Federal University Of techkigh - Paraná – UTFPR
Academic Department of Eletronic – DAELN
Academic Department of Informatics – DAINF
Computing Engineering
Integration Workshop 3 (EEX23) – S71 – 2023/1

Technical Report
Delivery Safe Box

Luis Camilo J. Moreira – luismoreira@alunos.utfpr.edu.br
Frank E. H. Bloemer – frank.e.h.bloemer@gmail.com
Raissa A. N. Higa – rhiga.2018@alunos.utfpr.edu.br
Ricky L. Habegger –ricky@alunos.utfpr.edu.br

June, 2023

Abstract

With the advances of internet, these days we can buy countless products anywhere and at any time. However, we cannot receive at any time the product we are expecting. This situation brings unwanted consequences, such as packages being returned to the delivery company if the recipient is absent or busy at the moment, packages being dropped (containing some high-value product) or delivery being made to some neighbor (need to have trust in the neighbor). Therefore, focusing on small condominiums without doorman, *the Safebox - a safe delivery box* project intends to solve this problem, providing an asynchronous delivery system, so that the recipient doesn't need to be present at the same moment of the delivery, aiming to be a safe and assertive system for receiving deliveries. In which, it will be a system consisting of safes, being handled by a control system, in which the safes can be opened by any delivery person. In addition, the system will notify the specific user that his or her package has arrived, and the user will be able to pick up the product at another time in a secure way, by means of authentication systems. Thus, the report in question is about the development of the project, which consists of 5 parts, being the Safe system, which represents an actual safe, the Control system, which has the purpose of managing the safes, as well as the facial recognition and biometric module for unlocking the safe, besides interacting with an application on a cloud server. To serve the end users, there is a web application for managing the system, being accessed only by the manager, as well as a mobile application, so that users can interact with the system and make reservations of the safes for their purchased packages.

1 Introduction

This project was developed for the Integration Workshop 3 university discipline, of the Computer Engineering course at UTFPR, Curitiba campus.

As motivating factors for the development of this prototype, we can cite the following problem: in today's world, with the advent of the internet and mobile

devices, we can buy online whenever and wherever we are, without restrictions. However, we are not always at home when the packages arrive at our residence, or we are busy and cannot take delivery. When this happens, some unpleasant situations can occur, such as the package getting dropped, being delivered to a neighbor, or simply returning to the sender.

Taking these factors into consideration, it was possible to define our mission and objectives: to enable an asynchronous delivery system for residents of small condominiums or even residences, providing greater care and security for packages ordered and keeping a log of the actions of this system for security and supervision.

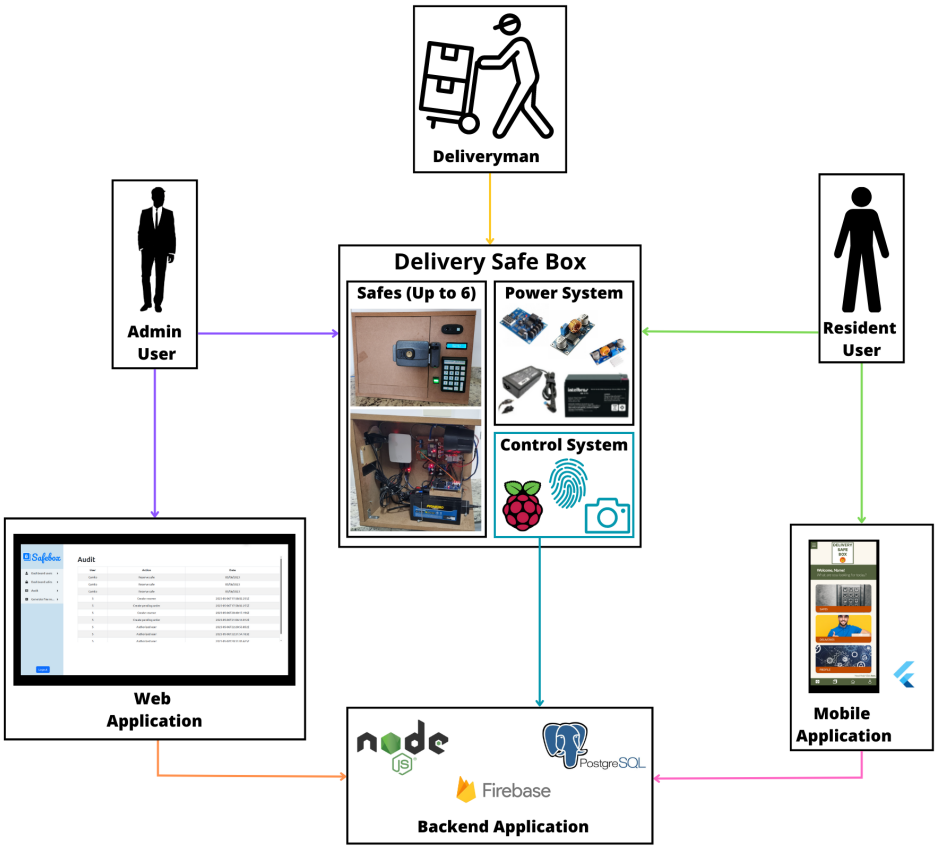

Figure 1: High-level diagram of how the project works

Considering these problems and motivations, it was possible to develop the project described in this document. Analyzing the diagram in Figure 1, it is possible to understand how the actors (Admin User, Resident User and Deliveryman) interact with the central and parallel modules, as well as how they interact with each other and the technologies used to make this possible.

The **Resident User** can interact with the Delivery Safe Box (via keyboard, facial recognition or fingerprint recognition), as well as interact with the mobile application to register/log in, reserve vaults, register deliveries and other functionalities.

The **Admin User** can interact with the Delivery Safe Box in the same way as the Resident User, and can also access the Web Application for management and system control functions.

The **Deliveryman** can place deliveries inside the safe box with the single-use code he receives in the order description.

The Mobile Application and Web Application interact with each other and with the Delivery Safe Box via backend applications, which will be better described in the next sections of this technical report.

The Delivery Safe Box integrates the vaults (up to 6), Power System (electrical components) and Control System (camera, fingerprint reader and camera).

Presenting a complete use case, first the Admin must register the user (by fingerprint, facial registration, or secret password to be entered on the keyboard) then the user can access the system via Smartphone App, reserve a safe for a certain period, in which he waits for the delivery, then take the generated code and include it in the "complement" field of an online purchase address, then the deliveryman delivers (opening the safe by entering the one-time secret code), and then the user is informed by app notification, and can go and collect his delivery using facial recognition, fingerprint, or secret code, ending the use of the system.

2 Project Specification

The project was separated into five main divisions there are the Safe system, that represents a single vault, the control system, that is the controller of the vaults and intereacts with the backend server application and the backend server application, that manages the data between all parts. Finally, there are the parts that relate to the end user,that are the web application, used by admin user for management purposes and the mobile application, that is used by the resident users.

Furthermore, the requirements of all main project parts are described in the sections 2.1, 2.2, 2.3 respectively and in the section 2.4 is contained the anti-requirements of the project.

2.1 Mechanical Requirements

2.1.1 Mechanical Functional Requirements

- MF01 - The system's safe must have a lock
- MF02 - The system must lock the safe after an object is dispatched inside of it

- MF03 - The system's safe must have an internal scale
- MF04 - The system's safe must have a sensor to detect break-in attempts
- MF05 - The system must provide information about how use the system for the residents and the courier
- MF06 - The system must provide a QRCode with contact information of the admin in cases of exceptional issues
- MF07 - The system must have a control system for control the vaults
- MF08 - The system must have a primary safe with complete behavior
- MF09 - The system must have a secondary safe, that will simulate a second safe, for testing purposes

2.1.2 Mechanical Non-Functional Requirements

- MNF01 - The system must support an 25x25x20cm sized object
- MNF02 - The system must support object from 200g to 20Kg
- MNF03 - The system must have a full safe and a second safe with features adapted for cost and prototype purposes
- MNF04 - The system must have inform how use the system through a banner

2.2 Hardware Requirements

2.2.1 Hardware Functional Requirements

Safe system:

- HRF01 - The safe system must allow the courier to unlock the safe with one use code inserted on keyboard.
- HRF02 - The safe system must identify if the safeis being force into with a accelerometer sensor
- HRF03 - The safe system must identify if the safe is open or close with a limit sensor
- HRF04 - The safe system must identify if the safe contains a package with a load cell by weighting the delivery
- HRF05 - The safe system must allow users to view information of the safe system through a LCD Display

- HRF06 - The safe system must sound an alarm when it detects that the safe being forced
- HRF07 - The safe system must lock/unlock the safe with a solenoid lock
- HRF08 - The control system must control the safe system, the recognition module and communicate with the server using a microcontroller
- - HRF09 - The safe system must control the hardware components with a microcontroller

Control system

- HRF10 - The control system must allow the user unlock the safe with fingerprint recognition by a fingerprint module
- HRF11 - The control system must allow the user unlock the safe with facial recognition by a camera

Secondary Safe (Mock up):

- HRF13 - The secondary safe will be simply for demonstration purposes.
- HRF14 - The secondary safe must have a LED to emulate the electronic lock.
- HRF15 - The secondary safe must have a push button to emulate the detection of the object presence.
- HRF16 - The secondary safe must have a limit switch to identify if the door was closed in a simulated way
- HRF17 - The secondary safe must have a push button to emulate the detection of break-in Power System
- HRF18 - The power system must converter the battery volttage to 5V to power up the Raspberry pi microcontroller with a DC converter module
- HRF19 - The system must have a controller module to control the voltage of the 19V source to power the 12V battery

2.2.2 Hardware Non-Functional Requirements

- HNRF01 - The control system must use the microcontroller Raspberry pi 3B+
- HNRF02 - The fingerprint recognition must be done using the FPM10A fingerprint module

- HNRF03 - The facial recognition must be done using a webcam Logitech c270
  - HNRF03 .1 - The minimum resolution of the camera must be 720p.
- HNRF04 - The keyboard for insert the code by delivery person must be a numerical keyboard USB
- HNRF05 - The safe system must have a 16x2 LCD Display to show the information
- HNRF06 - The main safe must have a load cell to detect objects inside the safe
- HNRF07 - The main safe must have a amplifier HX711 to amplify the load cell signal
- HNRF08 - The main safe must have a solenoid lock to lock the safe
- HNRF09 - The main must have the accelerometer sensor MPU6050 to detect break-ins
- HNRF10 - The main and secondary safe must have limit switches 5A 250V to detect if the door is closed
- HNRF11 - The circuits must be mounted in a PCB board
- HNRF12 - The secondary safe must have a white led to simulate the activation of the solenoid lock
- HNRF13 - The secondary safe must have a push button to simulate the load cell sensor
- HNRF14 - The primary safe must have a relay to interact with the solenoid lock
- HNRF15 - The control system must have a relay to activate the sound alarm
- HNRF16 - The microcontroller Raspberry Pi in the control system must communicate with microcontroller ESP in the safe system with UART protocol
- HNRF17 - The power system must use the converter module XI4016
- HNRF18 - The power system must use the controller module Xh-m603
- HNRF19 - The power system must use a 12V battery
- HNRF20 - The power system must use a 19V switched power source
- HNRF21 - The safe sytem must have a microcontroller ESP32 - C3

2.3 Software Requirements

2.3.1 Software Functional Requirements

Mobile application:

- SRF01- The mobile application must allow the user to register an account
- SRF02 - The mobile application must allow the user to register an order that the resident user is waiting for delivery
  - SRF02.1 - The mobile application must allow the user to choose the package type
- SRF03 - The mobile application must allow the user to verify the unreserved/reserved safes and reserve an empty safes
- SRF04 - The mobile application must allow the user to view all his/hers order requests
- SRF05 - The mobile application must allow the user to delete a order request
- SRF06 - The mobile application must allow the user to view notifications of system events, like a package having been delivered or the safe being broken into

Server application (backend and web application)

- SRF07- The server application must generate the codes to unlock a safe by delivery person
- SRF08 - The server application must send the code to the mobile application
- SRF09 - The server application must allow the admin user to view a dashboard screen with a list of the safes empty or in use and if each safe is active or disable for use
- SRF10 - The server application must allow the admin user to view a dashboard with the audit of system, containing every interaction with the system, with action, date-time and user
- SRF11 - The server must generate a report and inform the admin in cases of no retrieval of the package by the user, to apply a fine
- SRF12 - The server must update the control system with the current keys to insert on keyboard
- SRF13 - The system must allow a unique admin user per system

- SRF14 - The control system must send signals periodically to the server system to inform that is active
- SRF15 - The server must send notification for the users on the mobile application, warning that the sever lost the connection with the control system

Control system:

- SRF16 - The control system must send the code inserted on keyboard to server application via Internet
- SRF17 The control system must recognize a person and unlock a vault with facial recognition, in no more than 12 seconds, for up to 120 different users.
- SRF18 - The control system must store four face images locally in microcontroller of control system
- SRF19 - The control system must recognize a person and unlock a vault with fingerprint recognition, in up to two seconds
- SRF20- The control system must store the fingerprint locally in fingerprint module of control system, one finger for each user
- SRF21 - The control system must send to safe system what safe needs to be unlocked/locked
- SRF22 - The control system must allow the resident user register the fingerprint and face to the recognition process

2.3.2 Software Non-Functional Requirements

Mobile application:

- SNRF01 - The mobile application must be developed in Flutter
- SNRF02 - The communication with the server is with HTTP requests
- SNRF03 - The notifications to the users is with pop-up notification

Server:

- SNRF04 - The max number of active reserves is equal to the number of safes
- SNRF05 - The backend server application must be developed with Node.js
- SNRF06 - The server application must be used the PostgreSQL Database

- SNRF07 - The server application must be running on Render Cloud
- SNRF08 - The server application must be receive the data from microcontroller by HTTP requests
- SNRF09 - The facial recognition application must be developed using python
  - SNRF09.1 - The facial recognition application must be developed using face _recognition API and OpenCV
  - SNRF09.2 - The facial recognition application must recognize the registered users as long as they are facing the camera
  - SNRF09.3 - The facial recognition application must recognize the registered users as long as they are positioned up to 1 meter away from the camera
- SNRF10 - The server application must be accessed by user admin only
- SNRF11 - The fingerprint recognition application must be developed using python
  - SNRF11.1 - The fingerprint recognition application must use the Raspberry Pi Fingerprint Library
- SNRF12 - The web application must be developed using React

2.4 Anti-requirements

- PAR01 - It will not allow package withdrawal if there is no electrical power (only via master key).
- PAR02 - Without an Internet connection, it will not be possible to receive notifications or register new deliveries.
- PAR03 - No more than 120 fingerprints can be stored in the biometric module.
- PAR04 - Storage limited by the memory capacity available in the microcontroller.
- PAR05 - It will not be able to detect failures automatically and contact the admin.
- PAR06 - It will not detect lockpicking attempts very easily if the criminal do not move the vault one bit.
- PAR07 - If there is no power source (not even UPS), the alarm will have no way to trigger.

- PAR08 - The scale will not recognize very light objects (load cell operating range: 0.2Kg - 20Kg)
- PAR09 - Facial recognition won't work with people wearing glasses or masks
- PAR10 - Facial recognition won't recognize people in dark places (it does not provide a light source to illuminate users during facial recognition)
- PAR11 - Facial recognition won't recognize a person that is more than 1 meter away from the camera
- PAR12 - It won't be possible to have more active reservations than the number of available vaults
- PAR13 - The facial recognition won't work for a user that doesn't agree with the system keeping his photos saved
- PAR14 - The fingerprint recognition won't work for a user that doesn't agree with the system keeping his fingerprint saved
- PAR15 - The condo must provide the monitoring system for the Delivery Safe Box, as appropriate
- PAR16 - The system's network configuration is not preconfigured. The admin user needs to configure it by following instructions or by interacting with support

3 Development

In this section, all the processes done and difficulties faced during the project's development will be described, together with the technical details.

3.1 Mechanical design

The Mechanic Droject can be divided into two main subprojects, the Safe part and the physical control part. Due to practical and budget reasons, it was decided to build only one complete safe for our system, it was also implemented a secondary mock up safe with features emulated by LEDs and buttons for testing the modularization. The main vault and the control part were assembled in the same unit wooden box with 44,5 x 33,5 x 30,5cm dimensions. The back wall was built in a way that can be removed with a key for easy maintenance. A picture of the mechanical part of the system is presented in figure 2.

Figure 2: Mechanical Structure (To the left the main vault, to the right the control unit)

3.1.1 Control System

The control unit is assembled at the right part of the box. Holes were drilled in the front to fit the hardware that will be used for users interactions, such as the numeric keyboard, the webcam, the fingerprint reader and the LCD.

Inside, the PCB boards and microcontrollers are screwed to the walls, the power system is also there. It was left a free space backside that covers all the box for cable management.

3.1.2 Primary Safe

Our primary safe was designed to be a secure storage unit installed on the left side of our wooden box. It features a hinged door at the front side with enough space to integrate a solenoid lock, providing controlled access to the safe, and a hidden compartment at the bottom to fit a scale. A limit switch was placed strategically to get the data if the door is open or closed. Conduits were made inside the wood for cable routing. An inside view of the safe can be seen in figure 3.

3.1.3 Mock-up Safe

The mock-up serves the purpose of simulating the functionalities of the main safe, serving as a proof of modularity. It allows for the demonstration and testing of the core features of the safe system in a controlled and practical manner.

Figure 3: Inside view

It consists of a cardboard box measuring 13.5 x 8.4 x 13.7cm, featuring a rear opening for cable routing. The design can be seen in figure 4.

3.2 Hardware design

The security and control system for safes consists of three main components: the control hardware, the safe hardware, and the mockup. The control hardware plays a vital role in managing and controlling the operation of the safes, as well as handling recognition and order data processing. The safe hardware is designed to guarantee the protection of stored objects, utilizing features such as the solenoid lock for locking and unlocking the safe, the limit switch for detecting the door position, and the load cell for detecting the presence of objects inside the safe. The system also incorporates the mockup, a simulated representation of other safes supported by the system. An overview of the hardware components can be observed in figure 5. The same ESP32C3-Mini microcontroller is shared between the control hardware and the safe hardware. Also, the PCB (Printed Circuit Board) was fabricated on the same board, as depicted in figure 6.

3.2.1 Control's Hardware

The control system hardware serves as the central unit, receiving information from the server regarding deliveries, providing user interaction through the keypad and display, and performing facial and biometric recognition processes. It

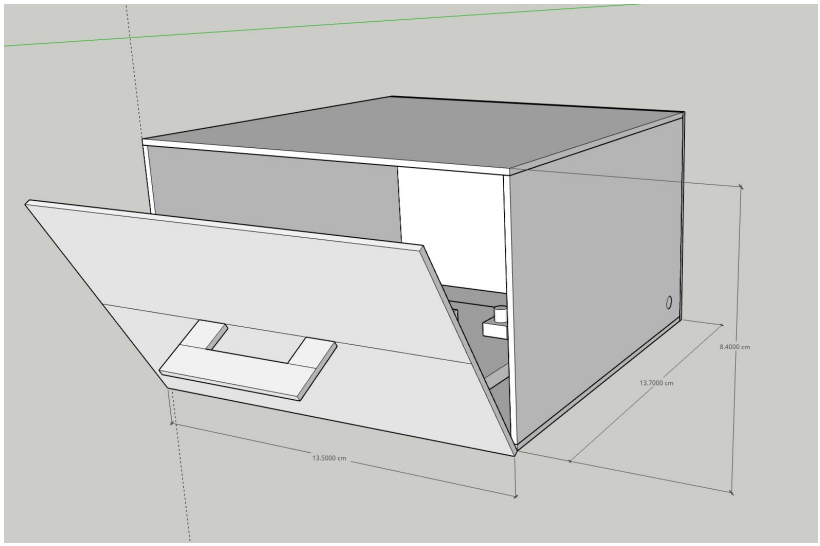

Figure 4: Mock-up design

consists of various components to enable these features, such as a Raspberry Pi 3B utilized for processing tasks, acting as the central unit.

To facilitate user interaction, the control system incorporates a numeric keypad, allowing users to conveniently interact with the system and input necessary information. Additionally, an ESP32C3-Mini microcontroller is connected to the Raspberry Pi via USB to provide a user-friendly interface through a 16x2 LCD utilizing the I2C protocol. For the recognition, the control system integrates a 720p resolution webcam. This webcam captures images that are utilized in the facial recognition process, enhancing security measures. Moreover, a fingerprint module is integrated into the system, connected through a TTL-USB converter. This module enables secure fingerprint recognition, adding an extra layer of authentication and ensuring authorized access to the system.

A relay is responsible for activating and deactivating the siren, enhancing the security features of the safe.

3.2.2 Safe's Hardware

An ESP32C3-Mini microcontroller receives inputs from sensors and send commands to the actuators. A solenoid lock, controlled by a relay, is used to lock and unlock the safe, a limit switch is employed to detect whether the door is open or closed. When the door is closed, the limit switch is pressed, and when it is open, the limit switch is released.

The safe also contains a load cell connected to an HX711 module, consisted of an amplifier and an analog-to-digital converter, is utilized. This module amplifies the load cell signal, enhancing its accuracy and sensitivity. The amplified signal is then easily interpreted by the microcontroller [1]. It is important to

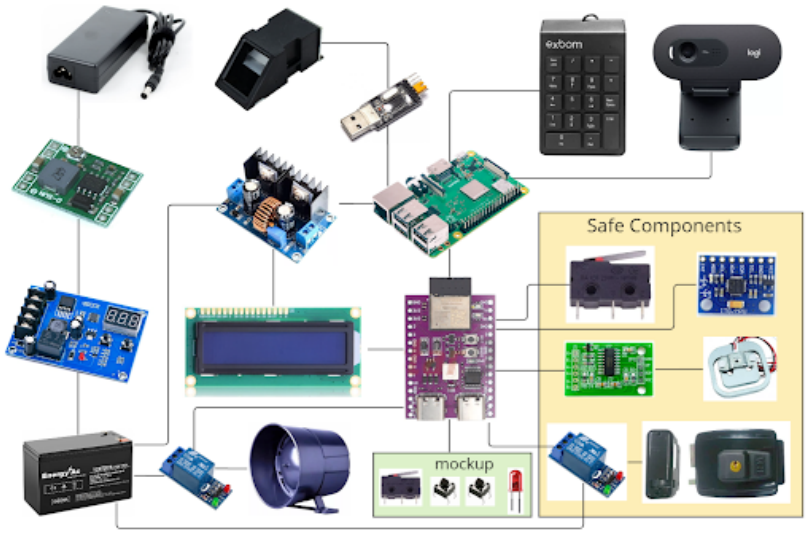

Figure 5: High Level Electronic Desing

note that the load cell is not used for measuring weight, but rather to detect the presence or absence of an object inside the safe.

Additionally, an accelerometer is utilized to measure the acceleration and the movement of the safe's structure, detecting potential attempts at tampering with the structure. The accelerometer chip includes an analog-to-digital converter (ADC) that converts the electrical signals from the sensor into digital signals. Theses digital signals are then passed through a digital filter to prepare them for processing [2].

3.2.3 Mock-up's Hardware

For the mock-up's hardware components, two push buttons represent the load cell and the accelerometer, simulating an object presence and the movement, respectively, when pressed. A limit switch represents the limit switch, however with reverse operation, for this one, the door is closed when pressed. Furthermore, a LED simulates the unlocking of the solenoid when lit. The final result of the mock-up board can be seen in figure 7 .

3.2.4 Power Supply Hardware

The power system is powered by a 19V bivolt power supply. In turn, this source charges a 12V 7Ah lead acid battery that ensures the power supply to the system during power outages. To do this safely we lower the voltage of the supply to 13.8V using a step down converter that feeds a charging controller. This controller starts charging when the battery reaches 13.4V and charges the battery until it reaches 13.8V. The battery powers the alarm siren and the safe locks us-

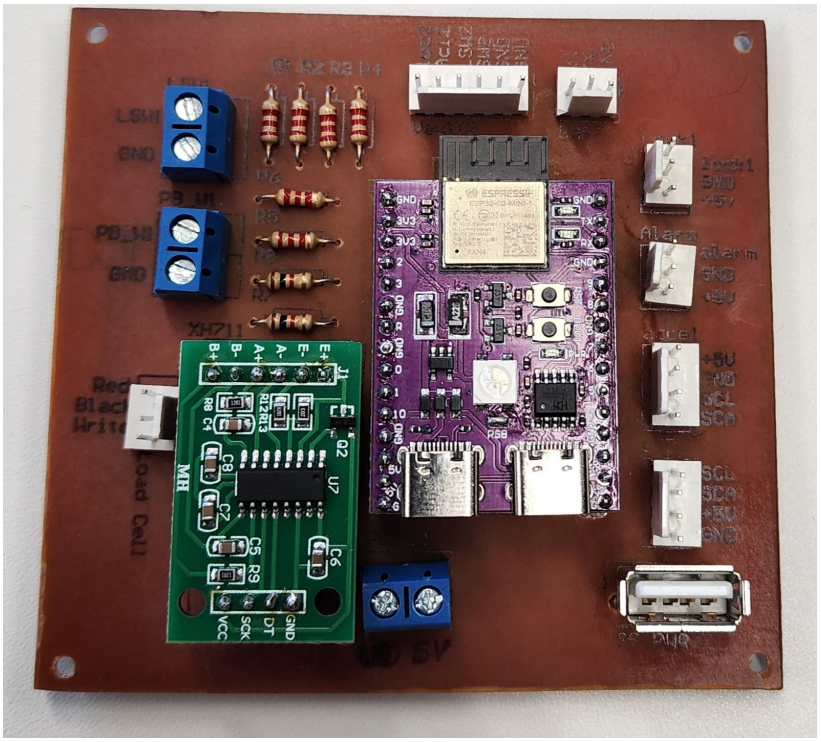

Figure 6: Control and Safe PCB Board

ing relays. To power all the other components we step down the battery voltage to 5V with another step down converter.

3.3 Software design

To have a better understanding of the software design of all parts of the system, in the footlink below there are numerous diagrams illustrating use cases, flowcharts, and some behaviors about the software system.[1]

3.3.1 Mobile application

The mobile application, which was developed in Flutter [3], has the intention of doing the vault reservations and the creation of orders to receive a code, in which such code is used at the time of delivery by the delivery person, besides this, there are other important features that will all be described below.

Initially, the application has the registration step and after the registration, the resident user can successfully login only if the admin user has authorized him to use the system (the authorization occurs in a web application accessed

[1] Software diagrams: https://frankbloemer.notion.site/Deliverable-4-Software-design-43ec9cbe575c4fe7a750322a1b1daa3e

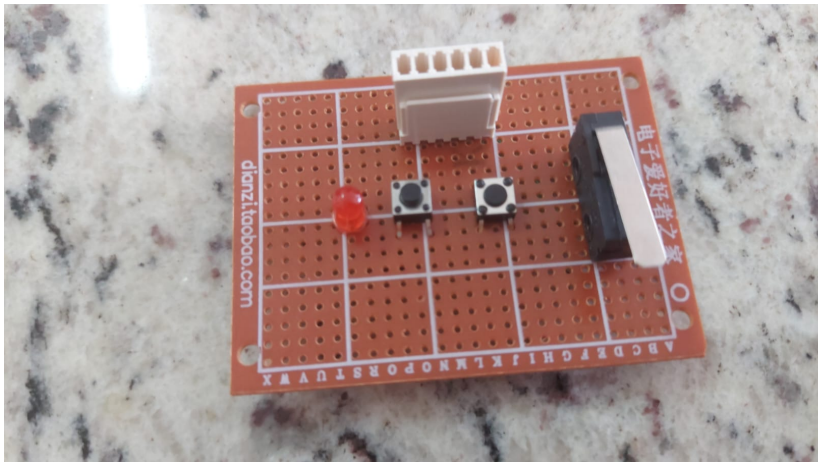

Figure 7: Mock-up Board

only by the admin). After authorization and when the user log in to the system, figure 8 contains the summary flowchart about using the application.

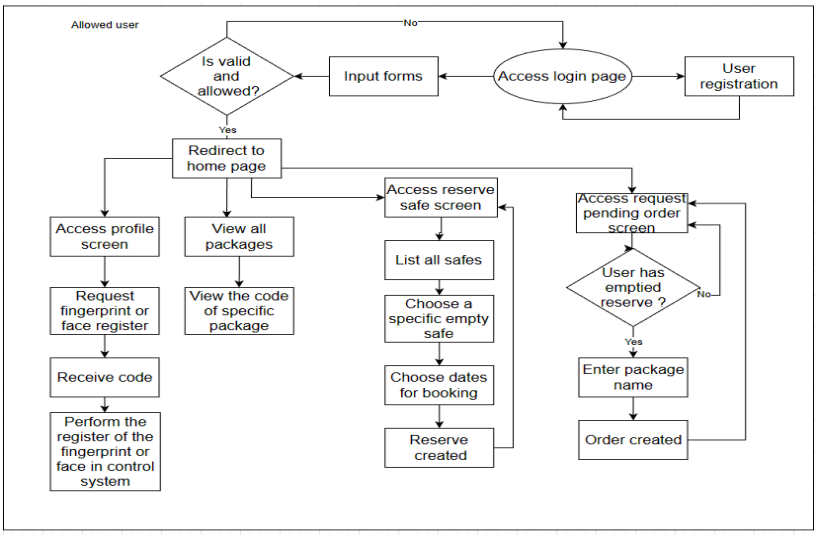

Figure 8: Mobile application flowchart diagram

By checking the figure, the user can, through the profile page, request the code that will be used to register the fingerprint and face in the system for the recognition process of both, besides updating registration data.

Furthermore, there is use with the screens referring to the core of the project, for reserving the vault and creating an order. First of all, it is necessary for the resident user to check the estimated delivery time, but not yet finalize the purchase of the product, because when reserving the safe, it is necessary to inform the reservation period and being aware of the date, he can start the safe's reservation step.

When you access the booking screen, all the vaults in the condominium will be listed, informing the status of each safe. The user can select the empty vaults to reserve one of them. After a successful reservation, the user can create an order, linked to the same reservation and after the reservation, the user will receive the delivery code in the application. Thus, the delivery code is the information that the resident user must enter in the address complement and the same code will be used by the deliveryman during the delivery. This code will be linked the delivery with the order and the specific user. Finally, there will also be a screen to check the orders on hold and completed. Application screens are shown in the figures 9 , 10 , 11 , 12 , 13 and 14.

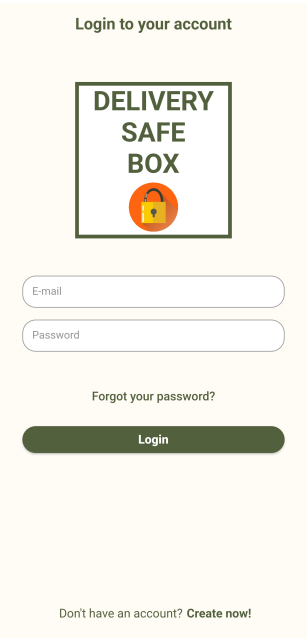

Figure 9: Login Screen

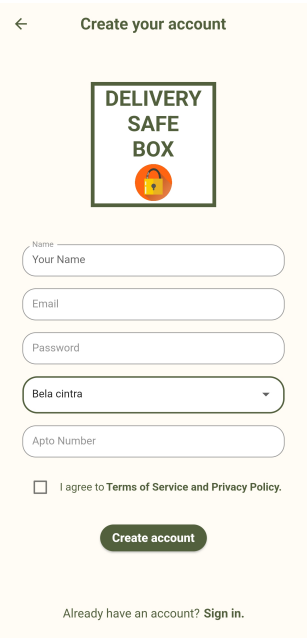

Figure 10: Sign Up Screen

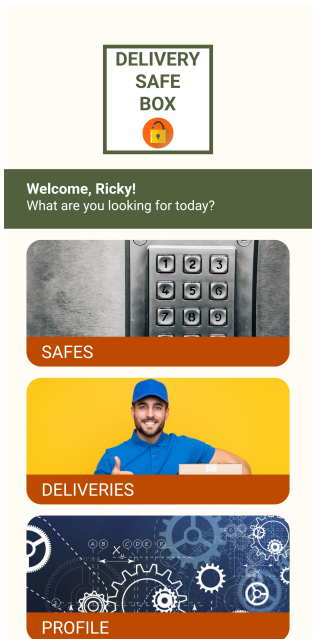

Figure 11: Main Screen

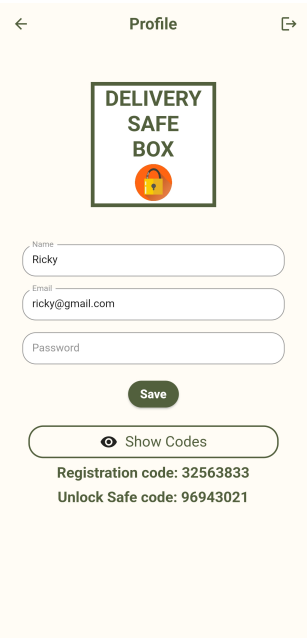

Figure 12: Profile Screen

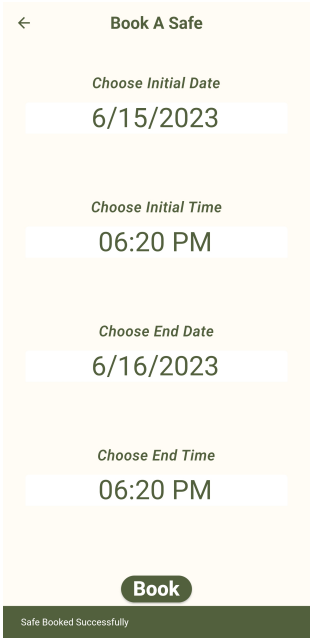

Figure 13: Book a Safe Screen

Figure 14: Safes Screen

3.3.2 Web application

The web application, which was developed with React [4], it's for management purposes, and can only be accessed by administrator users. Through the application, it will be possible to check the existing users in the system, as well as authorize them to use the mobile application.

Moreover, it will be possible to access the vault dashboard screen, which contains information about the vaults in the condominium, for example whether the vault is active and occupied.

Furthermore, there is the audit page, in which the admin user can check all the actions performed in the system, by which user and when this action was performed, for the purpose of control, security and management of the system.

Finally, the page for the generation of fine reports, which generates a PDF file containing the users who have used the safe for longer than the reserved time and the amount that each one should pay (in which the charge will be inserted in the condominium). Figure 15 shows the summarized flowchart of the web application. Finally, the web application is hosted in Render cloud server [5].

3.3.3 Backend application

The backend application, developed in Node.js [6] is running in a cloud server called Render [5] together with the PostgreSQL 12 database [7]. The application has the purpose of integrate each control system with the mobile application

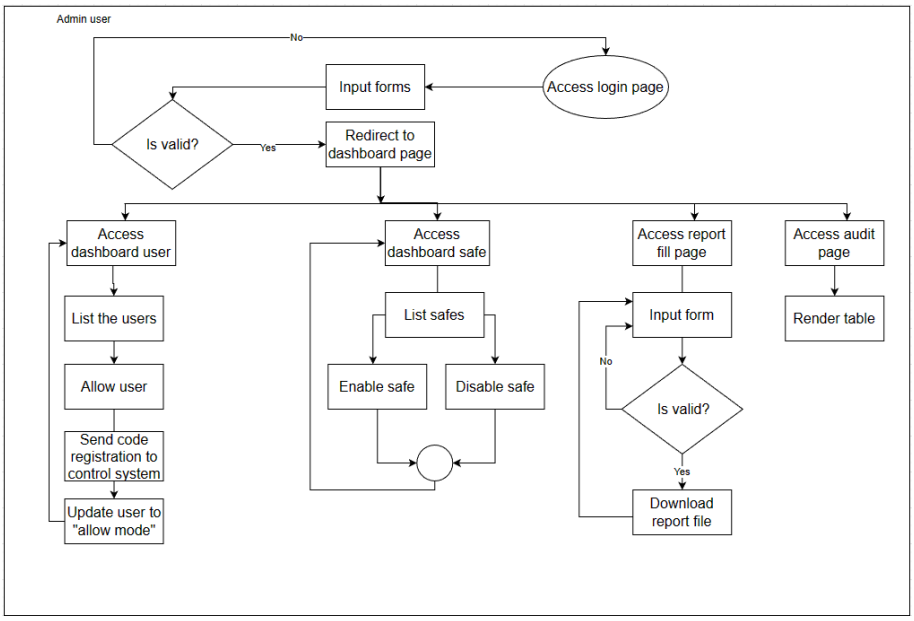

Figure 15: Web application flowchart diagram

and with the web application. So, the backend software manages all the users of the system, receives the requests to book a safe and create an order from the mobile application, generate the codes to the delivery, the registration face and fingerprint process and send these codes to both mobile application and control system.

In addition, the server receives from control system information like the package of the specific resident user was deliveried or withdraw or that a vault is being damaged, and send to the required users a push notification to mobile application using Firebase Cloud Messaging (FCM) [8].

3.3.4 Communication between the softwares

The communication between the mobile application and backend application and between the control system and backend application are executed using HTTP requests with JWT token to the authetication process and all the end-points are in the Node.js system.

Moreover, the communication between the control system (Raspberry Pi microcontroller) and safe system (ESP C3 microcontroller) is with UART (universal asynchronous receiver/transmitter) with a simplified implementation of HDLC (High-Level Data Link Control). The Raspberry requests the activation or reading of the peripherals and ESP32 sends a message to Raspberry when the alarm is triggered.

3.3.5 Recognition process

In the project, two methods for recognizing a resident user were addressed, which are through face and through fingerprint. For both methods, code was used that was created using Python and is running on the Raspberry Pi.

3.3.6 Facial recognition

For face recognition, both the openCV [9] library and the face_recognition api [10] were used, figure 16 contains a summary of the flow of how the face recognition process occurs.

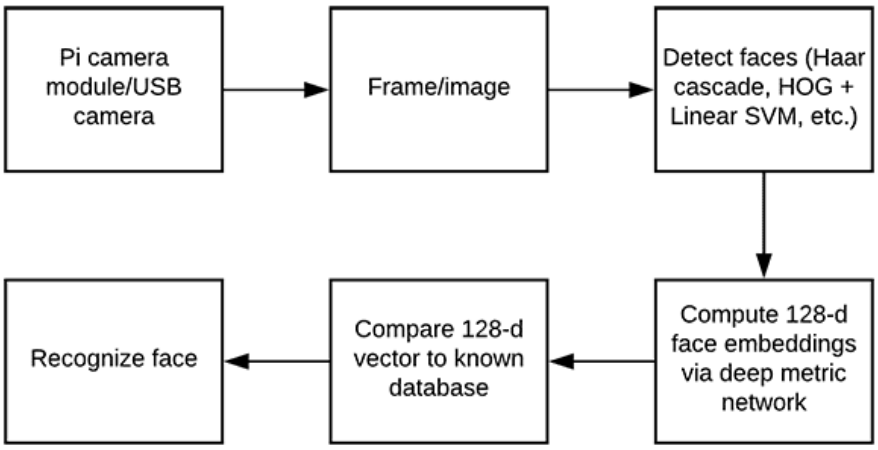

Figure 16: Face recognition flowchart diagram

Therefore, as can be seen in figure 16, a frame is first taken and face detection is performed, in order to optimize face recognition with only faces present.

Subsequently, a 128-d vector is computed, which represents the gradient of the image that shows from the flow of the lightest to the darkest pixels, thus representing the encoding of the person's face.

As there is the process of registering the user's face in the system, in addition to the photo that is stored locally on the Raspberry Pi, the 128-d vector is also stored. Thus, by comparing the 128-d vectors the software is able to recognize a person's face during the face recognition process.

At the time of face recognition the 128-d vector of the current person is computed and compared with all 128-d vectors already stored.

The fingerprint recognition process is a similar process, in which the fingerprint recognition module stores the photo, also creating a structure to represent the fingerprint, performing the comparisons by the structure created and stored in the module itself. In the development of the fingerprint software the Adafruit lib was used[11].

Table 1: Project budget

Product	Qtd.	Total price
Leac Acid Battery 12V	1	R$70,00
Microcontroller Raspberry pi 3B+	1	R$450,00
Microcontroller ESP32-c3-mini	2	R$30,00
USB Numeric keyboard	1	R$34,99
Display LCD 16x2	1	R$15,68
Webcam Logitech c270	1	R$130,00
Fingerprint module FPM10A	1	R$52,30
Sound alarm 12V	1	R$23,42
Accelerometer and gyroscope MPU-6050 Gy-521	1	R$14,98
Load cell 3 wires - 50Kg	1	R$9,50
Solenoid lock Morelle GM2 12V	1	R$60,00
Wood safe	1	R$200,00
Miscellaneous	*	R$280,37,00
Total	*	R$1371,24,00

4 Results

4.1 Budget

Table 1 contains the list of materials used to build the safebox. Initially, the budget foreseen in the project overview was R$ 1072,74, however, R$ 1271,24 were spent, thus, an increase of 18% in the foreseen budget. However, the project contains 100 reais for additional components for risks.

Miscellaneous contains electronic materials for the battery system, as well as materials for making the system's primary safebox and the control system.

4.2 Schedule

The development time for the project was 7 weeks and more two weeks to the final report. Table 2 shows an overview of the entire schedule, noting that total hours contains some tasks that were added during the development. The estimated hours include the 30% extra hours and was determinated during the project planning. Thus, there was a 30% increase in hours when comparing the quantity worked with the estimated quantity (considering the 30% margin of error). The full sheet can be viewed in the footlink below. [2]

[2]Schedule: https://frankbloemer.notion.site/13b92bbf4f584d00b24cc657eee877b9?v=7bfa4623ad6a4aebb9af70805a932d97

Table 2: Project schedule.

Week	Estimated hours	Worked hours
Week 2 - Blog	40	70
Week 3 - Mechanical	65,5	95,7
Week 4 - Hardware	52	93,5
Week 5 - Software	64	58
Week 6 - Integration I	40,5	55
Week 7 - Integration II	60,5	120
Week 8 - Final integration	72,5	118
Week 9 - Final Report - Part I	35	35
Week 10 - Final Report - Presentation	25	25
Total estimated hours	455	670,2
Total estimated hours + 30% error	516,75	

5 Conclusions

With the development of this project, it was possible to obtain several interesting conclusions and learnings. To begin with, developing extensive projects in teams always involves technical and interpersonal challenges. Conflicts of schedule, time, and friction are inevitable. In these moments, the importance of good, flexible, and realistic planning becomes even more evident. In addition, clear, honest communication and good leadership are essential to keep the team united and in harmony, focused on the objectives and completing the project with the planned quality, time, and budget.

In addition, the help and cooperation among the members further reinforces the synergy and learning for a successful project. In several moments, it was possible to experience difficulties due to lack of knowledge, experience, or unforeseen events in the development of the project. A good action plan and well-done risk assessment were essential to have backup plans, as well as to save time and not lock the development with unplanned blocks.

In the case of the Safe Delivery Box project, the development proved to be much more complex than initially foreseen. In the planning phase, much less time was estimated to accomplish the tasks than what was actually given, mainly due to lack of knowledge and unforeseen blocks. Given this, the final result of time spent was much longer than planned. However, the budget estimate was very coherent, and was not extrapolated by a large margin.

Unforeseen problems and blockages occurred mainly in the integration phase as previously planned, with time allotted for bug fixing and final adjustments. The assembly of the mechanical structure occurred in the first weeks, proving to be a challenge for students in a course where there is not such a strong focus on mechanics. The hardware and software development also had its challenges, but it was possible to keep up the pace without major delays during the course's

deliveries and partial weekly presentations.

Finally, it was possible to deliver the project as initially planned, with many learnings, hits and misses along the way. We leave the course with much more hands-on knowledge than when we started at the beginning of the semester, and with the certainty that the project developed here presents interesting perspectives for future projects, in an increasingly digital and asynchronous world.

References

[1] Medição de força com o Arduino e um módulo HX711. https://www.aranacorp.com/pt/medicao-de-forca-com-o-arduino-e-um-modulo-hx711/#:~:text=PrincÃηpio%20de%20funcionamento,Ãl'%20aplicada%20num%20sinal%20analÃşgico.

[2] Acelerômetro: como funciona e como usar? https://www.electricalelibrary.com/2022/04/29/acelerometro-como-funciona-e-como-usar/.

[3] Flutter documentation. https://docs.flutter.dev/.

[4] React documentation. https://legacy.reactjs.org/docs/getting-started.html.

[5] Render Cloud Application Hosting for Developers. https://render.com/.

[6] Node documentation. https://nodejs.org/en/docs.

[7] PostgreSQL Documentation. https://www.postgresql.org/docs/.

[8] Configurar um app cliente do Firebase Cloud Messaging no Android. https://firebase.google.com/docs/cloud-messaging/flutter/client?hl=pt-br.

[9] OpenCV-Python Tutorials. https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html.

[10] Raspberry Pi Face Recognition. https://pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/.

[11] Adafruit-Fingerprint-Sensor-Library. https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library.