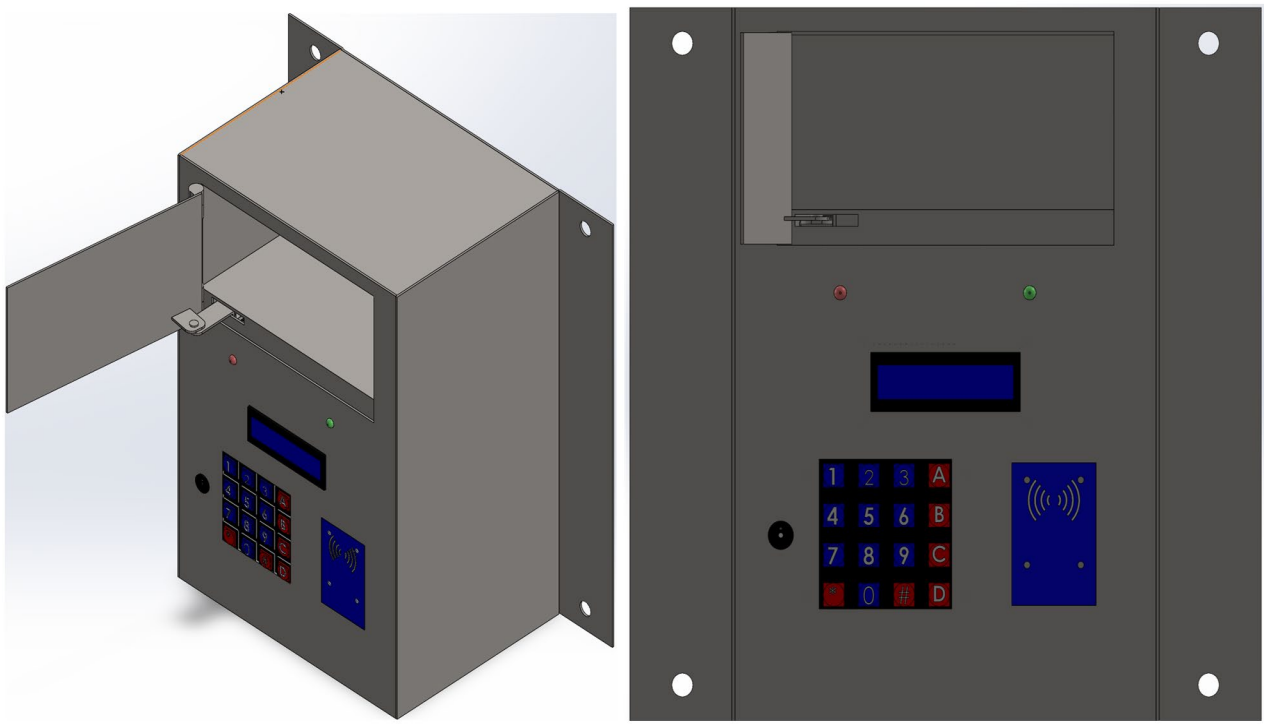


Figure 17: 3D model of assembled key access box

## 4.3 Construction of the system

The construction of this system involved both the electronic section and the exterior casing of the key storage box.

### 4.3.1 The construction of an electric circuit and component

### Key Tag/ Key fob

There was a need to design and construct a receiving Key fob, in this case known as Key Tag, which would communicate with the key box HM-10 module to measure distance away from the box using the RSSI method. The Key was made of an HM-10 module, a coin battery (CR2032) and AT Command Code.

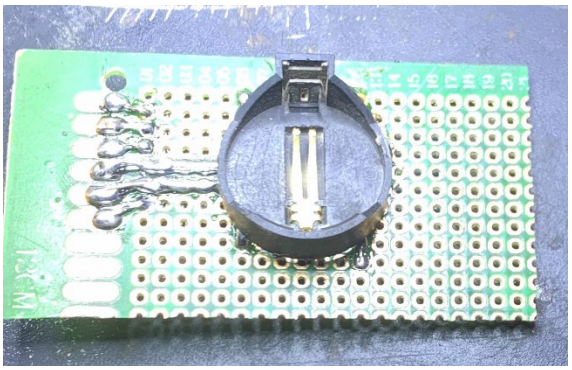

Figure 18: CR2032 Holder soldered on the prototyping board

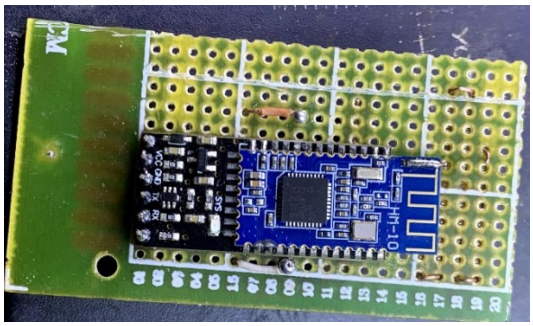

Figure 19: HM-10 module soldered on the same board as the CR2032 holder

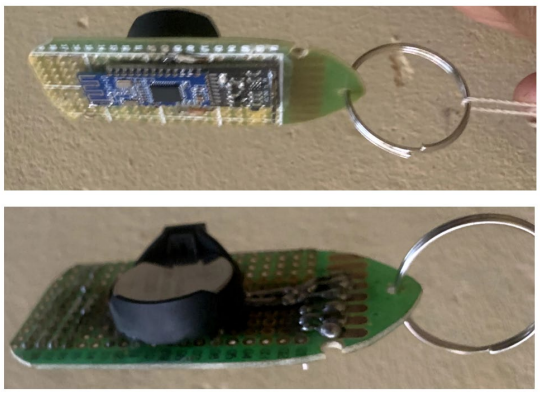

Figure 20: Complete Key Tag Construction

The CR2032 was used to power the HM-10 module; the two were soldered together using a soldering gun and soldering wire on a prototyping board. The VCC and GND of the module were soldered to the positive and negative terminals of the CR2032 holder, respectively.

### The Battery Backup System

The battery backup system was made up of 4 battery cells (ICR18765), a battery holder, and a charging and power management module.

To complete the backup system, was constructed using soldering gun, soldering wire, wires, and hot glue were used. The batteries were wired in parallel connection since the BMS board /charging board required 3.8V from the battery cells that were connected to it. The connection increased the current output, keeping the voltage constant. The red wire represented the positive side, and the black for the negative side.

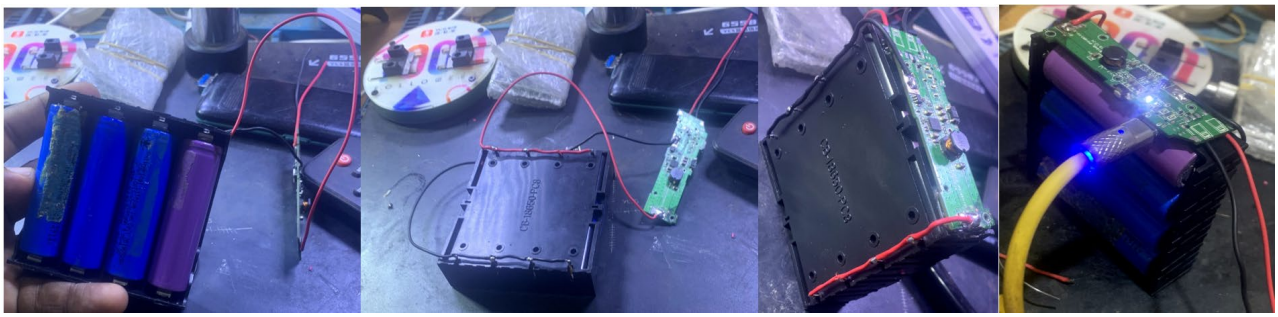

Figure 21: Power management board soldered to the cell holder

The backup could power the system while charging, and its output was 5V and a maximum current of 8A, since each cell had a maximum current of 2A.

### The MCU wiring and construction

The MCU of this system was an Arduino Mega, which was the brain of the whole system. The wiring of the system was done based on the schematic in Figure 10. A prototyping soldering board was used as the basis for the whole system's connections. The connection and soldering were done based on different positions of electronic components in the system, as in Figures 22 and 23.

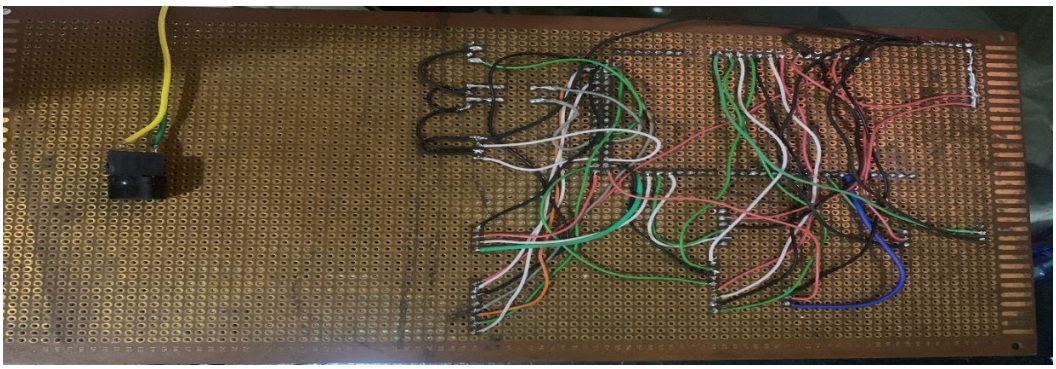

Figure 22: Wiring of the Rear of the Prototyping Board

### Completely functional, wired, and soldered system

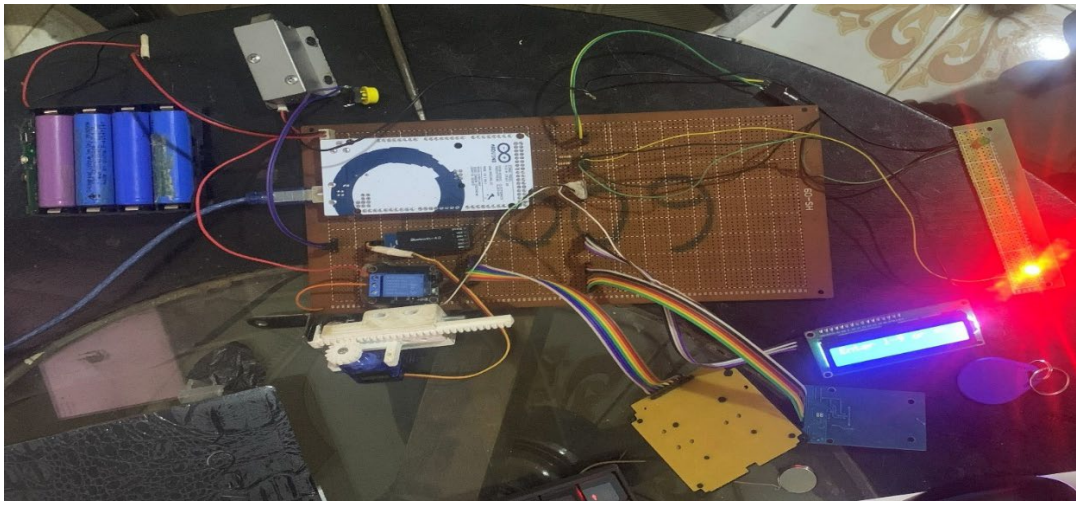

Figure 23: Complete the wire system circuit board

### 4.3.2 The casing and interior construction

The 3D model drawings were used to construct the external and internal parts and plates. The rack and pinion mechanism was 3D printed using ABS plastic, and the casing was welded from mild steel sheet, bolts, and nuts using arc welding.

### 3D printing the rack and pinion mechanism

The mechanism was printed from ABS plastic, and the total time taken to print the whole component was 5 hours. The components were the bracket, rack, and gear.

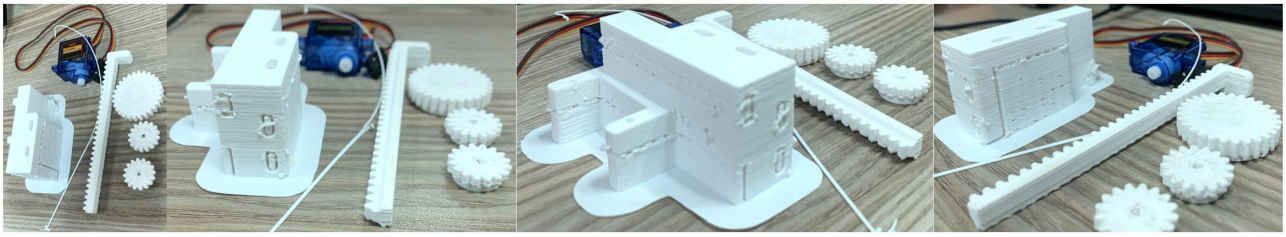

Figure 24: 3D printed rack and pinion components

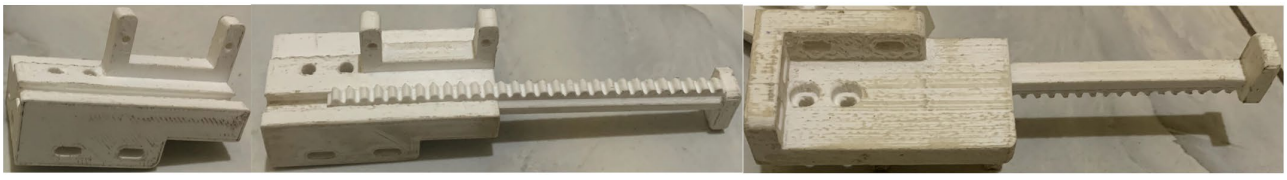

Figure 25: Partially assembled, rack, and bracket

### Casing construction

At first, a trace of the front plate was drawn on a cardboard box, as shown in Figure 26, which was later transferred to the steel plate. The dimensions were measured and traced on the cardboard box using a rule, angle square, measuring tape, pencil, pen, cutter, rubber, calculator, and board saw.

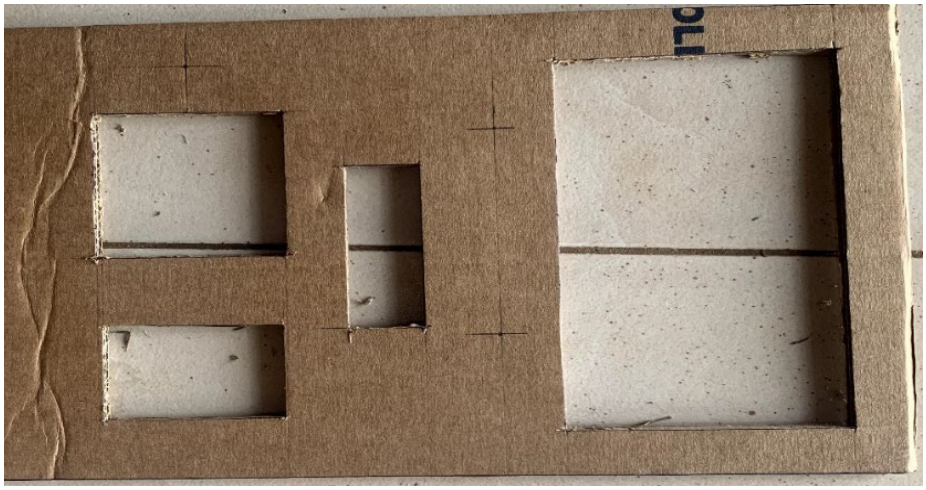

Figure 26: Location of different components cut out of the traces

## Welding of the casing

The front plate was traced using the cardboard cutout, and the dimensions were spot on. The tools that were used during the welding session were a portable arc welding machine, welding electrodes(E6013), angle square, drill bits, drilling machine, marking rod, measuring tape, grinding machine, cutting and grinding discs, engineering vice, a metallic ruler, divining compass, welding shield, chaffing brush, and HB pencil. The steps that were taken to weld the case are shown in Figures 27 to 30.

Figure 27: The tools that were used in construction

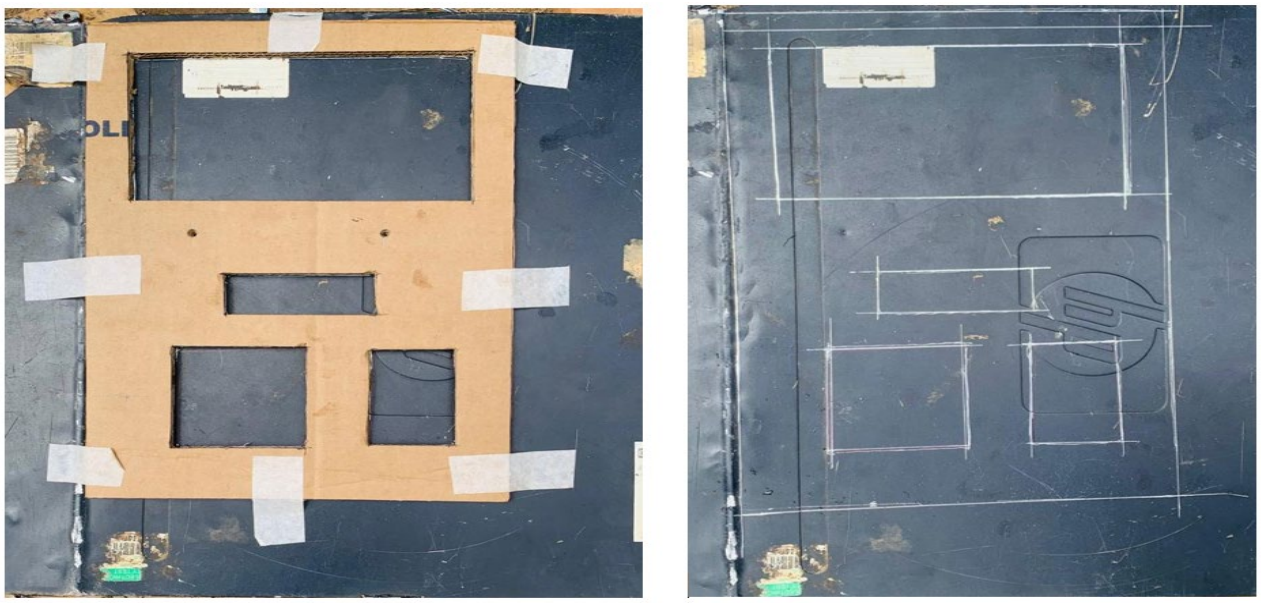

Figure 28: Tracing of the Front plate design and construction

Figure 29: Construction process of the System

The key access box was constructed based on the drawing made, and the appearance of the box was achieved as stated in the design. Figure 31 shows a completely constructed and assembled key box with all the features implemented on it.


Figure 30: The full assembled key access box

## 4.4 Performance and evaluation of the system

### 4.4.1 Performance and Evaluation of Key Tag

The key tag's operational performance was evaluated through a series of tests measuring broadcast range, signal stability, and battery endurance. The tag was placed at increasing distances from the receiver in an indoor office environment with typical obstacles (wooden desks, metal filing cabinets, and concrete walls). RSSI readings were recorded at each distance interval, and the corresponding distance estimates were calculated using the calibrated path loss model. Table 16 in the appendix presents the results of this evaluation.

The results demonstrated that the key tag provided adequate distance estimation accuracy for the 5-metre proximity threshold required by the system. The maximum error observed within the operational range (0.5 to 5 metres) was 4.0%, which was acceptable for the application where precise distance measurement was not required—only whether the tag was within or beyond the 5-metre threshold, as shown in Table 17 in the appendix.

### 4.4.2 Performance and Tests of the Key Access box

When the power switch was pressed, the system booted up, and a welcome message was displayed as shown in Figure 31 above.

### The tests below were performed and evaluated on the system

- Status LEDs operation test

- Password input and system access via PIN or RFID test
- Limit switch test
- Admin Mode test
- Emergency Mode test

### Status LEDs operation test

The red LED represented that the system door was closed, the system was locked due to wrong PIN or RFID, the emergency Button press, and in case of any other failures. The green LED represented that the system door was opened, access was granted and when the system is in emergency mode. The test for the LED was a pass, and it operated as per set parameters in the firmware of the system. The tests performed are shown in Figures 31 to 33.

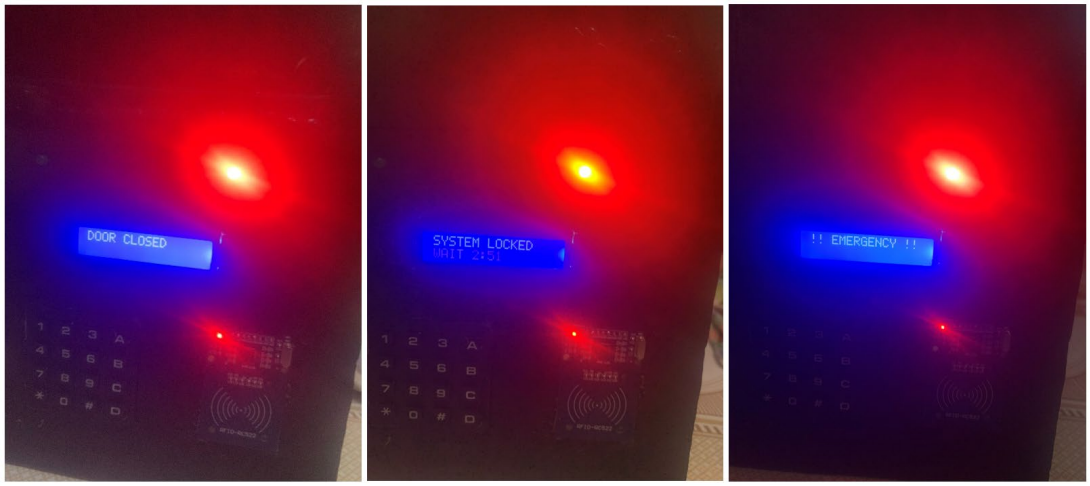


Figure 31: Some of the Red LED operations tested


Figure 32: Some of the green LED Operations tested


Figure 33: Some of the other green LED Operations tested

### Password input and system access via PIN or RFID test

During this test, when a User typed in the PIN, the system could auto-detect the user ID and display it before the user completes the PIN, and when the PIN was complete, he or she was granted access immediately, as shown in Figure 34. In case of an unknown or unregistered user they system displayed User ???, and when a wrong PIN or RFID was input, the system displayed INVALID PIN or RFID and the number of attempts was displayed as shown in figure 35.

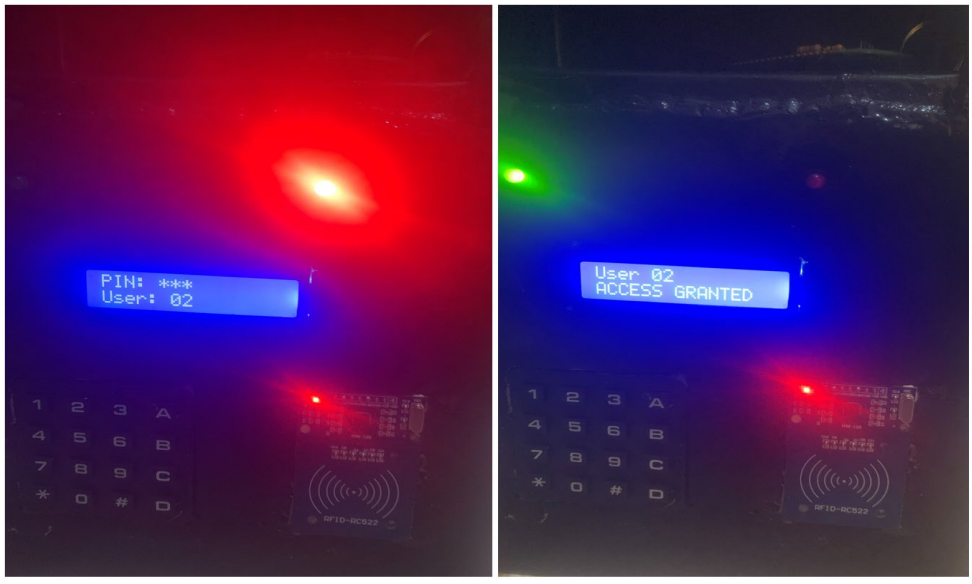


Figure 34: Known User ID auto detection and access granted test

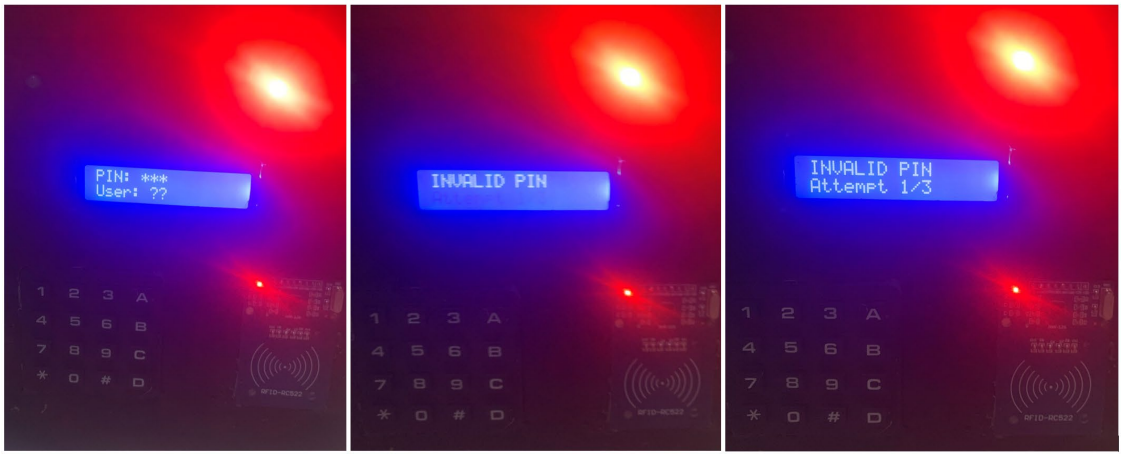


Figure 35: Unknown User ID auto detection and failed attempts display test

When all the 3 attempts were completed using the invalid PIN or RFID, the system was locked for 3 minutes, and a time countdown was displayed showing the remaining time to enable the keypad or RFID module for use again, as shown in Figure 36.


Figure 36: Wrong PIN/ RFID, 3 attempts, and a 3-minute countdown when the system was locked

### Limit switch test

When a correct PIN was input, the status of the system changed from red to green on the LEDs, and the display showed "DOOR OPENING" until the open limit switch was hit to stop the movement of the servo motor. The display showed "DOOR IS OPEN, Closes in 20s" as shown in Figure 37. The time countdown started, and when it hit 0, the door was closed. The display showed "DOOR CLOSING" and when the close limit switch was hit,

the movement of the servo motor stopped, the solenoid was activated and door was locked, and the display showed "DOOR CLOSED" as shown in figure 37

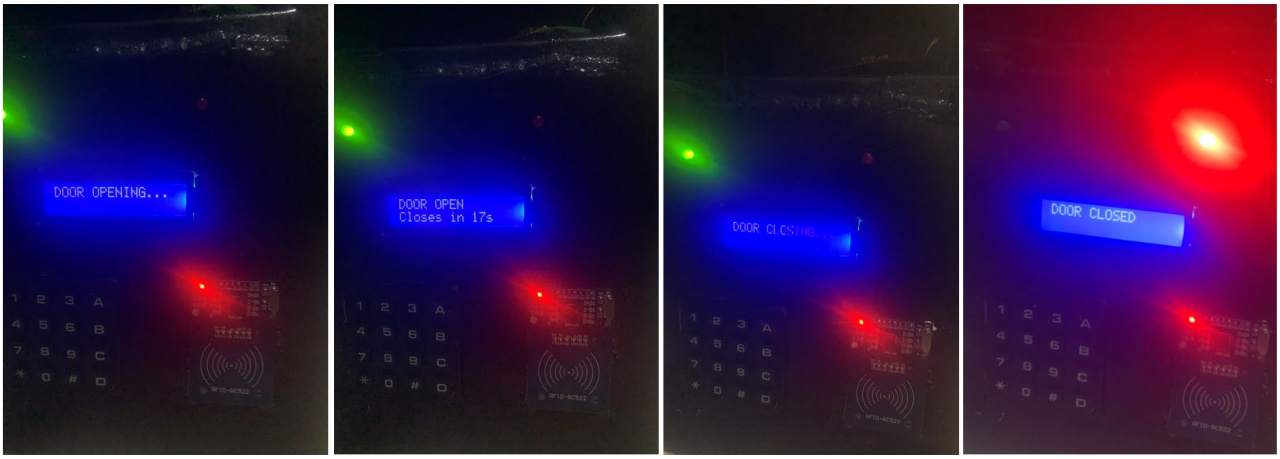


Figure 37: Door Opening and Closing Display

### Admin Mode test

To access the admin mode, the key "A" on the keypad was pressed and a display requesting the Admin PIN. In the administration Mode, a list of different activity that can be performed by the administer were display and these included; add PIN – add new user PIN, DEL PIN – delete added PIN, ADD RFID add new RFID, DEL RFID – delete added RFID, LIST – provide a list of both PINs and RFIDs, CHG PIN – used to change Admin PIN, RESET – used to Reset the System ROM or memory, EEPROM – used to send the storage audit data to serial monitor of the system, PROXIMITY – provided the distance of the key from the system, and EXIT – used to go out of the admin mode. Each was assigned a digital or a symbol on the keypad as shown in Figure 38. To access any item on the list, a key corresponding to the item was pressed, as seen in figures 38 to 40

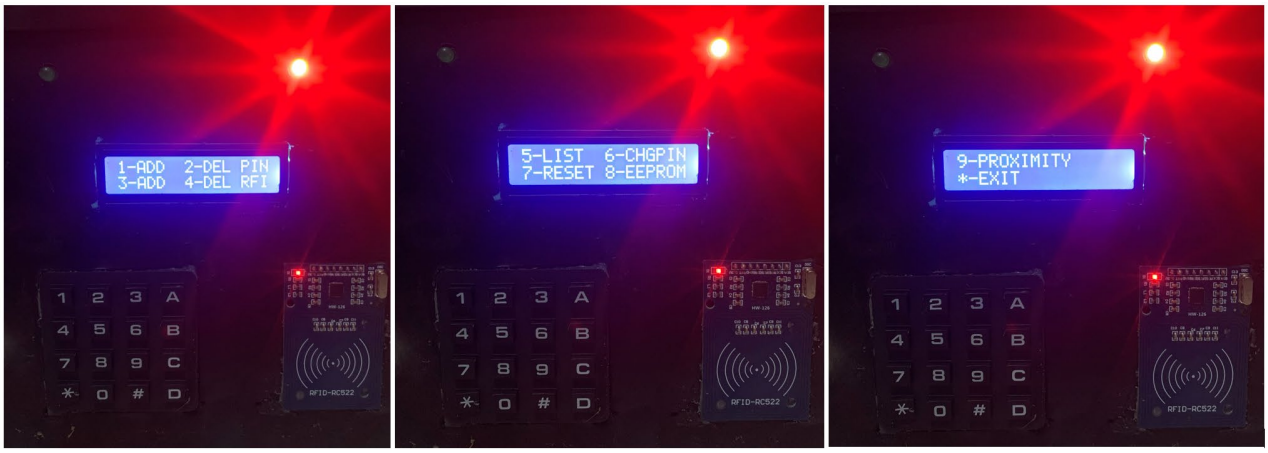


Figure 38: The menu in the Admin Mode

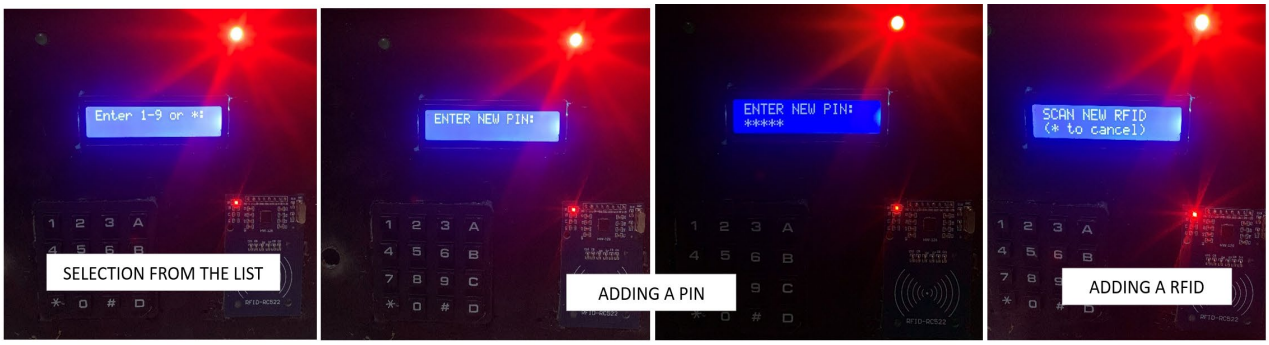


Figure 39: Selection and Adding PIN/ RFID test

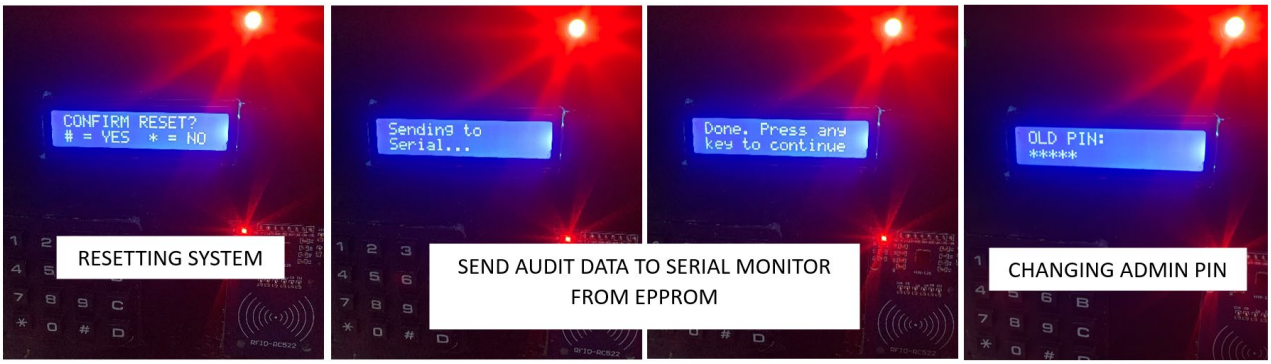


Figure 40: Testing the rest of the system functions

### Emergency Mode test

When the emergency button was pressed, the door was opened. This was usually done during door or any other failures, or low power. The emergency mode was used to reset any errors in the system to the default settings. This button could only be accessed by the administrator of the system, as shown in Figure 41.

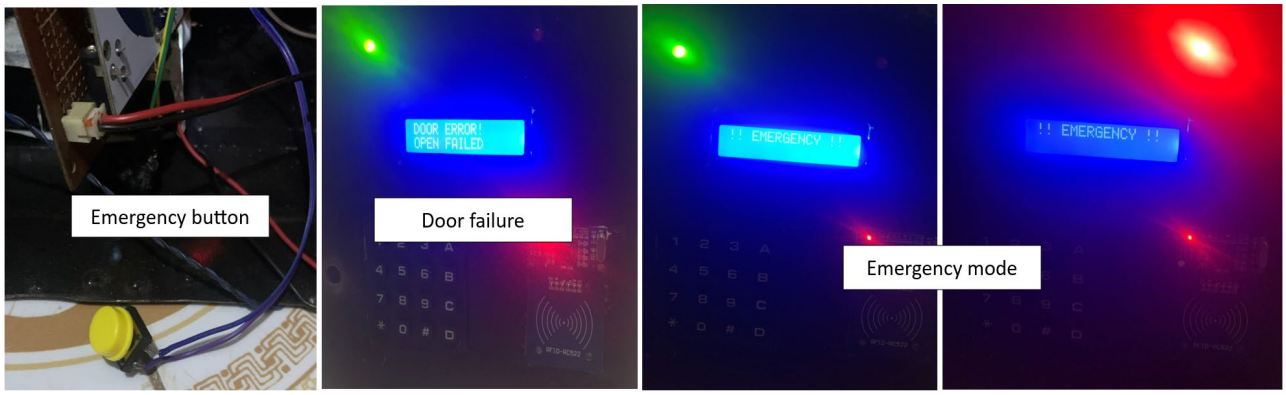


Figure 41: Emergency Button and Mode

### 4.4.3 Evaluation of the Key Access Box

The assembled system was subjected to a series of functional tests covering its start-up, indication, authentication, locking, administrative and safety behaviour. Each test was carried out on the physical prototype and assessed against the behaviour defined in firmware.

The door access control system was successfully validated across all functional areas, meeting every design specification. The system booted correctly with all peripherals initialised and displayed the ready state as expected. Status LEDs accurately reflected all system conditions—red for locked/failed states and green for granted/open states. Both PIN and RFID authentication performed flawlessly, with the partial-PIN matching feature correctly identifying users in real-time and granting immediate access upon credential completion.

The security mechanisms operated effectively, with invalid credentials properly rejected, failed attempts counted, and three consecutive failures triggering a 3-minute lockout with a live countdown. The closed-loop door control worked precisely, with limit switches stopping the servo correctly, the door maintaining its open position for the configured hold period with a visible countdown, and the solenoid automatically re-locking upon closure.

Admin mode provided comprehensive system management, with all menu functions including user and RFID management, PIN changes, system reset, and EEPROM auditing, responding correctly to assigned keys. The emergency override function performed reliably, instantly opening the door even during lockout conditions and restoring the system to its default state upon release. All tests passed with zero failures, confirming the system is stable, secure, and ready for deployment in multiple-occupancy office environments.

# CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

## 5.1 Conclusion

A cost-effective and secure 5-code electronic key box was successfully designed and constructed around an Arduino Mega 2560, integrating a matrix keypad, an RC522 RFID reader, an LCD interface and a servo-driven door secured by a solenoid lock with limit-switch feedback, thereby eliminating the manual key management issues associated with key losses. Multi-user access and physical security were achieved through dual PIN and RFID authentication, with each five-digit PIN offering 100,000 possible combinations (about 16.6 bits of entropy) reinforced by a three-attempt lockout and a three-minute freeze that make brute-force attempts impractical, while accountability was provided through real-time-clock–timestamped event logging. The system was implemented using a non-blocking control architecture that kept it responsive even during door motion, with closed-loop door operation confirmed by limit switches and an explicit failure state, storage of up to ten PINs and ten RFID tags in EEPROM that persisted across power loss, on-device administrative enrolment, and an administrator-controlled emergency override that prioritised user safety. Overall, the project aligns with SDG 9 (Industry, Innovation and Infrastructure) by delivering an affordable, locally engineered security innovation and with SDG 8 (Decent Work and Economic Growth) by enabling safer, more efficient and more accountable work environments, demonstrating how low-cost embedded technology can address a common organisational management problem.

## 5.2 Recommendations for Future Work

Scaling down the prototype case to be better place on the Office door, integration of Wi-Fi/GSM module for real-time SMS alerts and cloud-based audit logs, enhanced Biometrics by integration of fingerprint sensor or facial recognition as a third authentication factor, extended Battery Life by optimizing HM-10 key tag advertising interval such as AT+ADVI9 for 30 days of operation, for full-Scale Deployment, Use a stainless-steel enclosure (IP66, IK10), and commercial-grade PCB for production.

# REFERENCES


Allen, A., Mylonas, A., Vidalis, S., & Gritzalis, D. (2024). Smart homes under siege: Assessing the robustness of physical security against wireless network attacks. *Computers & Security*, *139*, 103687. https://doi.org/10.1016/J.COSE.2023.103687

Auditor General Report. (2024). *REPORT OF THE AUDITOR GENERAL ON MAKERERE UNIVERSITY FOR THE AUDIT YEAR ENDED DECEMBER 2024*.

Avigilon. (2023). *Wiring Fail-Safe And Fail-Secure Lock Hardware - Avigilon OP-R2X-STND Installation Manual [Page 51] | ManualsLib*. https://www.manualslib.com/manual/3930870/Avigilon-Op-R2x-Stnd.html?page=51#manual

Blocki, J., & Zhang, W. (2020). *DALock: Distribution Aware Password Throttling*. http://arxiv.org/abs/2005.09039

Bluetooth SIG. (2010). *Covered Core Package version: 4.0 Current Master TOC Specification Volume 0*. http://www.bluetooth.com

Colby Gutierrez-Kraybill. (2016). *Password Security — MIT Media Lab*. https://www.media.mit.edu/posts/password-security/?utm_source=chatgpt.com

David Odu, A., Chinaza Alice, M., & Odinya, O. J. (2017). Low Cost Removable (Plug-In) Electronic Password-Based Door Lock. *American Journal of Engineering Research (AJER)*, (6), 146–151. www.ajer.org

Electronic key and asset management. (n.d.). *Electronic key and asset management | Traka*. Retrieved 8 October 2025, from https://www.traka.com/global/en

EP4438837. (2024). *Lock with tamper-evident security*.

Farid, Z., Nordin, R., & Ismail, M. (2013). Recent advances in wireless indoor localization techniques and system. *Journal of Computer Networks and Communications*, *2013*. https://doi.org/10.1155/2013/185138

Gomez, C., Oller, J., & Paradells, J. (2012). Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Italian National Conference on Sensors*, *12*(9), 11734–11753. https://doi.org/10.3390/S120911734

IoT SIM Card. (2025). *IoT SIM Card — Global Cellular M2M connectivity | Hologram*. https://www.hologram.io/products/global-iot-sim-card/

Jin. (2016). *HtMaA: Bluetooth Tutorial*. https://fab.cba.mit.edu/classes/863.15/doc/tutorials/programming/bluetooth.html

Laitinen, E., Talvitie, J., & Lohan, E. S. (2015). On the RSS biases in WLAN-based indoor positioning. *2015 IEEE International Conference on Communication Workshop, ICCW 2015*, 797–802. https://doi.org/10.1109/ICCW.2015.7247277

Lawal-solarin, E., Opeola, E., & Olubunmi, G. (2021). Preservation, Security practices, and Use of Students' Theses in University Libraries in South-West. In *Nigeria Part of the Social and Behavioral Sciences Commons LAWAL-SOLARIN*.

Liu, L. S., Ni, J. Q., Zhao, R. Q., Shen, M. X., He, C. L., & Lu, M. Z. (2018). Design and test of a low-power acceleration sensor with Bluetooth Low Energy on ear tags for sow behaviour monitoring. *Biosystems Engineering*, *176*, 162–171. https://doi.org/10.1016/J.BIOSYSTEMSENG.2018.10.011

Luo, H., Niu, X., Li, J., Chen, J., Zou, Y., & Shen, Y. (2018). Research on an Adaptive Algorithm for Indoor Bluetooth Positioning. *Https://Doi.Org/10.1142/S0218001418540149*, *32*(6). https://doi.org/10.1142/S0218001418540149

Nonthaputha, T., Torteanchai, U., Kumngern, M., & Phookwantong, J. (2021). Design of Smart Key Box Using IoT. *International Conference on ICT and Knowledge Engineering*. https://doi.org/10.1109/ICTKE52386.2021.9665693

nVent Hoffman. (2024). *c, EKOM-SW | nVent HOFFMAN*. https://www.nvent.com/ru-ru/hoffman/products/compact-single-wall-version-aluminum-single-and-multiple-doors-ekom-sw-0?f%255B0%255D=attribute_filters%253AMaterial%257CMaterial%257CAluminum%257CAluminum

Park, H., & Hong, J. (2023). BackProx: Secure Backscatter-Assisted Proximity Detection for Passive Keyless Entry and Start Systems. *Sensors*, *23*(4). https://doi.org/10.3390/s23042330

Physical Security Market by Systems. (2025). *Physical Security Market worth $151.50 billion by 2030*. https://www.marketsandmarkets.com/PressReleases/physical-security.asp?utm_source=chatgpt.com

Rahman, Md. M., Ali, Dr. M. S., & Akther, Md. S. (2018). Password Protected Electronic Lock System for Smart Home Security. *International Journal of Engineering Research & Technology*, *7*(4). https://doi.org/10.17577/IJERTV7IS040396

Rana, M., Mamun, Q., & Islam, R. (2023). Enhancing IoT Security: An Innovative Key Management System for Lightweight Block Ciphers. *Sensors (Basel, Switzerland)*, *23*(18), 7678. https://doi.org/10.3390/S23187678

Schneider Electric. (2025). *All Products | Schneider Electric India*. https://www.se.com/in/en/all-products/

Siekkinen, M., Hiienkari, M., Nurminen, J. K., & Nieminen, J. (2012). How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4. *2012 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2012*, 232–237. https://doi.org/10.1109/WCNCW.2012.6215496

Smart Key Storage Systems. (2025). *How Smart Key Storage Systems Are Transforming Security in 2025*. https://www.signifi.com/blog/how-smart-key-storage-systems/?utm_source=chatgpt.com

Video Surveillance. (2021). *IFSEC Global's most read in security: 2021 edition*. https://www.ifsecglobal.com/security/ifsec-globals-most-read-in-security-2021-edition/

Winstead, V., & Xu, C. (2018). Low Energy Bluetooth Inter-Node Communication Schemes via Randomized Reconfiguration. *IEEE International Conference on Electro Information Technology*, *2018-May*, 836–839. https://doi.org/10.1109/EIT.2018.8500301

Zhang, G., Wang, P., Chen, H., & Zhang, L. (2019). Wireless Indoor Localization Using Convolutional Neural Network and Gaussian Process Regression. *Sensors 2019, Vol. 19, Page 2508*, *19*(11), 2508. https://doi.org/10.3390/S19112508

# APPENDICES

## Code Used to Run the System

```
/*
 * ============================================================
 *  DESIGN AND CONSTRUCTION OF A 5-CODE KEY BOX FOR
 *  MULTIPLE OCCUPANCY OFFICES OF THE MECHANICAL ENGINEERING DEPARTMENT
 *  Author     : Mugerwa Richard Jackson
 *  Supervisor : Mr. Edmond Mpagi  /  Dr. Nobert Mukasa
 *  Board      : Arduino Mega 2560 (ATmega2560)
 */

// ──────────────────────────── LIBRARIES ────────────────────────────
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <SPI.h>
#include <MFRC522.h>
#include <RTClib.h>
#include <Servo.h>
#include <EEPROM.h>

// ──────────────────────────── PIN DEFINITIONS ──────────────────────
#define RFID_SS_PIN              53
#define RFID_RST_PIN             49

const byte ROW_PINS[4] = { A0, A1, A2, A3 };
const byte COL_PINS[4] = { 22, A4, A5, 23 };  // [FIX] columns are physically wired
in this order

#define RED_LED_PIN              30
#define GREEN_LED_PIN            31
#define BUZZER_PIN               32
#define DOOR_SERVO_PIN            8
#define DOOR_SOLENOID_PIN        10
#define DOOR_OPEN_LIMIT_PIN      24
#define DOOR_CLOSED_LIMIT_PIN    25
#define EMERGENCY_BUTTON_PIN     A6

// Servo angle constants
#define SERVO_OPEN  180
#define SERVO_STOP   90
#define SERVO_CLOSE   0

// ──────────────────────────── SYSTEM PARAMETERS ────────────────────
const int           PIN_LENGTH             = 5;
const int           MAX_ATTEMPTS           = 3;
```

```
const unsigned long LOCKOUT_TIME            = 180000UL;   // 3 min
const unsigned long DOOR_AUTO_CLOSE_TIME    =  20000UL;   // 40 s
const unsigned long SERVO_TRAVEL_TIMEOUT    =  15000UL;   // max time the servo
keeps rotating to reach a limit switch (tune to your door)
const unsigned long ATTEMPTS_RESET_TIMEOUT  = 300000UL;   // 5 min
const unsigned long ADMIN_SESSION_TIMEOUT   =  60000UL;   // 60 s
const unsigned long KEYPAD_ENTRY_TIMEOUT    =  10000UL;   // 10 s
const unsigned long EMERGENCY_DEBOUNCE_MS   =    100UL;
const unsigned long ADMIN_KEY_WAIT_MS       =  30000UL;   // [BUG-3/4] per-key
timeout in admin helpers

// EEPROM layout  (Mega 2560 = 4096 bytes, addresses 0–4095)
const int PIN_COUNT_ADDR         =   0;
const int RFID_COUNT_ADDR        =   2;
const int ADMIN_PIN_ADDR         =  10;   // 6 bytes  (10-15)
const int USER_PINS_START        =  50;   // up to 10 x 6 = 60 bytes
const int RFID_START             = 200;   // up to 10 x 15 = 150 bytes
const int USER_IDS_START         = 350;   // [FEAT-5] 1 byte per slot x 10 = 10
bytes (350-359)
const int PROXIMITY_ENABLED_ADDR = 500;
const int  FIRSTRUN_FLAG_ADDR    =  16;   // [REPAIR] free byte (16-49 unused) -
one-time admin-PIN repair marker
const byte FIRSTRUN_SIGNATURE    = 0x6B;  // [REPAIR] signature for THIS firmware
build
const int EEPROM_TOTAL           = 4096;  // [BUG-5] total byte count

const int MAX_USERS      = 10;
const int MAX_RFID_TAGS  = 10;

// ──────────────────────── DOOR STATE MACHINE ──────────────────
enum DoorState {
  DOOR_IDLE,
  DOOR_UNLOCKING,
  DOOR_OPENING,
  DOOR_WAITING,
  DOOR_CLOSING,
  DOOR_FAILURE
};

// ──────────────────────── GLOBAL OBJECTS ──────────────────────
LiquidCrystal_I2C lcd(0x27, 16, 2);
RTC_DS3231        rtc;
Servo             doorServo;
MFRC522           rfid(RFID_SS_PIN, RFID_RST_PIN);

char keys[4][4] = {
  {'1','2','3','A'},
```

```
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
Keypad keypad = Keypad(makeKeymap(keys), ROW_PINS, COL_PINS, 4, 4);

// ──────────────────────── GLOBAL VARIABLES ──────────────────
char authorizedPins [MAX_USERS]    [6];
char authorizedRFIDs[MAX_RFID_TAGS][15];
char adminPin[6] = "12345";

// [FEAT-1] User ID tracking — parallel array to authorizedPins
// userIDs[i] holds the display ID (1-based) for slot i.
// IDs are auto-assigned sequentially; never re-used after deletion.
int  userIDs  [MAX_USERS] = {0};
int  nextUserID           = 1;   // next ID to assign

int  pinCount  = 0;
int  rfidCount = 0;

// Attempt tracking
int  attempts      = 0;
int  rfidFailCount = 0;
unsigned long lastActivityTime = 0;

// Keypad entry
char enteredPin[6] = "";
int  enteredPinLen = 0;
unsigned long lastKeypadTime = 0;

// Lockout
bool          systemLocked             = false;
unsigned long lockoutStartTime         = 0;
unsigned long lastLockoutDisplayUpdate = 0;
bool          lockoutLcdSet            = false;
// [BUG-6] non-blocking "TRY AGAIN" display after lockout expires
bool          lockoutJustCleared       = false;
unsigned long lockoutClearedTime       = 0;

// Door state machine
DoorState     doorState          = DOOR_IDLE;
unsigned long doorStateStartTime = 0;
bool          doorFailure        = false;

// Emergency
bool          emergencyActive       = false;
unsigned long lastEmergencyDebounce = 0;
```

```
bool          lastEmergencyState    = HIGH;

// RTC
bool rtcAvailable = false;

// ──────────────────────────── FUNCTION PROTOTYPES ──────────────────
void defaultState();
void emergencyMode();
void authenticateRFID(const char* uid);
void authenticatePIN (const char* pin);
void handleKeypadInput(char key);
int  findPartialMatch(const char* partial, int len); // [FEAT-6]
void handleLockout();
void updateDoorState();
void servoRun(int dir);   // [SERVO] continuous-rotation: attach + run in a
direction
void servoStop();         // [SERVO] continuous-rotation: detach = a real stop (no
creep)
void adminMode();
void adminLogin();   // [ADMIN] 'A' key -> prompt for admin PIN, then verify
char adminKeyWait(unsigned long* sessionStart);  // [BUG-3/4]
void addUserPIN();
void deleteUserPIN();
void addRFID();
void deleteRFID();
void listUsers();
void changeAdminPIN();
void resetSystem();
void viewEEPROM();
void debugReadEEPROM();
void logEvent(const char* event);
void saveAuthorizedPins();
void loadAuthorizedPins();
void saveUserIDs();        // [FEAT-5]
void loadUserIDs();        // [FEAT-5]
void saveAuthorizedRFIDs();
void loadAuthorizedRFIDs();
void saveAdminPin();
void loadAdminPin();
void clearEEPROM();
void getRFIDString(char* buf, int maxLen);
void beepShort();
void beepLong();
void beepAlarm();
void beepWarning();

// ═══════════════════════════════════════════════════════════════
```

```
//                          SETUP
// ═══════════════════════════════════════════════════════════════
void setup() {
  Serial.begin(9600);
  Wire.begin();

  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.print("System Starting");

  rtcAvailable = rtc.begin();
  if (!rtcAvailable) {
    lcd.clear();
    lcd.print("RTC WARNING");
    lcd.setCursor(0, 1);
    lcd.print("No RTC detected");
    delay(2000);
  }

  SPI.begin();
  rfid.PCD_Init();

  pinMode(RED_LED_PIN,           OUTPUT);
  pinMode(GREEN_LED_PIN,         OUTPUT);
  pinMode(BUZZER_PIN,            OUTPUT);
  pinMode(DOOR_SOLENOID_PIN,     OUTPUT);
  pinMode(DOOR_OPEN_LIMIT_PIN,   INPUT_PULLUP);
  pinMode(DOOR_CLOSED_LIMIT_PIN, INPUT_PULLUP);
  pinMode(EMERGENCY_BUTTON_PIN,  INPUT_PULLUP);

  digitalWrite(DOOR_SOLENOID_PIN, LOW);   // solenoid locked
  servoStop();                            // CR servo idle (detached) at boot

  // [BUG-12] Set explicit debounce time — prevents double-press issue
  // where the default 10ms debounce matched the loop() delay exactly.
  keypad.setDebounceTime(50);

  loadAuthorizedPins();
  loadAuthorizedRFIDs();
  loadAdminPin();

  // [REPAIR] One-time admin-PIN repair. If this firmware build has never run
  // on this board (marker byte absent), the EEPROM may hold a stale/forgotten
  // admin PIN that locks you out. Force it back to the default "12345" ONCE,
  // then write the marker so this never runs again - any admin PIN you set
  // later via the admin menu will persist normally. (After a full EEPROM
```

```
  // clear the marker is wiped too, so admin correctly returns to default.)
  if (EEPROM.read(FIRSTRUN_FLAG_ADDR) != FIRSTRUN_SIGNATURE) {
    strcpy(adminPin, "12345");
    saveAdminPin();
    EEPROM.update(FIRSTRUN_FLAG_ADDR, FIRSTRUN_SIGNATURE);
    lcd.clear();
    lcd.print("ADMIN PIN SET");
    lcd.setCursor(0, 1);
    lcd.print("DEFAULT = 12345");
    logEvent("First run: admin PIN initialised to default");
    delay(2500);
  }

  // [RECOVERY] Hold the EMERGENCY button while powering on to force the
  // admin PIN back to the default "12345" (use if it was changed/forgotten).
  if (digitalRead(EMERGENCY_BUTTON_PIN) == LOW) {
    strcpy(adminPin, "12345");
    saveAdminPin();
    lcd.clear();
    lcd.print("ADMIN PIN RESET");
    lcd.setCursor(0, 1);
    lcd.print("DEFAULT = 12345");
    logEvent("Admin PIN reset to default via boot button");
    delay(2500);
  }

  loadUserIDs();     // [FEAT-5] load stored user IDs and nextUserID

  delay(1000);
  defaultState();

  // [BUG-1] corrected version string from "v5.1" to "v5.3"
  Serial.println("+---------------------------------------+");
  Serial.println("|  KEY BOX SYSTEM v6.0  (Mega 2560)    |");
  Serial.println("+---------------------------------------+");
  Serial.print("| User PINs loaded   : ");
  Serial.println(pinCount);
  Serial.print("| RFID tags loaded   : ");
  Serial.println(rfidCount);
}

// ============================================================
//                          MAIN LOOP
// ============================================================
void loop() {

  // ── 1. Debounced emergency button (checked first, always) ──
```

```
  bool curEmergency = digitalRead(EMERGENCY_BUTTON_PIN);
  if (curEmergency == LOW && lastEmergencyState == HIGH &&
      (millis() - lastEmergencyDebounce > EMERGENCY_DEBOUNCE_MS) &&
      !emergencyActive) {
    lastEmergencyDebounce = millis();
    if (systemLocked) {
      systemLocked  = false;
      lockoutLcdSet = false;
    }
    emergencyMode();
    return;
  }
  lastEmergencyState = curEmergency;

  // ── 2. Non-blocking lockout display ──
  if (systemLocked) {
    handleLockout();
    delay(10);
    return;
  }

  // [BUG-6] Non-blocking "TRY AGAIN" display after lockout clears
  if (lockoutJustCleared) {
    if (millis() - lockoutClearedTime >= 2000UL) {
      lockoutJustCleared = false;
      defaultState();
    }
    delay(10);
    return;
  }

  // ── 3. Reset attempt counters after inactivity ──
  if ((attempts > 0 || rfidFailCount > 0) &&
      (millis() - lastActivityTime > ATTEMPTS_RESET_TIMEOUT)) {
    attempts         = 0;
    rfidFailCount    = 0;
    lastActivityTime = millis();
    Serial.println("Attempt counters reset (inactivity)");
  }

  // ── 4. Non-blocking door state machine ──
  if (doorState != DOOR_IDLE) {
    updateDoorState();
  }

  // ── 6. RFID scan (door idle + not locked) ──
  if (doorState == DOOR_IDLE) {
```

```
    if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
      char uid[15];
      getRFIDString(uid, 15);
      authenticateRFID(uid);
      rfid.PICC_HaltA();
      rfid.PCD_StopCrypto1();
    }
  }

  // ── 7. Keypad — only when door is IDLE ──
  if (doorState == DOOR_IDLE) {
    char key = keypad.getKey();
    if (key) {
      handleKeypadInput(key);
      lastKeypadTime = millis();
    } else if (enteredPinLen > 0 &&
               (millis() - lastKeypadTime > KEYPAD_ENTRY_TIMEOUT)) {
      enteredPin[0] = '\0';
      enteredPinLen = 0;
      lcd.clear();
      lcd.print("ENTRY TIMEOUT");
      beepWarning();
      delay(1500);
      defaultState();
    }
  }

  // [BUG-12] Reduced from delay(10) — the original value matched the
  // Keypad library debounce window exactly, causing missed keypresses.
  delay(1);
}

// ═══════════════════════════════════════════════════════════
//                       DEFAULT STATE
// ═══════════════════════════════════════════════════════════
void defaultState() {
  if (doorState != DOOR_IDLE) return;

  if (systemLocked) {
    lcd.clear();
    lcd.print("SYSTEM LOCKED");
    return;
  }

  digitalWrite(DOOR_SOLENOID_PIN, LOW);
  servoStop();
  digitalWrite(RED_LED_PIN,   HIGH);
```

```
  digitalWrite(GREEN_LED_PIN, LOW);
  lcd.clear();
  lcd.print("WELCOME!");
  lcd.setCursor(0, 1);
  lcd.print("PIN OR RFID PLS");
}

// ============================================================
//              NON-BLOCKING LOCKOUT HANDLER
// ============================================================
void handleLockout() {
  unsigned long elapsed = millis() - lockoutStartTime;

  // First call after lockout starts: paint top line once
  if (!lockoutLcdSet) {
    lcd.clear();
    lcd.print("SYSTEM LOCKED");
    digitalWrite(RED_LED_PIN,   HIGH);
    digitalWrite(GREEN_LED_PIN, LOW);
    lockoutLcdSet            = true;
    lastLockoutDisplayUpdate = millis();
  }

  // Lockout expired — [BUG-6] non-blocking transition
  if (elapsed >= LOCKOUT_TIME) {
    systemLocked       = false;
    attempts           = 0;
    rfidFailCount      = 0;
    lockoutLcdSet      = false;
    // Show "TRY AGAIN" without a blocking delay
    lcd.clear();
    lcd.print("TRY AGAIN");
    lockoutJustCleared  = true;
    lockoutClearedTime  = millis();
    return;
  }

  // Update countdown every second (bottom row only)
  if (millis() - lastLockoutDisplayUpdate >= 1000) {
    lastLockoutDisplayUpdate = millis();
    unsigned long remaining = (LOCKOUT_TIME - elapsed) / 1000UL;
    lcd.setCursor(0, 1);
    lcd.print("WAIT ");
    lcd.print(remaining / 60);
    lcd.print(":");
    if (remaining % 60 < 10) lcd.print("0");
    lcd.print(remaining % 60);
```

```
    lcd.print("   ");
  }
}

// ================================================================
//              NON-BLOCKING DOOR STATE MACHINE
// ================================================================
// ----------------------------------------------------------------
//   Continuous-rotation servo helpers
//   SERVO_STOP (90 deg) is NOT a true neutral for a CR servo, so writing it
//   makes the servo keep creeping/reversing instead of stopping. We therefore
//   STOP by detaching (no pulses = no motion) and only re-attach when moving.
// ----------------------------------------------------------------
void servoRun(int dir) {
  if (!doorServo.attached()) doorServo.attach(DOOR_SERVO_PIN);
  doorServo.write(dir);
}

void servoStop() {
  doorServo.detach();
}

void updateDoorState() {
  unsigned long elapsed = millis() - doorStateStartTime;

  switch (doorState) {

    case DOOR_UNLOCKING:
      if (elapsed >= 100) {
        servoRun(SERVO_OPEN);
        doorState          = DOOR_OPENING;
        doorStateStartTime = millis();
        lcd.clear();
        lcd.print("DOOR OPENING...");
      }
      break;

    case DOOR_OPENING:
      if (digitalRead(DOOR_OPEN_LIMIT_PIN) == LOW) {
        servoStop();
        doorState          = DOOR_WAITING;
        doorStateStartTime = millis();
        lcd.clear();
        lcd.print("DOOR OPEN");
        logEvent("Door opened");
        doorFailure = false;
      } else if (elapsed >= SERVO_TRAVEL_TIMEOUT) {
```

```
      servoStop();
      doorState          = DOOR_FAILURE;
      doorStateStartTime = millis();
      doorFailure        = true;
      lcd.clear();
      lcd.print("DOOR ERROR!");
      lcd.setCursor(0, 1);
      lcd.print("OPEN FAILED");
      beepAlarm();
      logEvent("FAILURE: door did not open (limit sw)");
    }
    break;

  case DOOR_WAITING: {
    // Live countdown of the time remaining before the door auto-closes.
    // Updated once per second on line 2 (line 1 still shows "DOOR OPEN").
    unsigned long remainingMs  = (elapsed < DOOR_AUTO_CLOSE_TIME)
                                 ? (DOOR_AUTO_CLOSE_TIME - elapsed) : 0;
    unsigned int  remainingSec = (unsigned int)((remainingMs + 999UL) / 1000UL);
    static unsigned int lastShownSec = 0xFFFF;
    if (remainingSec != lastShownSec) {
      lastShownSec = remainingSec;
      char line[17];
      snprintf(line, sizeof(line), "Closes in %2us  ", remainingSec);
      lcd.setCursor(0, 1);
      lcd.print(line);
    }

    if (elapsed >= DOOR_AUTO_CLOSE_TIME) {
      servoRun(SERVO_CLOSE);
      doorState          = DOOR_CLOSING;
      doorStateStartTime = millis();
      lcd.clear();
      lcd.print("DOOR CLOSING...");
    }
    break;
  }

  case DOOR_CLOSING:
    if (digitalRead(DOOR_CLOSED_LIMIT_PIN) == LOW) {
      servoStop();
      digitalWrite(DOOR_SOLENOID_PIN, LOW);
      doorState = DOOR_IDLE;
      lcd.clear();
      lcd.print("DOOR CLOSED");
      logEvent("Door closed and locked");
      delay(1000);
```

```
        defaultState();
      } else if (elapsed >= SERVO_TRAVEL_TIMEOUT) {
        servoStop();
        doorState          = DOOR_FAILURE;
        doorStateStartTime = millis();
        doorFailure        = true;
        lcd.clear();
        lcd.print("DOOR ERROR!");
        lcd.setCursor(0, 1);
        lcd.print("CLOSE FAILED");
        beepAlarm();
        logEvent("FAILURE: door did not close (limit sw)");
      }
      break;

    case DOOR_FAILURE:
      // Hold failure display for 10 s then return to idle
      if (elapsed >= 10000UL) {
        doorState = DOOR_IDLE;
        defaultState();
      }
      break;

    default:
      break;
  }
}

// ============================================================
//              RFID AUTHENTICATION
// ============================================================
void authenticateRFID(const char* uid) {
  if (systemLocked) return;

  bool authorized = false;
  int  matchedIdx = -1;
  for (int i = 0; i < rfidCount; i++) {
    if (strcmp(uid, authorizedRFIDs[i]) == 0) {
      authorized = true;
      matchedIdx = i;
      break;
    }
  }

  if (authorized) {
    // [FEAT-11] Proximity removed — grant access on valid RFID only
    char idLabel[12];
```

```
    snprintf(idLabel, sizeof(idLabel), "RFID %02d", matchedIdx + 1);

    lcd.clear();
    lcd.print(idLabel);
    lcd.setCursor(0, 1);
    lcd.print("ACCESS GRANTED");
    beepShort();

    char logMsg[40];
    snprintf(logMsg, sizeof(logMsg), "RFID access granted [%s]", idLabel);
    logEvent(logMsg);

    digitalWrite(GREEN_LED_PIN, HIGH);
    digitalWrite(RED_LED_PIN,   LOW);
    delay(1500);

    digitalWrite(DOOR_SOLENOID_PIN, HIGH);
    doorState          = DOOR_UNLOCKING;
    doorStateStartTime = millis();
    attempts           = 0;
    rfidFailCount      = 0;
    lastActivityTime   = millis();

  } else {
    // [FEAT-7] Unknown RFID — show specific INVALID message and
    // print the scanned UID to Serial so admin can identify the card
    lcd.clear();
    lcd.print("INVALID RFID");
    lcd.setCursor(0, 1);
    lcd.print("NOT REGISTERED");
    beepLong();

    char logMsg[50];
    snprintf(logMsg, sizeof(logMsg), "INVALID RFID scan — UID: %s", uid);
    logEvent(logMsg);

    digitalWrite(RED_LED_PIN,   HIGH);
    digitalWrite(GREEN_LED_PIN, LOW);
    rfidFailCount++;
    lastActivityTime = millis();

    if (rfidFailCount >= MAX_ATTEMPTS) {
      systemLocked             = true;
      lockoutStartTime         = millis();
      lockoutLcdSet            = false;
      lastLockoutDisplayUpdate = 0;
      rfidFailCount            = 0;
```

```
      lcd.clear();
      lcd.print("TOO MANY TRIES");
      lcd.setCursor(0, 1);
      lcd.print("LOCKED 3 MIN");
      beepAlarm();
      logEvent("System locked (repeated invalid RFID)");
      delay(5000);
    } else {
      delay(3000);   // [FEAT-9] extended from 2 s to 3 s
      defaultState();
    }
  }
}

// ============================================================
//              PIN AUTHENTICATION
// ============================================================
void authenticatePIN(const char* pin) {
  if (systemLocked) return;

  // [ADMIN] Admin mode is now reached only via the 'A' key (see adminLogin()).
  // Entering the admin PIN through normal PIN entry is intentionally NOT
  // treated as admin, so it cannot auto-trigger admin mode during daily use.

  // User PINs
  bool authorized = false;
  int  matchedIdx = -1;
  for (int i = 0; i < pinCount; i++) {
    if (strcmp(pin, authorizedPins[i]) == 0) {
      authorized = true;
      matchedIdx = i;
      break;
    }
  }

  if (authorized) {
    // [FEAT-11] Proximity removed — grant access on valid PIN only
    char idLabel[12];
    snprintf(idLabel, sizeof(idLabel), "User %02d", userIDs[matchedIdx]);

    lcd.clear();
    lcd.print(idLabel);
    lcd.setCursor(0, 1);
    lcd.print("ACCESS GRANTED");
    beepShort();

    char logMsg[40];
```

```
    snprintf(logMsg, sizeof(logMsg), "PIN access granted [%s]", idLabel);
    logEvent(logMsg);

    digitalWrite(GREEN_LED_PIN, HIGH);
    digitalWrite(RED_LED_PIN,   LOW);
    delay(1500);

    digitalWrite(DOOR_SOLENOID_PIN, HIGH);
    doorState          = DOOR_UNLOCKING;
    doorStateStartTime = millis();
    attempts           = 0;
    rfidFailCount      = 0;
    lastActivityTime   = millis();

  } else {
    // [FEAT-8] Wrong/unregistered PIN — explicit INVALID PIN message
    lcd.clear();
    lcd.print("INVALID PIN");
    beepLong();
    logEvent("Invalid PIN attempt");
    digitalWrite(RED_LED_PIN,   HIGH);
    digitalWrite(GREEN_LED_PIN, LOW);
    attempts++;
    lastActivityTime = millis();

    if (attempts >= MAX_ATTEMPTS) {
      systemLocked             = true;
      lockoutStartTime         = millis();
      lockoutLcdSet            = false;
      lastLockoutDisplayUpdate = 0;
      lcd.clear();
      lcd.print("TOO MANY TRIES");
      lcd.setCursor(0, 1);
      lcd.print("LOCKED 3 MIN");
      beepAlarm();
      logEvent("System locked (too many invalid PINs)");
      delay(5000);
    } else {
      lcd.setCursor(0, 1);
      lcd.print("Attempt ");
      lcd.print(attempts);
      lcd.print("/");
      lcd.print(MAX_ATTEMPTS);
      delay(3000);   // [FEAT-9] extended from 2 s to 3 s
      defaultState();
    }
  }
```

```
}

// ═══════════════════════════════════════════════════════════════
//  [FEAT-6] PARTIAL PIN MATCH HELPER
//  Returns the userIDs[] value of the ONLY user whose PIN starts
//  with 'partial' (length 'len'). Returns:
//    > 0   — exactly one match found, value is the user ID
//    0     — multiple matches (ambiguous — don't reveal anything)
//   -1     — no matches at all
//  The admin PIN is deliberately excluded so admin identity is
//  never hinted at during normal user entry.
// ═══════════════════════════════════════════════════════════════
int findPartialMatch(const char* partial, int len) {
  int matchCount = 0;
  int matchedID  = -1;
  for (int i = 0; i < pinCount; i++) {
    if (strncmp(authorizedPins[i], partial, len) == 0) {
      matchCount++;
      matchedID = userIDs[i];
    }
  }
  if (matchCount == 1) return matchedID;
  if (matchCount  > 1) return 0;
  return -1;
}

// ═══════════════════════════════════════════════════════════════
//  [FIX-11] updatePinDisplay — write only what changed on LCD
//  Called after every digit. No lcd.clear() — only targeted
//  setCursor + print so the I2C bus is not blocked.
//  Line 0: "PIN: *****      "
//  Line 1: "User: 03        "  or blank or "User: ??"
// ═══════════════════════════════════════════════════════════════
void updatePinDisplay() {
  // ── Line 0: overwrite asterisk field only ──────────────
  // "PIN: " is 5 chars, then up to PIN_LENGTH asterisks,
  // then spaces to clear any leftover character.
  lcd.setCursor(5, 0);
  for (int i = 0; i < PIN_LENGTH; i++) {
    if (i < enteredPinLen) lcd.print('*');
    else                   lcd.print(' ');
  }

  // ── Line 1: user ID hint ───────────────────────────────
  lcd.setCursor(0, 1);
  if (enteredPinLen == 0) {
    lcd.print("               ");   // blank when no digits yet
```

```
    return;
  }
  int result = findPartialMatch(enteredPin, enteredPinLen);
  if (result > 0) {
    lcd.print("User: ");
    if (result < 10) lcd.print("0");
    lcd.print(result);
    lcd.print("         ");
  } else if (result == 0) {
    lcd.print("                ");   // ambiguous — stay silent
  } else {
    lcd.print("User: ??        "); // no match — early wrong-PIN hint
  }
}

// ============================================================
//              KEYPAD INPUT HANDLER  [FIX-11 updated]
// ============================================================
void handleKeypadInput(char key) {
  if (key == '#') {
    if (enteredPinLen == PIN_LENGTH) {
      // [BUG-14] Copy PIN and reset buffer BEFORE calling
      // authenticatePIN so re-press of # never re-submits
      // a stale buffer after a blocking admin/auth session.
      char pinCopy[6];
      strcpy(pinCopy, enteredPin);
      enteredPin[0] = '\0';
      enteredPinLen = 0;
      authenticatePIN(pinCopy);
    } else if (enteredPinLen == 0) {
      // With auto-submit, '#' is no longer needed. If a user presses it out
      // of habit after the PIN has already been submitted (buffer empty),
      // just refresh the idle screen instead of scolding "WRONG LENGTH".
      defaultState();
    } else {
      enteredPin[0] = '\0';
      enteredPinLen = 0;
      lcd.clear();
      lcd.print("WRONG LENGTH");
      lcd.setCursor(0, 1);
      lcd.print("Need 5 digits");
      beepLong();
      logEvent("PIN rejected (wrong length)");
      delay(2000);
      defaultState();
    }
```

```
  } else if (key == '*') {
    enteredPin[0] = '\0';
    enteredPinLen = 0;
    lcd.clear();
    lcd.print("CLEARED");
    delay(1000);
    defaultState();

  } else if (key >= '0' && key <= '9') {
    if (enteredPinLen < PIN_LENGTH) {

      // [FIX-11] Paint header ONCE on the very first digit
      // so "PIN: " exists on line 0 before updatePinDisplay
      // tries to write asterisks at position 5.
      if (enteredPinLen == 0) {
        lcd.clear();
        lcd.print("PIN: ");
      }

      enteredPin[enteredPinLen]     = key;
      enteredPin[enteredPinLen + 1] = '\0';
      enteredPinLen++;

      // [AUTO-SUBMIT] No '#' required: the moment the 5th digit is entered,
      // show the full mask briefly, then authenticate. Uses the same
      // copy-then-reset order as the '#' path (BUG-14) so a stale buffer
      // can never be re-submitted.
      if (enteredPinLen == PIN_LENGTH) {
        updatePinDisplay();
        delay(150);
        char pinCopy[6];
        strcpy(pinCopy, enteredPin);
        enteredPin[0] = '\0';
        enteredPinLen = 0;
        authenticatePIN(pinCopy);
        return;
      }

      // [FIX-11] Only update changed characters — no full clear
      updatePinDisplay();
    }

  } else {
    // Keys A, B, C, D — give clear feedback then restore display
    if (key == 'A') {                 // [ADMIN] 'A' opens admin login
      enteredPin[0] = '\0';
      enteredPinLen = 0;
```

```
      adminLogin();
      return;
    }
    lcd.clear();
    lcd.print("USE 0-9 # *");
    beepWarning();
    delay(1000);
    // Restore PIN entry screen without a redundant clear
    if (enteredPinLen == 0) {
      defaultState();
    } else {
      lcd.clear();
      lcd.print("PIN: ");
      updatePinDisplay();
    }
  }
}

// ============================================================
//              EMERGENCY MODE
//  Intentionally blocking (~11 s) — physical door safety
// ============================================================
void emergencyMode() {
  emergencyActive = true;
  lcd.clear();
  lcd.print("!! EMERGENCY !!");
  logEvent("Emergency override activated");
  beepAlarm();

  digitalWrite(DOOR_SOLENOID_PIN, HIGH);
  servoRun(SERVO_OPEN);
  digitalWrite(GREEN_LED_PIN, HIGH);
  digitalWrite(RED_LED_PIN,   LOW);

  // Hold open 10 s
  unsigned long start = millis();
  while (millis() - start < 10000UL) {
    delay(50);
  }

  // [BUG-2] Drive servo closed and wait for limit switch (up to 5 s)
  //         before de-energising the solenoid, so the bolt actually
  //         engages.  If the limit is never reached, log the failure.
  servoRun(SERVO_CLOSE);
  unsigned long closeStart = millis();
  bool closedOk = false;
  while (millis() - closeStart < SERVO_TRAVEL_TIMEOUT) {
```

```
    if (digitalRead(DOOR_CLOSED_LIMIT_PIN) == LOW) {
      closedOk = true;
      break;
    }
    delay(50);
  }
  servoStop();

  if (closedOk) {
    digitalWrite(DOOR_SOLENOID_PIN, LOW);
    logEvent("Emergency mode ended — door confirmed closed");
  } else {
    // Solenoid stays HIGH; door physically did not close — alert
    logEvent("FAILURE: door did not close after emergency (limit sw)");
    lcd.clear();
    lcd.print("CLOSE FAILED!");
    lcd.setCursor(0, 1);
    lcd.print("Check door!");
    beepAlarm();
    delay(3000);
    // De-energise anyway to avoid coil burn-out; door may need manual check
    digitalWrite(DOOR_SOLENOID_PIN, LOW);
  }

  noTone(BUZZER_PIN);
  digitalWrite(RED_LED_PIN,   HIGH);
  digitalWrite(GREEN_LED_PIN, LOW);

  lcd.clear();
  lcd.print(closedOk ? "DOOR CLOSED" : "CHECK DOOR");
  delay(2000);

  emergencyActive = false;
  defaultState();
}

// ═══════════════════════════════════════════════════════════
//  [BUG-3/4] ADMIN KEY-WAIT HELPER
//  Blocks until a key is pressed OR:
//    - ADMIN_KEY_WAIT_MS elapses with no key (returns 0)
//    - ADMIN_SESSION_TIMEOUT elapses (returns 0)
//  Caller must treat return value 0 as "abort / timed out".
// ═══════════════════════════════════════════════════════════
char adminKeyWait(unsigned long* sessionStart) {
  unsigned long waitStart = millis();
  while (true) {
    // Check overall session timeout
```

```
    if (millis() - *sessionStart > ADMIN_SESSION_TIMEOUT) return 0;
    // Check per-key idle timeout
    if (millis() - waitStart > ADMIN_KEY_WAIT_MS) return 0;
    char k = keypad.getKey();
    if (k) return k;
    delay(50);
  }
}

// ============================================================
//              ADMIN LOGIN  ('A' key -> enter admin PIN)
//  Press 'A' on the keypad to open this prompt, type the 5-digit
//  admin PIN (auto-submits on the 5th digit). Correct -> adminMode();
//  wrong -> "ADMIN DENIED".  '*' clears, 'D' cancels.
// ============================================================
void adminLogin() {
  lcd.clear();
  lcd.print("ADMIN PIN:");

  char buf[6];
  int  len = 0;
  buf[0] = '\0';

  unsigned long session = millis();
  while (true) {
    char key = adminKeyWait(&session);

    if (key == 0) {                       // idle / session timeout
      lcd.clear();
      lcd.print("TIMEOUT");
      beepWarning();
      delay(1000);
      defaultState();
      return;
    }

    if (key >= '0' && key <= '9') {
      if (len < PIN_LENGTH) {
        buf[len++] = key;
        buf[len]   = '\0';
        lcd.setCursor(0, 1);
        for (int i = 0; i < len; i++) lcd.print('*');

        // Auto-submit once 5 digits are in (same feel as normal PIN entry)
        if (len == PIN_LENGTH) {
          delay(150);
          if (strcmp(buf, adminPin) == 0) {
```

```
            adminMode();                  // correct PIN -> enter admin
          } else {
            lcd.clear();
            lcd.print("ADMIN DENIED");
            beepLong();
            logEvent("Admin login failed");
            delay(2000);
            defaultState();
          }
          return;
        }
      }
    } else if (key == '*') {              // clear entry
      len = 0;
      buf[0] = '\0';
      lcd.clear();
      lcd.print("ADMIN PIN:");
    } else if (key == 'D') {              // cancel / exit
      defaultState();
      return;
    }
    // other keys ignored
  }
}

// ============================================================
//              ADMIN MODE  (session timeout = 60 s)
// ============================================================
void adminMode() {
  lcd.clear();
  lcd.print("ADMIN MODE");
  beepShort();
  logEvent("Admin session started");
  delay(1500);

  bool inAdmin = true;
  unsigned long sessionStart = millis();

  while (inAdmin) {

    // Session timeout guard
    if (millis() - sessionStart > ADMIN_SESSION_TIMEOUT) {
      lcd.clear();
      lcd.print("SESSION TIMEOUT");
      logEvent("Admin session timed out");
      delay(1500);
      break;
```

```
  }

  // ── Menu page 1 ──
  lcd.clear();
  lcd.print("1-ADD  2-DEL PIN");
  lcd.setCursor(0, 1);
  lcd.print("3-ADD  4-DEL RFID");
  delay(2500);

  // ── Menu page 2 ──
  lcd.clear();
  lcd.print("5-LIST  6-CHGPIN");
  lcd.setCursor(0, 1);
  lcd.print("7-RESET 8-EEPROM");
  delay(2500);

  // ── Menu page 3 ──
  lcd.clear();
  lcd.print("9-PROXIMITY");
  lcd.setCursor(0, 1);
  lcd.print("*-EXIT");
  delay(2000);

  lcd.clear();
  lcd.print("Enter 1-9 or *:");

  // [BUG-3] Use adminKeyWait so the session clock is honoured
  char choice = adminKeyWait(&sessionStart);

  if (choice == 0) {
    // Timed out — session timeout message will show on next iteration
    continue;
  }

  sessionStart = millis();  // Reset session timer on activity

  switch (choice) {
    case '1': addUserPIN();      break;
    case '2': deleteUserPIN();   break;
    case '3': addRFID();         break;
    case '4': deleteRFID();      break;
    case '5': listUsers();       break;
    case '6': changeAdminPIN();  break;
    case '7': resetSystem();     break;
    case '8': viewEEPROM();      break;
    case '9': lcd.clear(); lcd.print("NOT AVAILABLE"); delay(1000); break;
    case '*': inAdmin = false;   break;
```

```
      default:
        lcd.clear();
        lcd.print("INVALID CHOICE");
        delay(1000);
        break;
    }
  }

  logEvent("Admin session ended");
  defaultState();
}

// ═══════════════════════════════════════════════════════════
//              ADD USER PIN  [BUG-4 fixed]
// ═══════════════════════════════════════════════════════════
void addUserPIN() {
  if (pinCount >= MAX_USERS) {
    lcd.clear(); lcd.print("MAX USERS REACHED");
    lcd.setCursor(0, 1); lcd.print("Cannot add more");
    delay(1500); return;
  }

  lcd.clear();
  lcd.print("ENTER NEW PIN:");
  char newPIN[6] = "";
  int  len = 0;
  unsigned long session = millis();

  while (true) {
    char key = adminKeyWait(&session);
    if (key == 0) { lcd.clear(); lcd.print("TIMEOUT"); delay(1000); return; }
    if (key >= '0' && key <= '9' && len < PIN_LENGTH) {
      newPIN[len] = key; newPIN[++len] = '\0';
      lcd.setCursor(len - 1, 1); lcd.print('*');
    } else if (key == '#' && len == PIN_LENGTH) {
      break;
    } else if (key == '*') {
      return;
    }
  }

  // Duplicate check
  for (int i = 0; i < pinCount; i++) {
    if (strcmp(authorizedPins[i], newPIN) == 0) {
      lcd.clear(); lcd.print("PIN ALREADY");
      lcd.setCursor(0, 1); lcd.print("EXISTS");
      delay(1500); return;
```

```
    }
  }

  strcpy(authorizedPins[pinCount], newPIN);
  userIDs[pinCount] = nextUserID;   // [FEAT-2] assign next sequential ID
  pinCount++;
  nextUserID++;
  saveAuthorizedPins();
  saveUserIDs();                    // [FEAT-5] persist IDs to EEPROM

  // Show which ID was assigned
  char idLabel[12];
  snprintf(idLabel, sizeof(idLabel), "User %02d", userIDs[pinCount - 1]);
  lcd.clear();
  lcd.print(idLabel);
  lcd.setCursor(0, 1);
  lcd.print("PIN ADDED");

  char logMsg[40];
  snprintf(logMsg, sizeof(logMsg), "Admin: PIN added [%s]", idLabel);
  logEvent(logMsg);
  delay(1500);
}

// ============================================================
//              DELETE USER PIN  [BUG-4 fixed]
// ============================================================
void deleteUserPIN() {
  lcd.clear();
  lcd.print("PIN TO DELETE:");
  char delPIN[6] = "";
  int  len = 0;
  unsigned long session = millis();

  while (true) {
    char key = adminKeyWait(&session);
    if (key == 0) { lcd.clear(); lcd.print("TIMEOUT"); delay(1000); return; }
    if (key >= '0' && key <= '9' && len < PIN_LENGTH) {
      delPIN[len] = key; delPIN[++len] = '\0';
      lcd.setCursor(len - 1, 1); lcd.print('*');
    } else if (key == '#' && len == PIN_LENGTH) {
      break;
    } else if (key == '*') {
      return;
    }
  }
```

```
  for (int i = 0; i < pinCount; i++) {
    if (strcmp(authorizedPins[i], delPIN) == 0) {
      // [FEAT-3] capture ID before shifting array
      char idLabel[12];
      snprintf(idLabel, sizeof(idLabel), "User %02d", userIDs[i]);

      for (int j = i; j < pinCount - 1; j++) {
        strcpy(authorizedPins[j], authorizedPins[j + 1]);
        userIDs[j] = userIDs[j + 1];   // shift IDs in sync
      }
      userIDs[pinCount - 1] = 0;       // clear vacated slot
      pinCount--;
      saveAuthorizedPins();
      saveUserIDs();                    // [FEAT-5]

      lcd.clear(); lcd.print(idLabel);
      lcd.setCursor(0, 1); lcd.print("PIN REMOVED");

      char logMsg[40];
      snprintf(logMsg, sizeof(logMsg), "Admin: PIN removed [%s]", idLabel);
      logEvent(logMsg);
      delay(1500);
      return;
    }
  }
  lcd.clear(); lcd.print("PIN NOT FOUND");
  delay(1500);
}

// ═══════════════════════════════════════════════════════════
//              ADD RFID TAG  [BUG-4 fixed — cancel timeout]
// ═══════════════════════════════════════════════════════════
void addRFID() {
  if (rfidCount >= MAX_RFID_TAGS) {
    lcd.clear(); lcd.print("MAX RFID REACHED");
    delay(1500); return;
  }

  lcd.clear();
  lcd.print("SCAN NEW RFID");
  lcd.setCursor(0, 1);
  lcd.print("(* to cancel)");

  unsigned long session = millis();

  while (true) {
    // [BUG-4] Honour both the per-function timeout and session clock
```

```
    if (millis() - session > ADMIN_KEY_WAIT_MS) {
      lcd.clear(); lcd.print("TIMEOUT"); delay(1000); return;
    }

    if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
      char newUID[15];
      getRFIDString(newUID, 15);

      bool dup = false;
      for (int i = 0; i < rfidCount; i++) {
        if (strcmp(authorizedRFIDs[i], newUID) == 0) { dup = true; break; }
      }

      if (!dup) {
        strcpy(authorizedRFIDs[rfidCount++], newUID);
        saveAuthorizedRFIDs();
        lcd.clear(); lcd.print("RFID ADDED");
        logEvent("Admin: RFID tag added");
      } else {
        lcd.clear(); lcd.print("RFID EXISTS");
      }
      delay(1500);
      rfid.PICC_HaltA();
      rfid.PCD_StopCrypto1();
      return;
    }
    char key = keypad.getKey();
    if (key == '*') return;
    delay(100);
  }
}

// ═══════════════════════════════════════════════════════════
//              DELETE RFID TAG  [BUG-4 fixed]
// ═══════════════════════════════════════════════════════════
void deleteRFID() {
  lcd.clear();
  lcd.print("SCAN RFID TO DEL");
  lcd.setCursor(0, 1);
  lcd.print("(* to cancel)");

  unsigned long session = millis();

  while (true) {
    if (millis() - session > ADMIN_KEY_WAIT_MS) {
      lcd.clear(); lcd.print("TIMEOUT"); delay(1000); return;
    }
```

```
    if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
      char delUID[15];
      getRFIDString(delUID, 15);

      for (int i = 0; i < rfidCount; i++) {
        if (strcmp(authorizedRFIDs[i], delUID) == 0) {
          for (int j = i; j < rfidCount - 1; j++)
            strcpy(authorizedRFIDs[j], authorizedRFIDs[j + 1]);
          rfidCount--;
          saveAuthorizedRFIDs();
          lcd.clear(); lcd.print("RFID REMOVED");
          logEvent("Admin: RFID tag removed");
          delay(1500);
          rfid.PICC_HaltA();
          rfid.PCD_StopCrypto1();
          return;
        }
      }
      lcd.clear(); lcd.print("RFID NOT FOUND");
      delay(1500);
      rfid.PICC_HaltA();
      rfid.PCD_StopCrypto1();
      return;
    }
    char key = keypad.getKey();
    if (key == '*') return;
    delay(100);
  }
}

// ═══════════════════════════════════════════════════════════
//              LIST USERS  (LCD summary + full Serial dump)
// ═══════════════════════════════════════════════════════════
void listUsers() {
  lcd.clear();
  lcd.print("PINs: ");
  lcd.print(pinCount);
  lcd.setCursor(0, 1);
  lcd.print("RFID tags: ");
  lcd.print(rfidCount);
  delay(2500);

  lcd.clear();
  lcd.print("Full list sent");
  lcd.setCursor(0, 1);
  lcd.print("to Serial port");
```

```
  delay(2000);

  Serial.println("\n+--- USER PINs ---+");
  for (int i = 0; i < pinCount; i++) {
    Serial.print("  [User ");
    if (userIDs[i] < 10) Serial.print("0");
    Serial.print(userIDs[i]);
    Serial.print("] PIN: ");
    Serial.println(authorizedPins[i]);
  }
  Serial.print("  Total: "); Serial.println(pinCount);

  Serial.println("+--- RFID TAGS ---+");
  for (int i = 0; i < rfidCount; i++) {
    Serial.print("  RFID ");
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.println(authorizedRFIDs[i]);
  }
  Serial.print("  Total: "); Serial.println(rfidCount);
  Serial.println("+-----------------+\n");
}

// ═══════════════════════════════════════════════════════════════
//              CHANGE ADMIN PIN  [BUG-4 fixed]
// ═══════════════════════════════════════════════════════════════
void changeAdminPIN() {
  unsigned long session = millis();

  // Step 1: verify old PIN
  lcd.clear(); lcd.print("OLD PIN:");
  char old[6] = ""; int ol = 0;
  while (true) {
    char key = adminKeyWait(&session);
    if (key == 0) { lcd.clear(); lcd.print("TIMEOUT"); delay(1000); return; }
    if (key >= '0' && key <= '9' && ol < PIN_LENGTH) {
      old[ol] = key; old[++ol] = '\0';
      lcd.setCursor(ol - 1, 1); lcd.print('*');
    } else if (key == '#' && ol == PIN_LENGTH) { break; }
      else if (key == '*') { return; }
  }
  if (strcmp(old, adminPin) != 0) {
    lcd.clear(); lcd.print("WRONG PIN"); delay(1500); return;
  }

  // Step 2: new PIN
  lcd.clear(); lcd.print("NEW PIN:");
```

```
  char np[6] = ""; int nl = 0;
  while (true) {
    char key = adminKeyWait(&session);
    if (key == 0) { lcd.clear(); lcd.print("TIMEOUT"); delay(1000); return; }
    if (key >= '0' && key <= '9' && nl < PIN_LENGTH) {
      np[nl] = key; np[++nl] = '\0';
      lcd.setCursor(nl - 1, 1); lcd.print('*');
    } else if (key == '#' && nl == PIN_LENGTH) { break; }
      else if (key == '*') { return; }
  }

  // Step 3: confirm
  lcd.clear(); lcd.print("CONFIRM PIN:");
  char cp[6] = ""; int cl = 0;
  while (true) {
    char key = adminKeyWait(&session);
    if (key == 0) { lcd.clear(); lcd.print("TIMEOUT"); delay(1000); return; }
    if (key >= '0' && key <= '9' && cl < PIN_LENGTH) {
      cp[cl] = key; cp[++cl] = '\0';
      lcd.setCursor(cl - 1, 1); lcd.print('*');
    } else if (key == '#' && cl == PIN_LENGTH) { break; }
      else if (key == '*') { return; }
  }

  if (strcmp(np, cp) != 0) {
    lcd.clear(); lcd.print("MISMATCH"); delay(1500); return;
  }

  strcpy(adminPin, np);
  saveAdminPin();
  lcd.clear(); lcd.print("ADMIN PIN");
  lcd.setCursor(0, 1); lcd.print("CHANGED");
  logEvent("Admin PIN changed");
  delay(1500);
}

// ═══════════════════════════════════════════════════════════
//              RESET SYSTEM  [BUG-4 fixed]
// ═══════════════════════════════════════════════════════════
void resetSystem() {
  lcd.clear();
  lcd.print("CONFIRM RESET?");
  lcd.setCursor(0, 1);
  lcd.print("# = YES  * = NO");

  unsigned long session = millis();
  while (true) {
```

```
    char key = adminKeyWait(&session);
    if (key == 0) { lcd.clear(); lcd.print("TIMEOUT"); delay(1000); return; }
    if (key == '#') {
      clearEEPROM();
      pinCount  = 0;
      rfidCount = 0;
      strcpy(adminPin, "12345");
      saveAdminPin();
      lcd.clear(); lcd.print("SYSTEM RESET");
      logEvent("Full system reset by admin");
      delay(2000);
      return;
    } else if (key == '*') {
      return;
    }
  }
}

// ═══════════════════════════════════════════════════════════
//              CLEAR EEPROM  [BUG-5 fixed]
// ═══════════════════════════════════════════════════════════
void clearEEPROM() {
  lcd.clear();
  lcd.print("CLEARING EEPROM");
  // [BUG-5] Use i < 4096 (not <= EEPROM_MAX_ADDR == 4095) to avoid
  //         overflow, and fix the percentage formula with a long cast.
  for (int i = 0; i < EEPROM_TOTAL; i++) {
    EEPROM.update(i, 0);
    if (i % 512 == 0 && i > 0) {
      lcd.setCursor(0, 1);
      lcd.print((int)((long)i * 100L / EEPROM_TOTAL));
      lcd.print("%  ");
    }
  }
  lcd.clear(); lcd.print("EEPROM CLEARED");
  delay(1000);
}

// ═══════════════════════════════════════════════════════════
//              VIEW EEPROM
// ═══════════════════════════════════════════════════════════
void viewEEPROM() {
  lcd.clear();
  lcd.print("Sending to");
  lcd.setCursor(0, 1);
  lcd.print("Serial...");
  logEvent("Admin: EEPROM dump requested");
```

```
  delay(1000);
  debugReadEEPROM();
  lcd.clear();
  lcd.print("Done. Press any");
  lcd.setCursor(0, 1);
  lcd.print("key to continue");
  while (!keypad.getKey()) delay(100);
  delay(300);
}

// Plain ASCII box drawing for reliable Serial Monitor rendering
void debugReadEEPROM() {
  Serial.println("\n+================================================================
==+");
  Serial.println("|           EEPROM CONTENTS  (Mega 2560 - 4096
bytes)           |");
  Serial.println("+================================================================
+");

  byte pc = EEPROM.read(PIN_COUNT_ADDR);
  byte rc = EEPROM.read(RFID_COUNT_ADDR);
  byte px = EEPROM.read(PROXIMITY_ENABLED_ADDR);

  Serial.print("| PIN count  (addr  0): ");
Serial.print(pc);  Serial.println("                                    |");
  Serial.print("| RFID count (addr  2): ");
Serial.print(rc);  Serial.println("                                    |");
  Serial.print("| Proximity  (addr500): "); Serial.println((px == 1) ?
"ENABLED                          |"
                                                                    :
"DISABLED                         |");

  Serial.println("+--- Admin PIN (addr 10-15) ----------------------------------
+");
  Serial.print("| ");
  for (int a = ADMIN_PIN_ADDR; a < ADMIN_PIN_ADDR + 6; a++) {
    char c = (char)EEPROM.read(a);
    if (c == '\0' || (byte)c == 0xFF) break;
    Serial.print(c);
  }
  Serial.println("                                                          |"
);

  Serial.println("+--- User PINs (addr 50-199) -----------------------------------
+");
  int addr = USER_PINS_START;
  if (pc == 0 || pc > MAX_USERS) {
```

```
    Serial.println("|  (none
stored)                                                |");
  } else {
    for (int i = 0; i < pc && i < MAX_USERS; i++) {
      Serial.print("| PIN "); Serial.print(i + 1); Serial.print(": ");
      int idx = 0;
      while (true) {
        char c = (char)EEPROM.read(addr++);
        if (c == '\0' || idx >= 5) break;
        Serial.print(c); idx++;
      }
      Serial.println("                                                        |")
;
    }
  }

  Serial.println("+--- RFID UIDs (addr 200-499) ----------------------------------
+");
  addr = RFID_START;
  if (rc == 0 || rc > MAX_RFID_TAGS) {
    Serial.println("|  (none
stored)                                                |");
  } else {
    for (int i = 0; i < rc && i < MAX_RFID_TAGS; i++) {
      Serial.print("| RFID "); Serial.print(i + 1); Serial.print(": ");
      int idx = 0;
      while (true) {
        char c = (char)EEPROM.read(addr++);
        if (c == '\0' || idx >= 14) break;
        Serial.print(c); idx++;
      }
      Serial.println("                                                      |");
    }
  }

  Serial.println("+--- Raw hex dump (first 256 bytes) -----------------------------
+");
  for (int i = 0; i < 256; i++) {
    if (i % 16 == 0) {
      Serial.print("| 0x");
      if (i < 0x10) Serial.print("0");
      Serial.print(i, HEX);
      Serial.print(": ");
    }
    byte v = EEPROM.read(i);
    if (v < 0x10) Serial.print("0");
    Serial.print(v, HEX);
```

```
    Serial.print(" ");
    if ((i + 1) % 16 == 0) {
      Serial.print("| ");
      for (int j = i - 15; j <= i; j++) {
        char c = (char)EEPROM.read(j);
        Serial.print((c >= 32 && c <= 126) ? c : '.');
      }
      Serial.println(" |");
    }
  }
  Serial.println("+=================================================================
+\n");
}

// ═══════════════════════════════════════════════════════════════
//              EEPROM PERSISTENCE
// ═══════════════════════════════════════════════════════════════
void saveAuthorizedPins() {
  EEPROM.update(PIN_COUNT_ADDR, (byte)pinCount);
  int addr = USER_PINS_START;
  for (int i = 0; i < pinCount; i++) {
    for (int j = 0; authorizedPins[i][j] != '\0' && addr < RFID_START - 1; j++)
      EEPROM.update(addr++, (byte)authorizedPins[i][j]);
    EEPROM.update(addr++, 0);
  }
}

void loadAuthorizedPins() {
  pinCount = (int)EEPROM.read(PIN_COUNT_ADDR);
  if (pinCount > MAX_USERS || pinCount == 255) pinCount = 0;
  int addr = USER_PINS_START;
  for (int i = 0; i < pinCount; i++) {
    int idx = 0; char c;
    while ((c = (char)EEPROM.read(addr++)) != '\0' && addr < RFID_START && idx < 5)
      authorizedPins[i][idx++] = c;
    authorizedPins[i][idx] = '\0';
  }
}

// [FEAT-5] Save user IDs — 1 byte per slot at USER_IDS_START
// Also saves nextUserID in the byte after the last slot (addr + MAX_USERS)
void saveUserIDs() {
  for (int i = 0; i < MAX_USERS; i++)
    EEPROM.update(USER_IDS_START + i, (byte)userIDs[i]);
  EEPROM.update(USER_IDS_START + MAX_USERS, (byte)nextUserID);
}
```

```
// [FEAT-5] Load user IDs from EEPROM; reconstruct nextUserID
void loadUserIDs() {
  for (int i = 0; i < MAX_USERS; i++) {
    byte v = EEPROM.read(USER_IDS_START + i);
    userIDs[i] = (v == 255) ? 0 : (int)v;
  }
  byte nxt = EEPROM.read(USER_IDS_START + MAX_USERS);
  if (nxt == 0 || nxt == 255) {
    // Fresh EEPROM or reset — derive nextUserID from loaded IDs
    nextUserID = 1;
    for (int i = 0; i < MAX_USERS; i++)
      if (userIDs[i] >= nextUserID) nextUserID = userIDs[i] + 1;
  } else {
    nextUserID = (int)nxt;
  }
}

void saveAuthorizedRFIDs() {
  EEPROM.update(RFID_COUNT_ADDR, (byte)rfidCount);
  int addr = RFID_START;
  for (int i = 0; i < rfidCount; i++) {
    for (int j = 0; authorizedRFIDs[i][j] != '\0' && addr < PROXIMITY_ENABLED_ADDR
- 1; j++)
      EEPROM.update(addr++, (byte)authorizedRFIDs[i][j]);
    EEPROM.update(addr++, 0);
  }
}

void loadAuthorizedRFIDs() {
  rfidCount = (int)EEPROM.read(RFID_COUNT_ADDR);
  if (rfidCount > MAX_RFID_TAGS || rfidCount == 255) rfidCount = 0;
  int addr = RFID_START;
  for (int i = 0; i < rfidCount; i++) {
    int idx = 0; char c;
    while ((c = (char)EEPROM.read(addr++)) != '\0' && addr < PROXIMITY_ENABLED_ADDR
&& idx < 14)
      authorizedRFIDs[i][idx++] = c;
    authorizedRFIDs[i][idx] = '\0';
  }
}

void saveAdminPin() {
  int addr = ADMIN_PIN_ADDR;
  for (int i = 0; adminPin[i] != '\0' && addr < USER_PINS_START - 1; i++)
    EEPROM.update(addr++, (byte)adminPin[i]);
  EEPROM.update(addr, 0);
}
```

```
void loadAdminPin() {
  int addr = ADMIN_PIN_ADDR;
  int idx  = 0; char c;
  while ((c = (char)EEPROM.read(addr++)) != '\0' &&
         (byte)c != 0xFF && addr < USER_PINS_START && idx < 5)
    adminPin[idx++] = c;
  adminPin[idx] = '\0';

  // [FIX] The admin PIN must be EXACTLY 5 digits. A blank, short, or
  // garbage EEPROM slot (stale bytes that are neither 0x00 nor 0xFF) used
  // to load as a bogus admin PIN, locking out the real default. Validate
  // it; if invalid, restore "12345" and persist so it self-heals.
  bool valid = (idx == 5);
  for (int i = 0; valid && i < 5; i++)
    if (adminPin[i] < '0' || adminPin[i] > '9') valid = false;
  if (!valid) {
    strcpy(adminPin, "12345");
    saveAdminPin();
  }
}

// ═══════════════════════════════════════════════════════════════
//              RFID UID HELPER
// ═══════════════════════════════════════════════════════════════
void getRFIDString(char* buf, int maxLen) {
  int idx = 0;
  for (byte i = 0; i < rfid.uid.size && (idx + 2) < maxLen; i++)
    idx += sprintf(buf + idx, "%02X", rfid.uid.uidByte[i]);
  buf[idx] = '\0';
}

// ═══════════════════════════════════════════════════════════════
//              EVENT LOGGING
// ═══════════════════════════════════════════════════════════════
void logEvent(const char* event) {
  char ts[22];
  if (rtcAvailable) {
    DateTime now = rtc.now();
    snprintf(ts, sizeof(ts), "%02d/%02d/%02d %02d:%02d:%02d",
             now.day(), now.month(), now.year() - 2000,
             now.hour(), now.minute(), now.second());
  } else {
    snprintf(ts, sizeof(ts), "T+%lus", millis() / 1000UL);
  }
  Serial.print(ts);
  Serial.print(" | ");
```

```
  Serial.println(event);
}

// ═══════════════════════════════════════════════════════════
//              BUZZER HELPERS
// ═══════════════════════════════════════════════════════════
void beepShort() {
  tone(BUZZER_PIN, 2000, 200);
  delay(250);
}

void beepLong() {
  tone(BUZZER_PIN, 500, 1000);
  delay(1050);
}

void beepAlarm() {
  for (int i = 0; i < 5; i++) {
    tone(BUZZER_PIN, 1000, 200);
    delay(300);
    noTone(BUZZER_PIN);
    delay(100);
  }
}

// Used for non-critical alerts (invalid key, entry timeout)
void beepWarning() {
  tone(BUZZER_PIN, 2500, 100); delay(120);
  tone(BUZZER_PIN, 2000, 100); delay(120);
}
```

## AT Command that was flashed to Key Tag

```
// ═══════════════════════════════════════════════════

#include <SoftwareSerial.h>

#define HM10_RX_PIN   2
#define HM10_TX_PIN   3
#define MONITOR_BAUD  9600
#define HM10_BAUD     9600
#define CMD_DELAY     600
#define RENEW_DELAY   1500

// Advertising interval — change for longer battery life
#define ADVI_SETTING  "AT+ADVI5"    // ~6 days on CR2032
```

```
SoftwareSerial HM10(HM10_RX_PIN, HM10_TX_PIN);

void sendAT(const char* cmd);
void readResponse();
void printDivider();

// ================================================
void setup() {
  Serial.begin(MONITOR_BAUD);
  HM10.begin(HM10_BAUD);
  delay(1000);

  printDivider();
  Serial.println(F("  key Tag — Standalone Coin Battery Setup"));
  printDivider();

  sendAT("AT");

  sendAT("AT+RENEW");          // Factory reset
  delay(RENEW_DELAY);

  sendAT("AT+ROLE0");          // Peripheral mode
  sendAT("AT+ADTY0");          // Connectable undirected — allows receiver
                               // to find this tag by name using AT+DISC?
                               // After MAC is discovered, AT+DISI? handles
                               // all subsequent RSSI reads automatically.
  sendAT(ADVI_SETTING);        // Advertising interval
  sendAT("AT+PWRM1");          // Auto-sleep between broadcasts (saves battery)
  sendAT("AT+NOTI0");          // No connection notifications
  sendAT("AT+NAMEkey Tag");    // Device name (must match TARGET_TAG_NAME)
  sendAT("AT+IBEA1");          // iBeacon mode ON
  sendAT("AT+RESET");          // Save and apply all settings
  delay(1000);

  printDivider();
  Serial.println(F("  DONE! Settings saved to HM-10."));
  Serial.println(F("  Disconnect Arduino, connect CR2032:"));
  Serial.println(F("    HM-10 VCC → battery (+)"));
  Serial.println(F("    HM-10 GND → battery (-)"));
  printDivider();
}

void loop() {
  if (HM10.available())   Serial.write(HM10.read());
  if (Serial.available()) HM10.write(Serial.read());
}
```

```
void sendAT(const char* cmd) {
  HM10.print(cmd);
  Serial.print(F("  >> ")); Serial.println(cmd);
  delay(CMD_DELAY);
  Serial.print(F("  << "));
  readResponse();
  Serial.println();
  delay(100);
}

void readResponse() {
  unsigned long timeout = millis() + 400;
  bool got = false;
  while (millis() < timeout) {
    if (HM10.available()) {
      Serial.write(HM10.read());
      got = true;
      timeout = millis() + 100;
    }
  }
  if (!got) Serial.print(F("(no response)"));
}

void printDivider() {
  Serial.println(F("  ------------------------------------------"));
}
```

# Budget of key access box

Table 15: Budget of key access box

| PROPOSED BUDGET OF THE KEY STORAGE BOX | | | | |
|---|---|---|---|---|
| **S/N** | **REQUIREMENT** | **QUANTITY** | **UNIT PRICE** | **AMOUNT** |
| 1 | TRANSPORT | 3 | UGX 10,000 | UGX 30,000 |
| **MECHANICAL REQUIREMENTS** | | | | |
| 2 | 1.5 mm mild steel | 1/2 a sheet | UGX 100,000 | UGX 50,000 |
| 3 | Hinge | 1 | UGX 1,500 | UGX 1500 |
| | | | | |
| **ELECTRONIC REQUIREMENTS** | | | | |
| 4 | LCD 16x2 Character Module Blue | 1 | UGX 30,000 | UGX 30,000 |
| 5 | 12x18cm Universal PCB DIY PROTOTYPE BOARD | 1 | UGX 8,000 | UGX 8,000 |
| 6 | 4x4 Matrix Keyboard Keypad Module | 1 | UGX 20,000 | UGX 20,000 |
| 7 | MFRC-522 RC522 RFID + S50 CARD | 1 | UGX 20,000 | UGX 20,000 |
| 8 | MG-90S Metal Gear Micro Servo | 2 | UGX 20,000 | UGX 40,000 |
| 9 | 5VDC Micro Mini Solenoid Lock | 1 | UGX 45,000 | UGX 45,000 |
| 10 | RELAY MODULE 5V 10A/250VAC | 1 | UGX 15,000 | UGX 15,000 |
| 11 | 5V 3A DC Adapter | 1 | UGX 30,000 | UGX 30,000 |
| 12 | 2IN1 Charger + Boost Converter (DD012CVSA) | | UGX 30,000 | UGX 30,000 |
| 13 | 4056A LIPO /Li-ion BATTERY CHARGE HOLDER | 1 | UGX 8,000 | UGX 8,000 |
| 14 | Original 3.7V 7800mAh 18650 Polymer Lithium Ion Battery (LiPo) | 4 | UGX 18,000 | UGX 72,000 |

| | | | | |
|---|---|---|---|---|
| 15 | HDC 3-24V High Decibel Active Buzzer | 1 | UGX 4,000 | UGX 4,000 |
| 16 | 560mm LED for Arduino Raspberry Pi | 2 | UGX 500 | UGX 1,000 |
| 17 | Limit Switch Micro Switch 5A 250V KW11-3Z | 3 | UGX 5,000 | UGX 15,000 |
| 18 | 250mA Glass Fuse 32 x 6mm | 2 | UGX 1,500 | UGX 3,000 |
| 19 | 10cm Multi-coloured Dupont Wire 40-pin Male to Female Jumper wires | 30 | UGX 500 | UGX 15,000 |
| 20 | Screws | 10 | UGX 300 | UGX 3,000 |
| 21 | Bolts and nuts | 10 | UGX 500 | UGX 5,000 |
| 22 | HM-10 modules | 2 | UGX 30,000 | UGX 60,000 |
| 23 | Arduino Mega MCU | 1 | UGX 100,000 | UGX 100,000 |
| 24 | Miscellaneous items | 1 | UGX 116,000 | UGX 116,000 |
| **TOTAL AMOUNT** | | | | **UGX 721,500** |

# Table Section

Table 16: Key Tag Distance Measurement Performance

| Actual Distance (m) | Average RSSI (dBm) | Calculated Distance (m) | Error (%) |
|---|---|---|---|
| 0.5 | -53.2 | 0.52 | 4.0 |
| 1.0 | -59.0 | 1.00 | 0.0 |
| 2.0 | -64.8 | 1.95 | 2.5 |
| 3.0 | -68.5 | 2.89 | 3.7 |
| 4.0 | -71.3 | 3.85 | 3.8 |
| 5.0 | -73.8 | 5.10 | 2.0 |
| 7.0 | -78.2 | 7.25 | 3.6 |
| 10.0 | -83.5 | 10.80 | 8.0 |

Table 17: Key Tag and Receiver Communication Architecture

| **Aspect** | **Key Tag (Transmitter)** | **Receiver (Key Box)** |
|---|---|---|
| HM-10 Role | Peripheral / Slave (AT+ROLE0) | Central / Master (AT+ROLE1) |
| Arduino Required | No (standalone after config) | Yes (Arduino Mega) |
| Power Source | CR2032 coin cell (3V) | 5V mains + 3.7V LiPo backup |
| Function | Broadcast iBeacon packets | Scan, parse RSSI, calculate distance |
| Device Name | "Key Tag" | N/A (scans for name) |
| Update Interval | 760 ms (configurable) | 2 seconds (scan interval) |

| | | |
|---|---|---|
| Battery Life | ~6 days (ADVI5) | Continuous (mains powered) |

Table 18: Key Tag AT Command Configuration Sequence

| **Step** | **Command** | **Function** | **Expected Response** |
|---|---|---|---|
| 1 | AT | Verify communication | OK |
| 2 | AT+RENEW | Factory reset (clears previous settings) | OK+RENEW |
| 3 | AT+RESET | Reboot module after reset | OK+RESET |
| 4 | AT+ROLE0 | Set as peripheral (slave/tag mode) | OK+Set:0 |
| 5 | AT+ADTY0 | Set connectable undirected advertising | OK+Set:0 |
| 6 | AT+ADVI5 | Set advertising interval to 760 ms | OK+Set:5 |
| 7 | AT+PWRM1 | Enable auto-sleep between broadcasts | OK+Set:1 |
| 8 | AT+NOTI0 | Disable connection notifications | OK+Set:0 |
| 9 | AT+NAMEkey Tag | Set device broadcast name | OK+Set:key Tag |
| 10 | AT+IBEA1 | Enable iBeacon mode | OK+Set:1 |
| 11 | AT+RESET | Save all settings and reboot | OK+RESET |

Table 19: Advertising Interval and Battery Life Trade-off

| **Setting** | **Advertising Interval** | **Average Current** | **Battery Life (CR2032, 220 mAh)** | **Application Suitability** |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| AT+ADVI3 | 417 ms | ~22 mA | ~3 days | Fast-moving tag, frequent updates |
| AT+ADVI4 | 589 ms | ~18 mA | ~4 days | Balanced performance |
| AT+ADVI5 | 760 ms | ~15 mA | ~6 days | **Selected for this project** |
| AT+ADVI6 | 1,020 ms | ~11 mA | ~9 days | Slow-moving tag |
| AT+ADVI9 | 10,240 ms | ~2.5 mA | ~90 days | Stationary asset tracking |

# Timeline of the project

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | GANTT CHART/ TIMELINE FOR DESIGN AND CONSTRUCTION OF A 5-CODE KEY BOX FOR MULTIPLE OCCUPANCY OFFICES OF THE | | | | | | | | | | | | | |
| 2 | | YEAR: | 2025 | | | | | 2026 | | | | | | |
| 3 | | MONTH: | AUG | SEPT | OCT | NOV | DEC | JAN | FEB | MAR | APR | MAY | JUNE | |
| 4 | S/N | TASK NAME | | | | | | | | | | | | |
| 5 | 1.0 | GETTING PROJECT TITLE | | | | | | | | | | | | |
| 6 | 1.1 | INTRODUCTION AND UNDERSTANDING THE TOPIC | | | | | | | | | | | | |
| 7 | 1.2 | BACKGROUND | | | | | | | | | | | | |
| 8 | 1.3 | PROBLEM STATEMENT | | | | | | | | | | | | |
| 9 | 1.4 | OBJECTIVES | | | | | | | | | | | | |
| 10 | 1.5 | SIGNIFICANCE | | | | | | | | | | | | |
| 11 | 1.6 | JUSTIFICATION | | | | | | | | | | | | |
| 12 | 1.7 | SCOPE OF THE WORK | | | | | | | | | | | | |
| 13 | 2.0 | LITERATURE REVIEW | | | | | | | | | | | | |
| 14 | 3.0 | METHODOLOGY | | | | | | | | | | | | |
| 15 | 3.1 | OBJECTIVE 1 | | | | | | | | | | | | |
| 16 | 3.2 | OBJECTIVE 2 | | | | | | | | | | | | |
| 17 | 3.3 | OBJECTIVE 3 | | | | | | | | | | | | |
| 18 | 4.0 | PROPOSED BUDGET | | | | | | | | | | | | |
| 19 | 5.0 | GANTT CHART/ TIMELINE | | | | | | | | | | | | |
| 20 | 6.0 | PROJECT PROPOSAL PRESENTATION | | | | | | | | | | | | |
| 21 | 7.0 | DETERMINING THE DESIGN REQUIREMENTS | | | | | | | | | | | | |
| 22 | 8.0 | MATHEMATICAL MODELING AND FORMULAE | | | | | | | | | | | | |
| 23 | 9.0 | CAD DESIGN MODEL | | | | | | | | | | | | |
| 24 | 10.0 | ELECTRONICS SCHEMATICS AND LAYOUT DESIGN | | | | | | | | | | | | |
| 25 | 11.0 | ASSEMBLYING TO FORM A PREIMINARY DESIGN | | | | | | | | | | | | |
| 26 | 12.0 | SIMULATING THE CAD DESIGN | | | | | | | | | | | | |
| 27 | 13.0 | BUYING AND TESTINGMATERIALS | | | | | | | | | | | | |
| 28 | 14.0 | WRITING CODES AND PROGRAM | | | | | | | | | | | | |
| 29 | 15.0 | TESTING AND DEBUGGING THE CODE | | | | | | | | | | | | |
| 30 | 16.0 | BUILDING AND TESTING DIFFERENT PROTOTYPES | | | | | | | | | | | | |
| 31 | 17.0 | PERFORMANCE EVALUTION OF FINAL PROTOTYPE | | | | | | | | | | | | |
| 32 | 18.0 | DOCUMENTATION OF THE FULL PROJECT | | | | | | | | | | | | |
| 33 | 19.0 | FINAL PHASE AND PRESENTATION OF THE KEY STORAGE BOX | | | | | | | | | | | | |