Federal University of Technology - Paraná - Brasil – UTFPR
Academic Department of Electronics – DAELN
Academic Department of Informatics – DAINF
Computer Engineering
Integration Workshop 3 (EEX23) – S71 – 2021/1

Technical Report
ElectionPipa

Alexandre N. Bonacim – alebonacim@hotmail.com
Gabriel K. Brylkowski – gabrielkuhnenb@gmail.com
Juliana R. Viscenheski – jviscenheski@alunos.utfpr.edu.br
Maria Fernanda Azolin – azolin.mf@gmail.com
Mateus V. Freitas – mateusvfreitas@gmail.com
Rafael H. Ramos – ramosrafh@gmail.com

August, 2021

Abstract

This document reports the devolpment of ElectionPipa, the project of a smarter paper-based election system. The proposal revolves around getting rid of most of the manual work that is required to hold an election in small instances – such as a school board election or a neighborhood council –, both for preparing the voting time and for accounting the results afterwards. ElectionPipa consists of an electronic system that can identify the voters by their fingerprint and a counting system that is based on image recognition. Alongside all that, a web application is used to show stats of the election during the voting time and the results after, while a mobile application is used to setup the voters and the schedule beforehand.

1 Introduction

Elections have been present in society for quite a while now. Most organizations, institutes or even small groups usually have some form of choosing a person to be their representative, but, in most cases, those elections take quite a while to be held, either because of the time necessary to count the votes afterwards or due to the long time it takes to verify the voters identities to make sure that no fraud happens.

With that in mind, creating a paper-based election system that can guarantee the unique identity of every voter and eliminate the manual work of counting – or even recounting, if necessary – the results is a good way to optimize the elections in small instances, such as college boards, student councils or even neighbourhood syndicates.

1.1 Requirements

To accomplish such goals, a list of functional and non-functional requirements was compiled. These requirements approach every instance of the project, in a way that every module has its requirements specified and the system can achieve its goal. All project's requirements are described from Table 1 through Table 5.

Web Application Functional Requirements	
W-FR01	The web app must display election results after voting time
W-FR02	The web app must display in real time the numbers of voters who have already voted
W-FR03	The web app must display the election schedule
Web Application Non-Functional Requirements	
W-NFR01	The frontend of the web application must be developed using Javascript + HTML + CSS
W-NFR02	The backend of the web application must be developed using NodeJS + Express
W-NFR03	The web server must be hosted at Heroku cloud platform

Table 1: Web application requirements

Mobile Application Functional Requirements	
M-FR01	The mobile app must allow the administrator to manage the candidate list outside voting time
M-FR02	The mobile app must allow the administrator to choose the election date and duration before voting time
M-FR03	The mobile app must allow the administrator to access recount mode in the ballot counter
Mobile Application Non-Functional Requirements	
M-NFR01	The mobile app must be developed in Swift
M-NFR02	The mobile app must read/write data from the database
M-NFR03	The mobile app must run on iOS devices

Table 2: Mobile application requirements

Database Functional Requirements	
D-FR01	The database must be accessible by the other modules through the internet
D-FR02	The database must store the voter's name, unique identifier, password, fingerprint and voting status
D-FR03	The database must store the election schedule
D-FR04	The database must store the candidates
D-FR05	The database must store each vote as ballot identifier and voted candidate
Database Non-Functional Requirements	
D-NFR01	The communication must use a socket-based request-response protocol
D-NFR02	The database must be cloud-based
D-NFR03	The database must be MongoDB based
D-NFR04	The stored data must be documents in JSON (JavaScript Object Notation) format
D-NFR05	The database must not provide correlation between a voter and his vote
D-NFR06	The database must have a backup

Table 3: Database requirements

Authentication Module Functional Requirements	
A-FR01	The authentication module must be able to authenticate the voters through their fingerprint
A-FR02	The authentication module must be able to authenticate the voter through their unique identifier and password as fallback method
A-FR03	The authentication module must display error/success messages when authenticating the voter
A-FR04	The authentication module must provide feedback when failing to authenticate the vote or if the voter has already voted
A-FR05	The authentication module must not authenticate a voter while another one hasn't had their vote validated by the ballot counter
Authentication Module Non-Functional Requirements	
A-NFR01	The authentication module must use the FPM10A fingerprint scanner
A-NFR02	The authentication module must use a USB keypad
A-NFR03	The authentication module must use a 16x2 LCD display
A-NFR04	The authentication module must use green and red LEDs
A-NFR05	The authentication module must swap authentication methods after 3 failed attempts

Table 4: Authentication module requirements

Ballot Counter Functional Requirements	
B-FR01	The lid must open when the voter is authenticated
B-FR02	The lid must remain closed if the voter is not authenticated
B-FR03	The ballot counter must have a conveyor belt
B-FR04	The conveyor belt must be activated after the voter is authenticated
B-FR05	The illumination inside the box must be turned on after the voter is authenticated
B-FR06	The camera must start capturing after the voter is authenticated
B-FR07	The ballot counter must allow recount of votes
B-FR08	The camera must detect the ballot QR code
B-FR09	The camera must identify the vote content
B-FR10	The conveyor belt must be able to spin both ways
B-FR11	The ballot counter must display instructions for inserting the ballot
B-FR12	The ballot counter must display which candidate was recognized in the ballot and wait for the voter confirmation
B-FR13	The ballot counter must allow the voter to confirm that their vote is correct through the use of the keypad

B-FR14	The ballot counter must allow the voter to disconfirm that their vote is correct through the use of the keypad
B-FR15	The ballot counter must store the vote in the box after the confirmation
B-FR16	The ballot counter must eject the vote after the disconfirmation
B-FR17	The ballot counter must wait for the voter to reinsert their vote in case of disconfirmation
B-FR18	The ballot counter must eject the ballot if the ballot identifier was already inserted in the vote count
B-FR19	The ballot counter must eject the ballot if the ballot was inserted with the back side facing up
B-FR20	The ballot counter must require the administrator to authenticate before entering recount mode
B-FR21	The ballot counter while on recounting mode must check for anomalies such as duplicated ballot identifier and identifier that were not accounted for during the voting time
B-FR22	The ballot counter must check the database on startup if an election is currently being held, which indicates that a power outage happened
B-FR23	The ballot counter after identifying a power outage will move the conveyor belt so counted ballots will be thrown in the box and otherwise returned
B-FR24	The ballot counter after identifying a power outage will require a voter to authenticate again
Ballot Counter Non-Functional Requirements	
B-NFR01	The ballot paper must have a ARTag with an identifier and the candidate options on the front side and a ARTag on the back that discerns it as the back side
B-NFR02	The ballot counter conveyor belt must be made using a friction material
B-NFR03	The ballot counter must use a step motor to move the conveyor belt
B-NFR04	The ballot counter must use a RPi Camera
B-NFR05	The ballot counter must have a locked box
B-NFR06	The ballot count must have light bulb inside the box
B-NFR07	The ballot counter must use a servomotor to open/close the box
B-NFR08	The microcontroller used must be the Raspberry Pi 3B
B-NFR09	The ballot counter must a 16x2 LCD display
B-NFR10	The ARTag must be generated using the OpenCV library
B-NFR11	The box must be built with cardboard
B-NFR12	The box must have approximately 10x20x30cm dimensions
B-NFR13	The box must have a lid with the same material as the box
B-NFR14	The lid must be controlled by the servomotor

B-NFR15	The ballot paper must be marked with a pen by the voter
B-NFR16	The ballot counter must wait up to 2 minutes for the voter to reinsert their ballot in case it was ejected, after the time has passed it returns to the authentication step

Table 5: Ballot counter requirements

1.2 Project overview

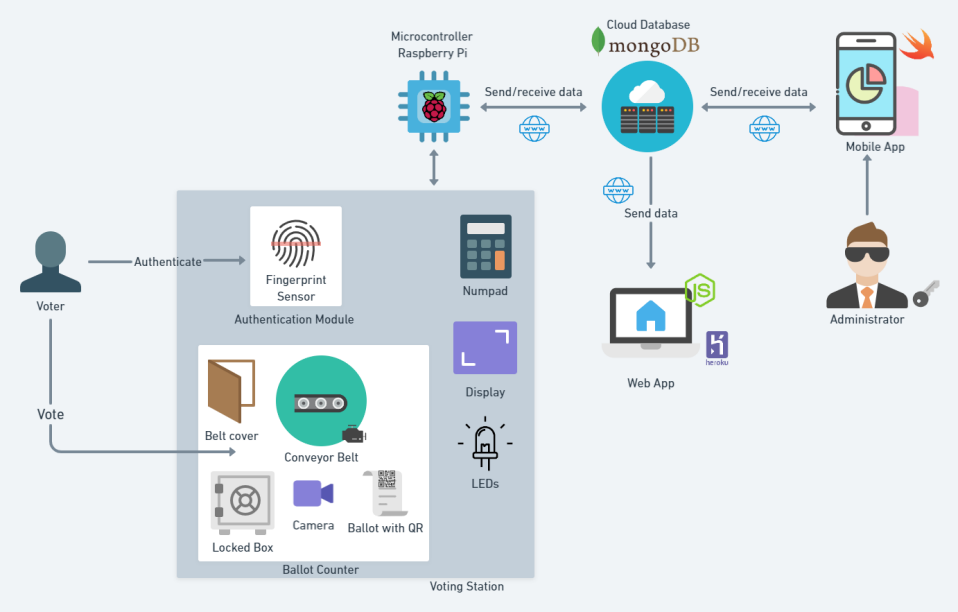

Figure 1: System overview

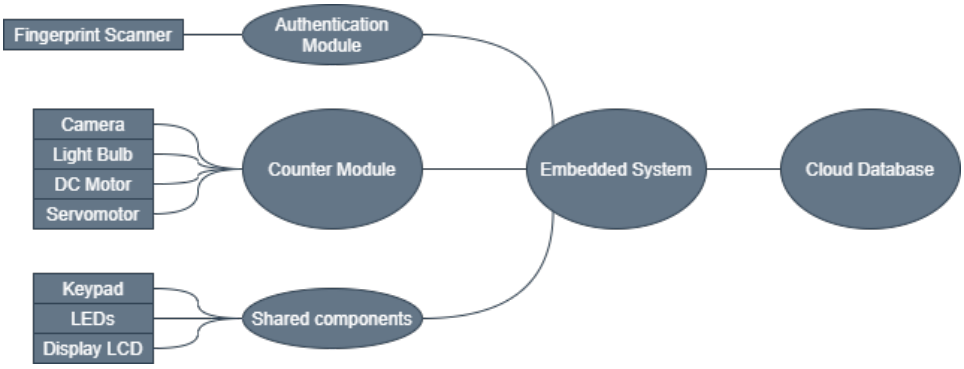

Figure 2: Subsystems overview

With all the requirements defined it is possible to create a block diagram of

the solution, as well as diagram of the project's subsystems, as shown in Figures 1 and 2.

The initial solution design consists mainly of a voting station, a cloud server, a web application and a mobile app. The voting station has many features: first, it has a fingerprint sensor attached to it, which the voters use to authenticate themselves during an election, so they can vote. Also, there is a numpad whose purpose is to be a fallback method in case a voter can't be authenticated with his finger – due to bad fingerprints or any other issue –, where every voter has a unique ID and password to be used in that case. The numpad is used to confirm or cancel the vote too.

The ballot counter is the main part of the voting station. Inside of it there is a camera and a conveyor belt: as the voter inserts the paper vote, the paper rolls on the conveyor belt, while the camera tries to detect the chosen candidate. The counter also has LEDs and a LCD display to give the user feedback on his actions, such as confirm detected vote, error with the inserted paper and so on.

The cloud database is where the candidates, the election schedule and the voter's information is stored. All these informations are necessary for the election to take place. The mobile app is used to create and send this data to the server, while the web application works as an election follow-up, showing the schedule, the real time statistics and, after voting time, the results.

2 Technologies

2.1 Fingerprint Recognition

The project uses a fingerprint optical sensor for the authentication system, model FPM10A [1]. The sensor generates an 8 bit grayscale image from the voter's finger and then extract the template from it.

The template is a set of features such as cores, deltas and minutiaes found on a fingerprint. Cores are points near the center of the finger that usually have a swirl around, deltas are points in a loop or whorl that usually have a triangular shape diverging parallel ridges and minutiae are the points where an ridge ends or bifurcates.

After the template extraction the sensor save it in a FLASH memory and assigns an unique identifier to the registered fingerprint. This unique identifier is what the team uses to assign a fingerprint to a enrolled voter. The returned id in the enrolled process is associated, manually, to the voter in the database collection.

During the authentication step the sensor search in the saved templates if there's a match for the finger on the sensor and, if it finds one, it'll return the unique identifier assigned to the found template. We check the database to find a voter with that ID so they can be authenticated.

2.2 Cloud Database

MongoDB [2] is an object-oriented, simple, dynamic, and scalable NoSQL database. It is based on the NoSQL document store model. The data objects are stored as separate documents inside a collection — instead of storing the data into the columns and rows of a traditional relational database. The motivation of the MongoDB usage is to implement a data store that provides high performance, high availability, and automatic scaling.

2.3 Mobile App

The mobile app was built using the Swift language, and can run in any iOS platform. To structure the project, the MVC architecture (Model View Controller) was chosen.

The RealmSwift framework [3] was used to build the communication between the app and the MongoDB server, to where all data was sent and read from.

2.4 Web Application

To build the web application of the project, it was used the NodeJS runtime environment [4]. It is an open-source, cross-platform, back-end JavaScript runtime environment that executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

Alongside with NodeJS, it was used the Express framework [5], which is a back-end web application framework for NodeJS. It is designed for building web applications and APIs. For the front-end experience, it was used EJS, which is a simple templating language that lets generate HTML markup with plain JavaScript.

Heroku is a container-based cloud Platform as a Service (PaaS) [6]. It is used to deploy, manage, and scale modern apps. This technology was chosen for being flexible and easy to use, offering the simplest path to deploying the web application publicly.

3 Development

3.1 Voting Station

The voting station consists of a reinforced cardboard box that holds all the sensors and actuators involved in the project: the conveyor belt, built out of LEGO pieces and controlled via stepper motor, the gate servomotor, the fingerprint sensor, the Raspberry Pi camera, the LCD display, the LEDs, the Raspberry Pi Model 3B+ microprocessor and the 5V power supply.

3.1.1 Mechanical Structure

The mechanical structure was modelled using FreeCAD, and it was the first contact of the group with 3D modelling. In it, the group could simulate the LEGO pieces used to build the conveyor belt and draw the box, as seen in Figures 3 - 5.

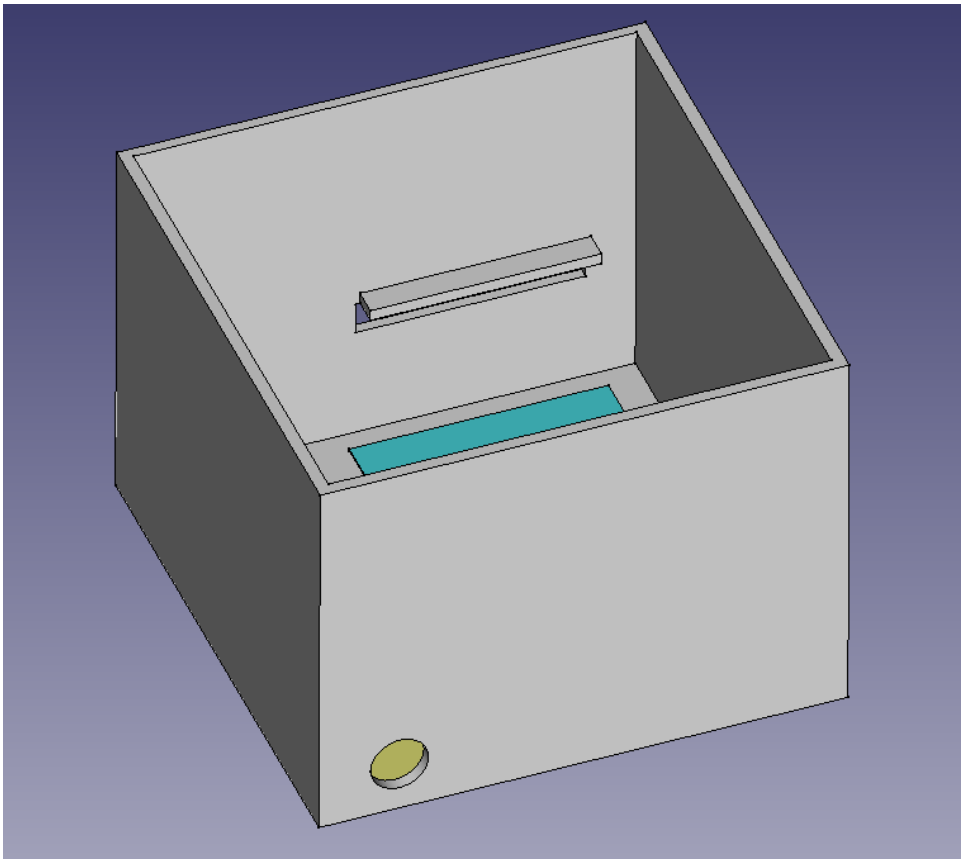

Figure 3: Box - back view

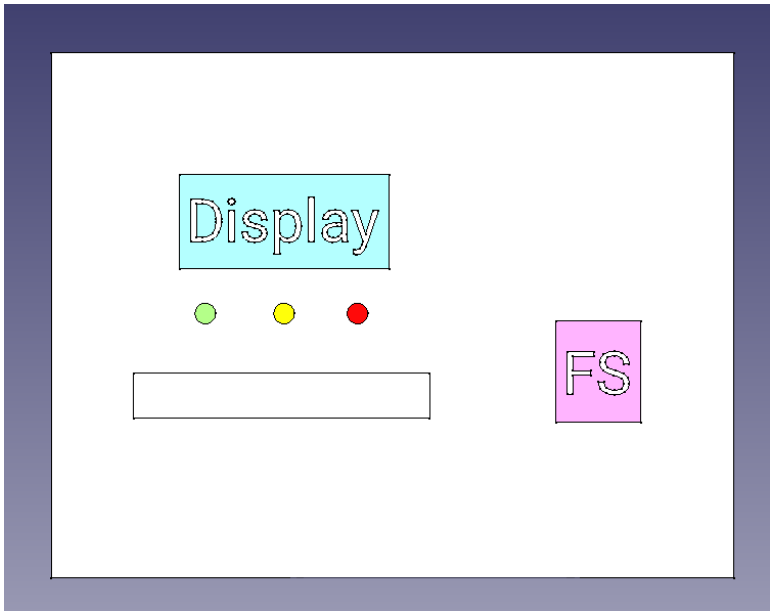

Figure 4: Box - front view

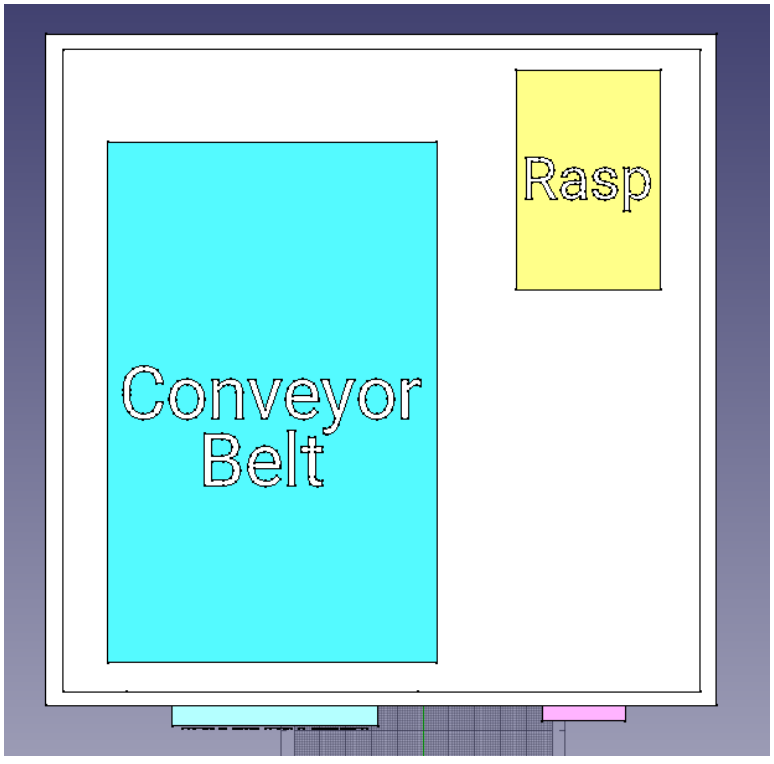

Figure 5: Box - top view

With all the diagrams finished, began the construction of the fundamental elements of the project. First of all, the conveyor belt was built and the motor integrated to the system, and the first tests were made to assure the system would work, as seen in Figures 6 and 7. An important decision was made about the materials of the conveyor belt: as the ballots are lightweight and easy to transport, the belt was built out of rubber bands instead of a file or other kinds of friction material, as the rubber bands provided enough friction to keep the paper in movement once the vote detection process began.

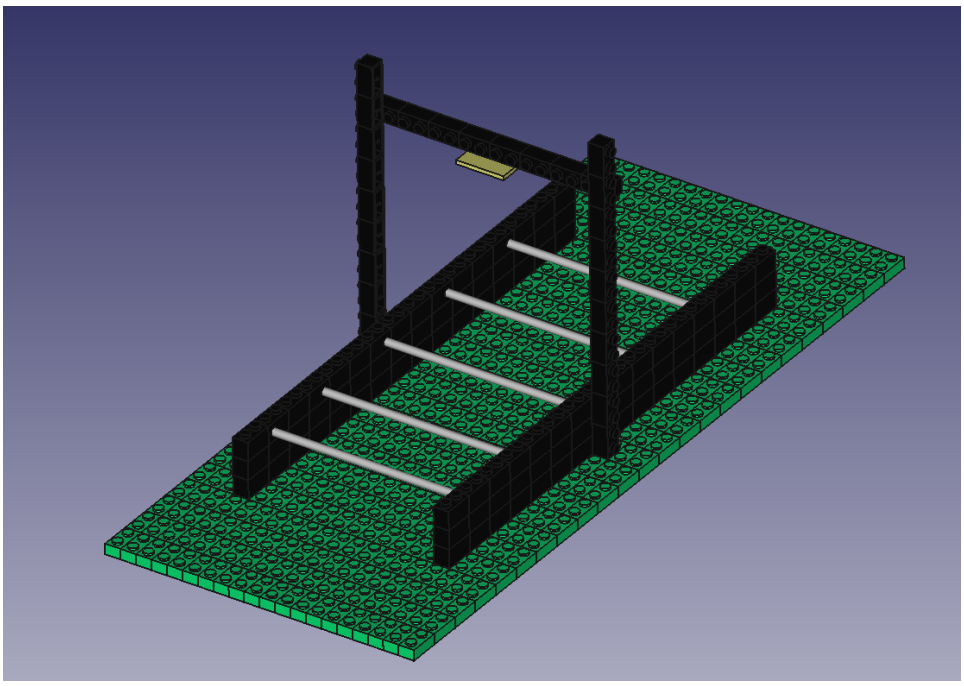

Figure 6: Conveyor belt project

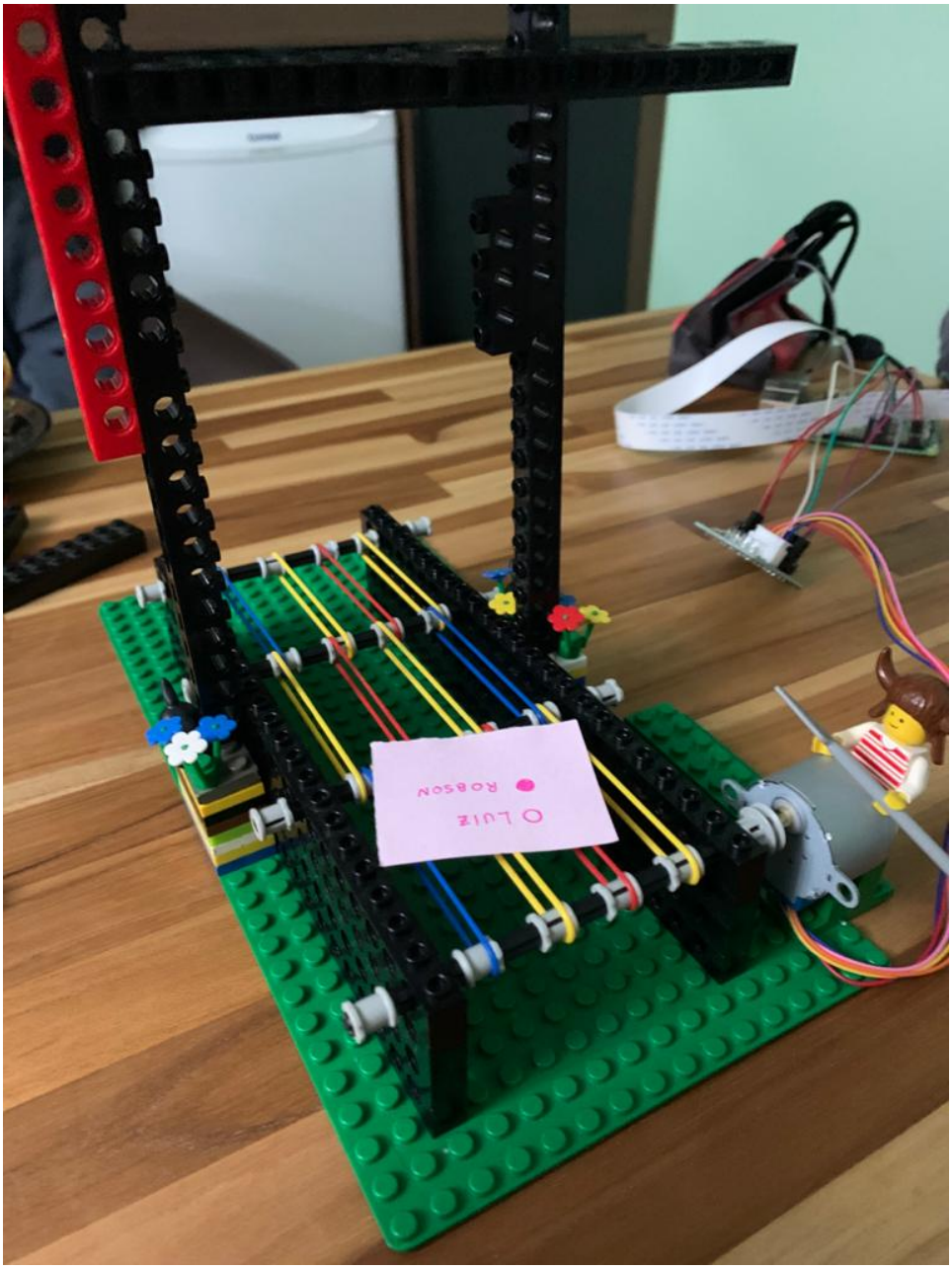

Figure 7: Conveyor belt built of LEGO pieces

The voting station was designed to be robust but portable, and to contain all the elements required for its correct function. To attain all objectives and due to ease of manipulation, the group used reinforced cardboard covered in contact paper with enough room to fit all components mentioned beforehand to build a box.

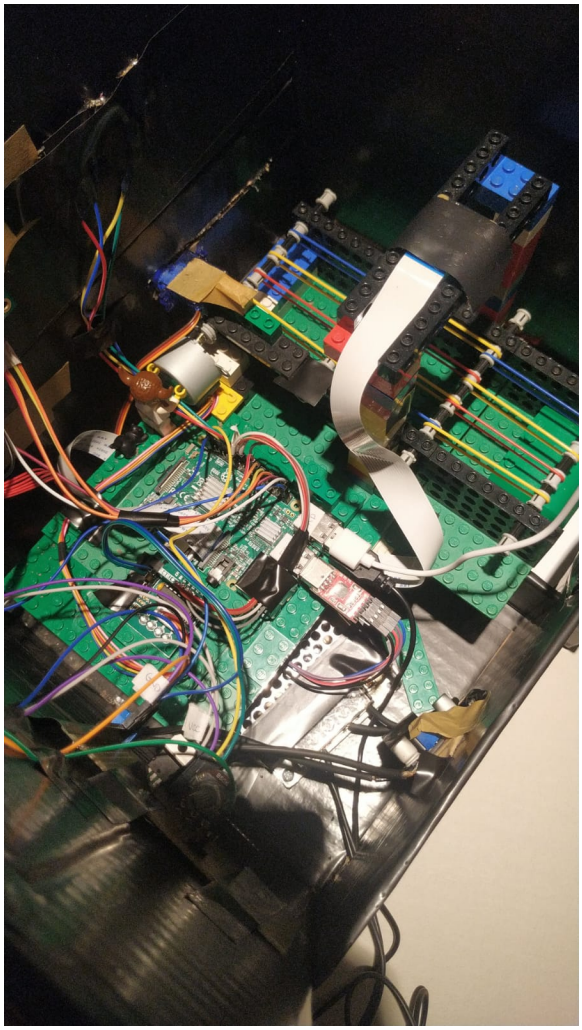

Figure 8: Built box - inside view

Figure 9: Built box - outside view

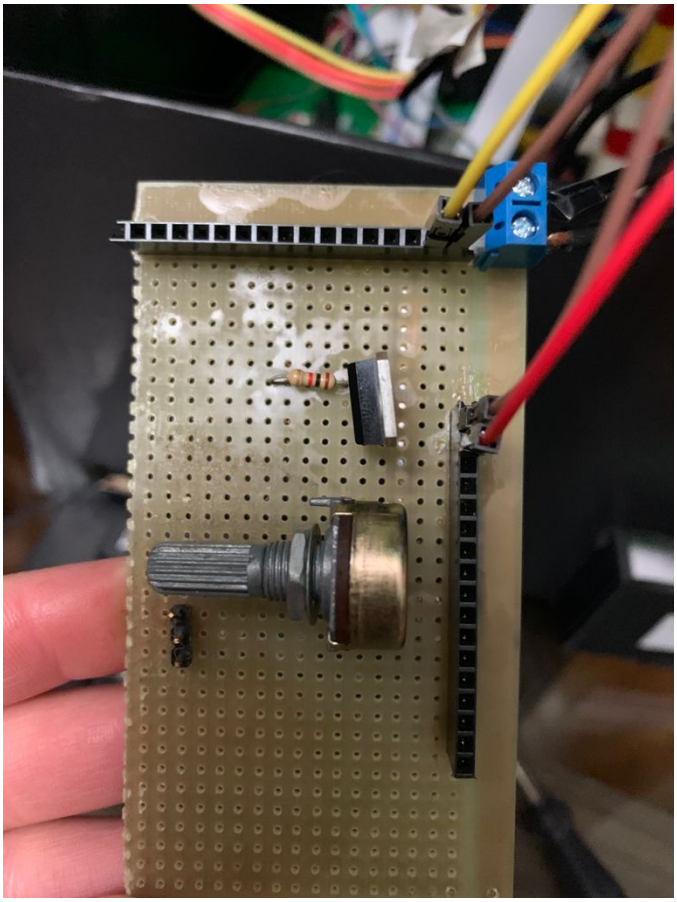

Figure 10: Built box - outside view

3.1.2 Hardware Project

For the hardware project and development, the diagram in Figure 11 was made using TinkerCAD [7], to verify if there would be enough General Purpose Input/Output ports for every functionality. There was also a need to use USB ports for the fingerprint sensor, for the numpad and for the ring light used to assist the camera in detecting the ballot contents.

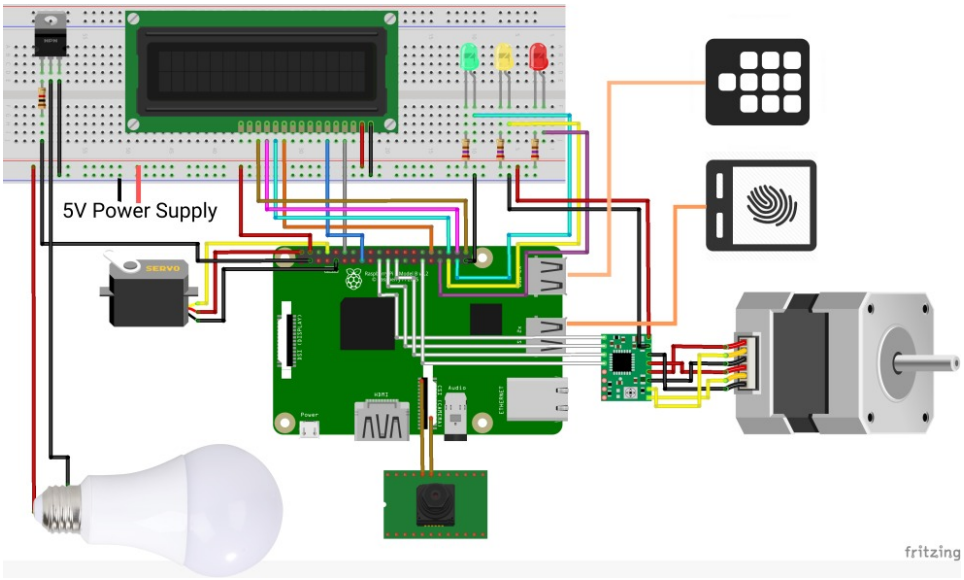

Figure 11: Hardware project

For the power supply distribution, a printed circuit board, as shown on Figure 10, was developed to join all positive and ground supplies in a single point. In this board it is also possible to control the LCD contrast. All the wires were split up according to their respective functions.

The stepper motor was set to half step mode and provided adequate torque to move the conveyor belt in both directions. The LCD display worked according to plan, but the downside is the small size. It is not possible to display a lot of information at a given time and some messages were split in two or more phrases. The 12V 5A power supply used is able to supply enough current for all hardware components without risk of an internal problem.

3.2 Cloud Database

The modeling and creation of the cloud database had to take two important things in consideration. First, due to the need of keeping the voter's identity anonymous, the database was projected in a way that no vote would be linked to any voter. Therefore, the only information from the vote that is stored on the cloud is its unique ID and the voted candidate selected on the paper.

The second important thing was making sure that the votes would only be sent to the database after the election was completed, to prevent any kind of fraud or improper access to the results before every vote was in fact, computed and the election finished. Until then the votes are stored in a local file to prevent being lost due to a power failure.

With all those issues in mind, the database was set up, according to Figure 12. It includes six collections: one for the administrators, which storages the

finger ID of the administrators; one for the election, including its schedule, one for the votes, with their chosen candidate, the paper ID and the type of the vote (valid, blank or null). Next there are the voters and the candidates collections, storing the relevant information from those two groups; and a recount collection, in case a recount is asked for.

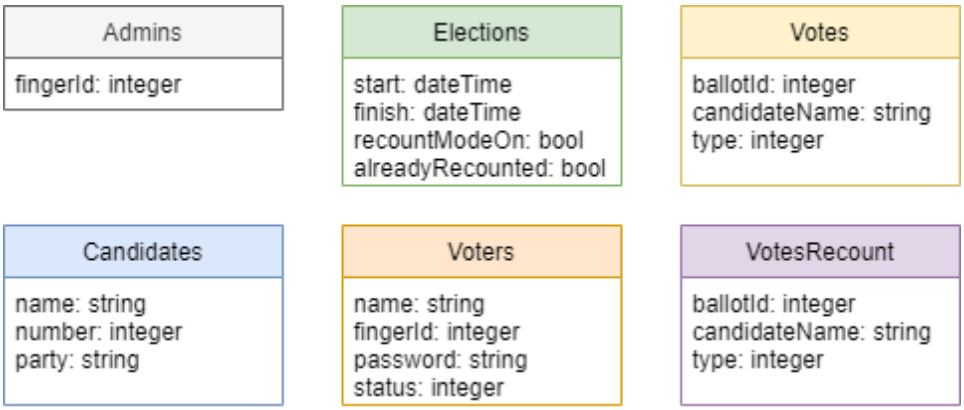

Figure 12: Database model

3.3 Mobile Application

The first step of the mobile application development was creating the UI/UX design, for which was used the Skecth desktop software [8]. Sketch is a powerful tool to prototype screens and build app mockups. Alongside with the screens, the color palette and font style of the project were decided, as seen on Figure 13

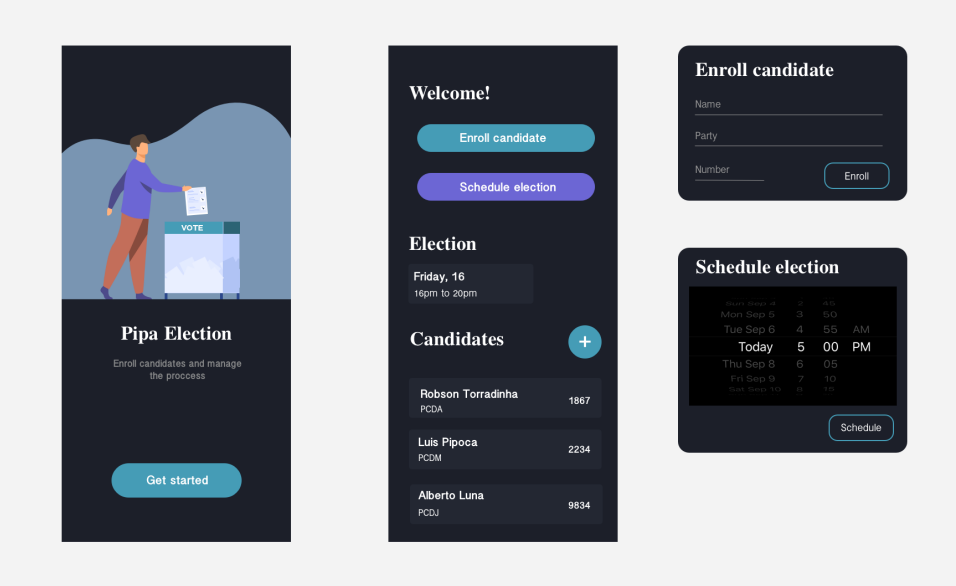

Figure 13: IOS app design

Knowing what the app would look like and what screens we would have, it was possible to build the class diagram, as shown on Figure 14. For this step, the draw.io platform [9] was used.

Then, the coding part started, and the XCode IDE [10] was used to build the iOS application with Swift. First, the visual part (screens and flows) of the app were developed. With that, it was possible to create the models and classes needed, and then finally implement the connection with the MongoDB [2] server. This was the most challenging part, as the app needed to be able to read and write data to the database to correctly enroll candidates, schedule elections and enable recount mode. Despite that, MongoDB [2] provides the RealmSwift framework [3] to ease this connection between the application and the server, and therefore it was possible to establish a realm time communication between the two with not so much effort.

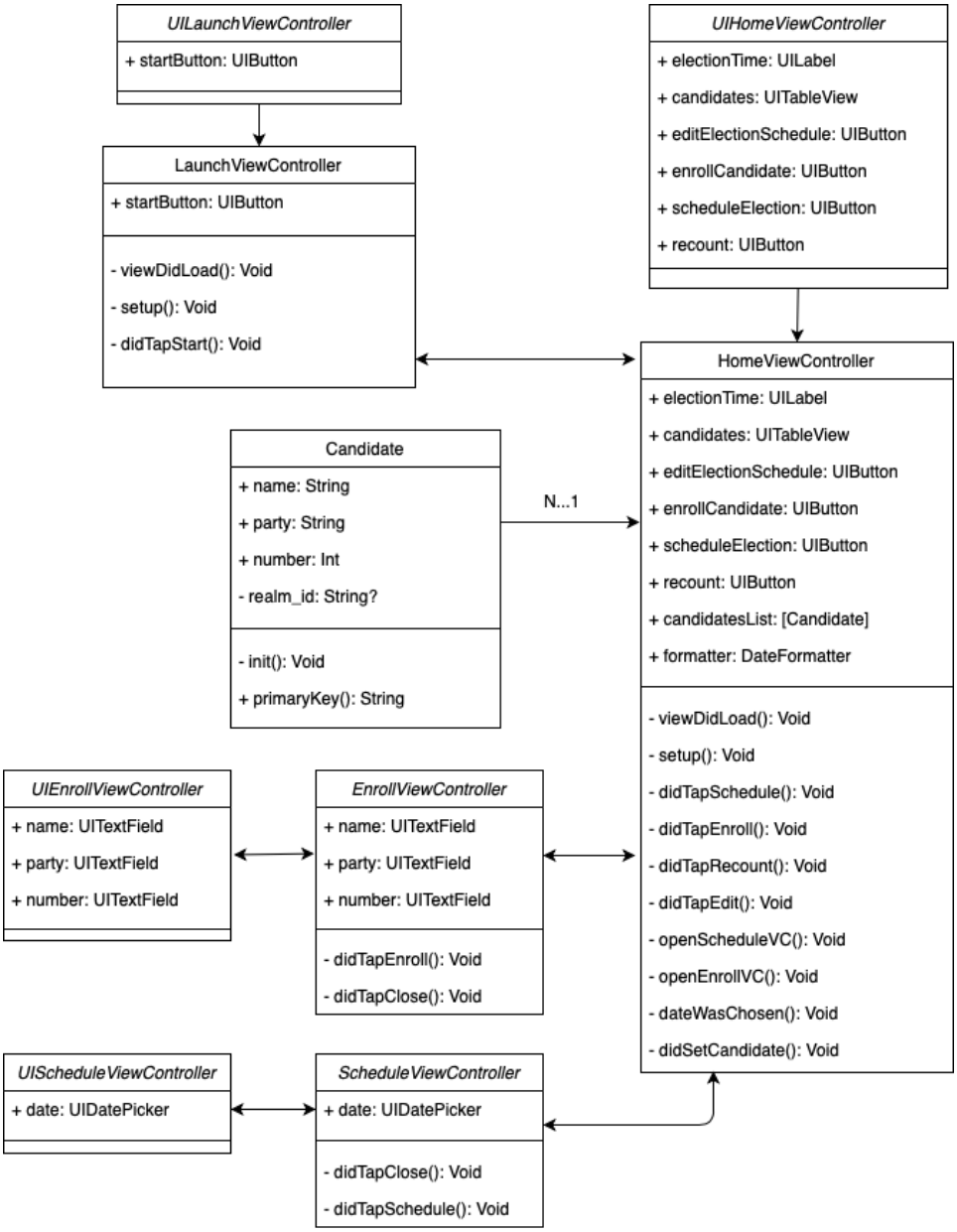

Figure 14: IOS app class diagram

3.4 Web Application

The first step of the web application development was the prototyping of the interface. It needed to be as user-friendly as possible, so the users were able to navigate through the website without any struggle. To make that possible, the front-end design and the user interface were made in a browser-based UI and UX design application.

Next, with the design completed, the front-end of the application was created with EJS. During this step, the main concern was to be as faithful as possible to the designed prototype. With this task concluded, it was possible to start the next part of the web development: the creation of the routes, linking the website to the database, consuming it and displaying it on the browser, and finally deploying the application.

Since NodeJS was the chosen technology for the development of the backend, it was very simple to connect the application to the database, as MongoDB has a built in feature for Node applications. With the connection in the picture, it was possible to seed the application with live data from the database, which then would be displayed on the browser. For that to be possible, the routes were developed so that each server request got the appropriate response – for each functionality of the application. The screens of the web application can be seen on Figures 15 through 19.

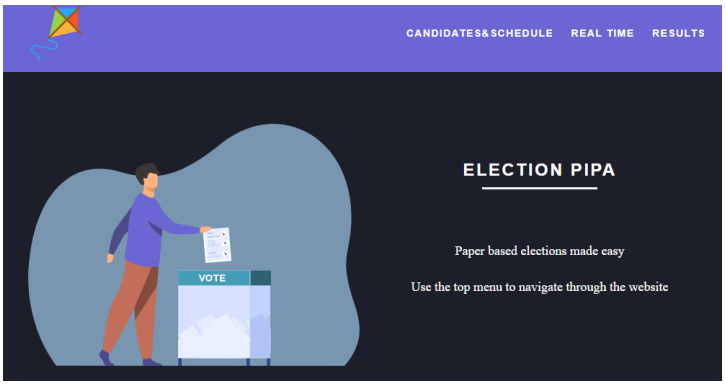

Figure 15: Web application home page

Figure 16: Web application schedule page

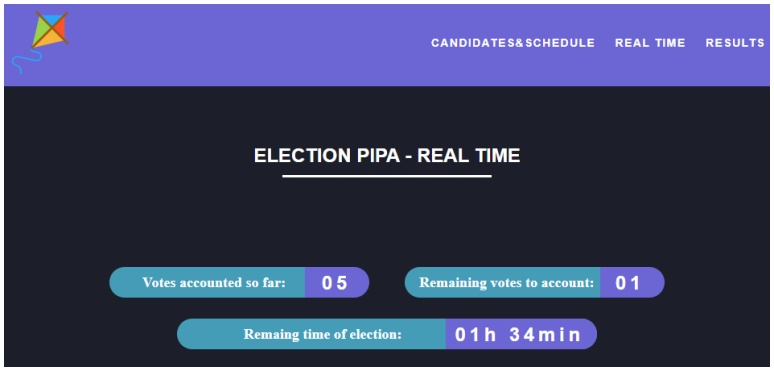

Figure 17: Web application real time page

Figure 18: Web application real time error page

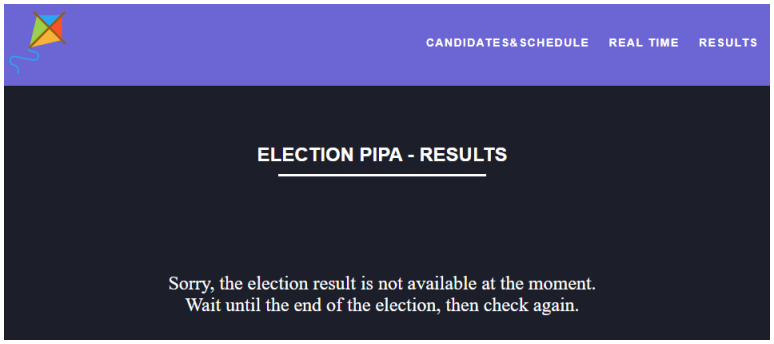

Figure 19: Web application results error page

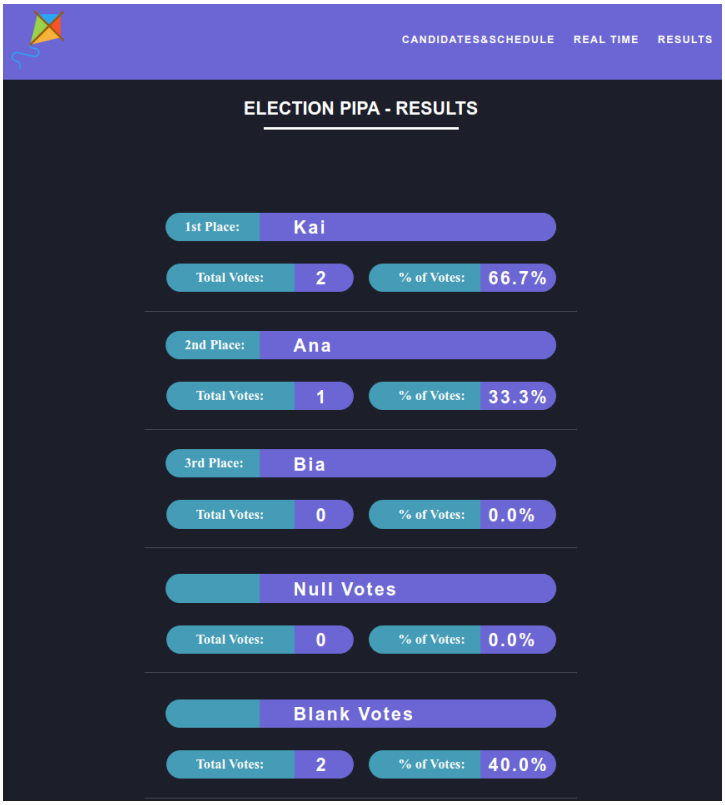

Figure 20: Web application results page

Finally, with all working together, front-end and back-end, it was possible to deploy the application so that anyone could access the application on the browser, through the correct URL.

3.5 Vote Detection

For the voting detection, image processing was used to identify the correct voter choice. In order to have success on this task, two approaches were made: The Red Bar Method and the Augmented Reality Method.

3.5.1 Red Bar Method

The first idea was called “Red Bar Method”. The first thing to do was the design of the ballot with the candidates, the QR Code identification of the ballot and, in this case, with a red bar as seen on the Figure 21.

After the design, the solution was implemented and tested. During the tests the following problem appeared: “How to know the position of the chosen candidate label in different angles?” So to solve it the process was basically:

1. Detect all the circles centroids.

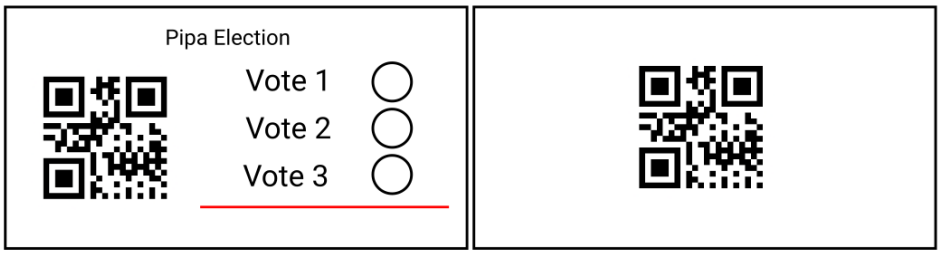

Figure 21: Ballot design

2. With the centroids and radius, the filled circle or circles (if there is any) was detected too using the most predominant color in the circle bounding-box.
3. Detect the red bar (or part of it) and it's center.
4. Calculate and sort the distances of the centroids to the bar since the distance won't change with the angle.
5. Check the position of the filled circle in the sorted circles list.

The parts 1, 3 and 4 of the process can be seen on Figure 22.

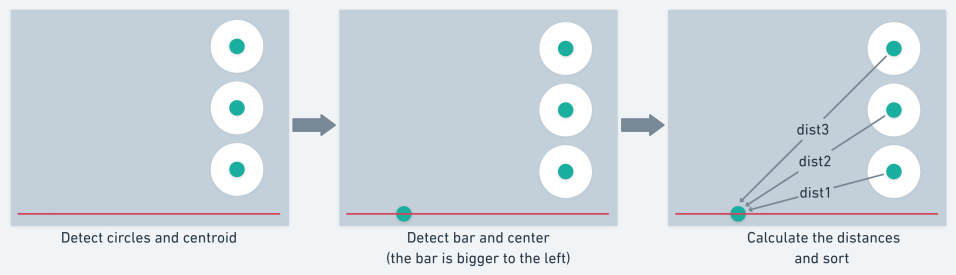

Figure 22: Vote position solution

Circle detection For the circle detection, a variant of the Hough Gradient Method, which is a feature extraction method for shapes [11], for circle detection was used. This implementation was made using OpenCV [12]. Since this method is really sensitive and can be computationally expensive even with small details, the team made some pre-processing. First the image was converted to gray-scale so OpenCV implementation can use it and then the image was slightly blurried using median blur so the image is smoother and since the circle is filled with regular pen it avoid some false-positives that could occur.

Filled circle detection After the circles detection, the algorithm have the centroids and the radius of each one, so using the circle bounding-box and Numpy [13] the team was able to detect the most predominant color inside this bounding-box, with black as predominant color meaning it's filled, otherwise if white is the predominant color the circle is not filled.

Bar detection In order to detect the bar, the team used the HSV color space and made a mask to avoid multiple false-positives of bars and then applied the variant of Hough Gradient Method for line detection.

After each part development, the whole process can be seen on Figure 23

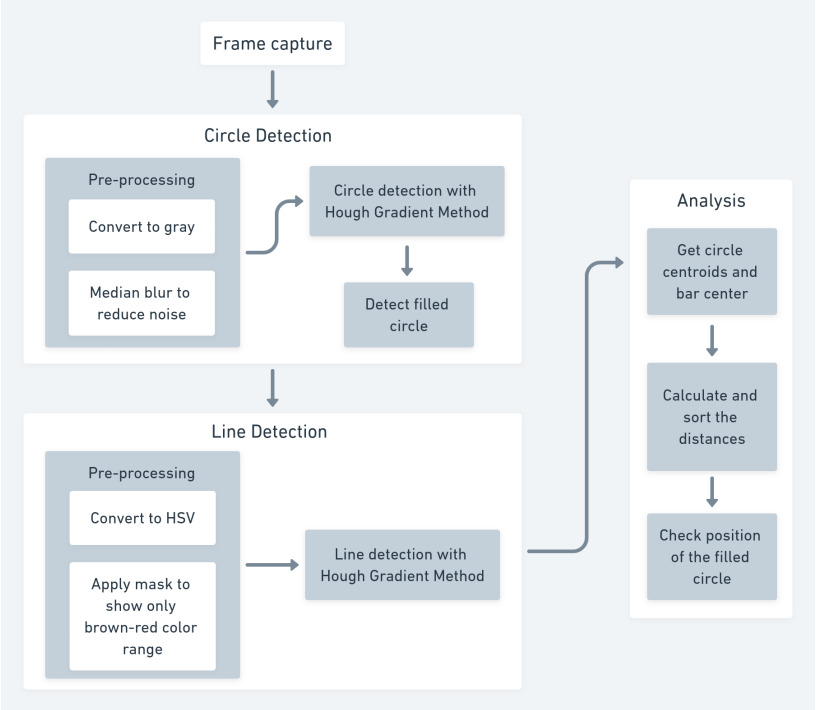

Figure 23: Red bar approach process

3.5.2 Augmented Reality Method

The red bar method worked, but had some disadvantages. The main disadvantages are the amount of processing to do it in real time on the Raspberry Pi, the need of a perfect white lightning to detect and the need of a specifically red bar to do the detection, so if the printer is missing red color the election is in danger. With those drawbacks, the team decided to spend some time to try this AR method. After professors suggestion and a lot of research, the ArUco [14] seemed to be exactly what the project needed to be simpler and faster.

In order to make it work, the first thing to do after thinking about the problem and realizing the steps needed, was to remake the ballot design, now without the red bar and replacing the QR Code with the AR Tag, as can be seen on Figure 24.

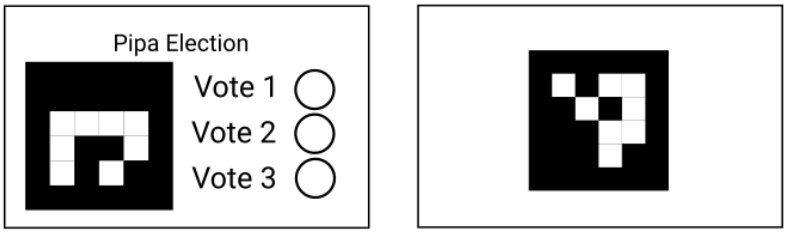

Figure 24: Redesigned ballot with AR Tag

The main advantage of the use of augmented reality on the project was the ability to get the tri-dimensional axis (Figure 25) and the rotation of the tag in relation to the camera, so with this information and some processing the algorithm could rotate the image back, in a way that the ballot is always in the correct position no matter the angle.

The process was similar to the previous approach, but some changes were made:

1. Detect the AR Tag and rotate back the image.
2. Detect all the circles centroids.
3. With the centroids and radius, the filled circle or circles (if there is any) was detected too using the most predominant color in the circle bounding-box.
4. Check the centroids in the y-axis and determine the filled circle position.

The circle detection and filled circle detection were exactly like the previous approach, the main difference was the AR Tag detection.

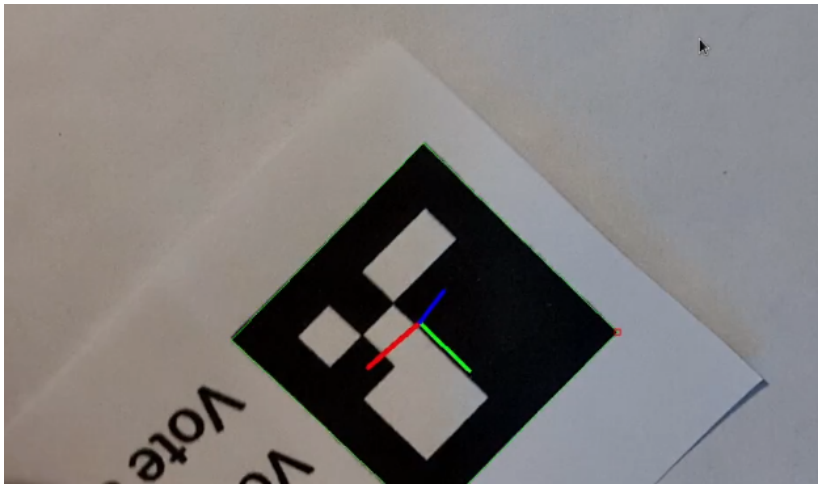

Figure 25: AR Tag detection and axis

AR Tag detection The AR Tag detection wasn't too hard since the OpenCV library already have a great implementation for it, the harder part was to find the rotation in relation to the camera. After some research about this topic, the team had to calibrate the camera to calculate the particular matrix and distortion of the camera used and then, with the rotation matrix, the algorithm could determine the angle in respect to each axis with a really good accuracy (the testing videos can be seen on the project blog [15]).

After some tests, the second approach proved to be really faster, specially because it doesn't need to keep tracking the bar and calculating distances all the time. Another point that helped on the performance was, since the second approach doesn't need the RGB or HSV color space anymore, the gray scale was the chosen one during the whole process, and clearly were more computationally efficient.

With this second approach implementation, the final voting detection process can be seen on Figure 26.

3.6 Power failure recovery

The system implements a state machine in which during state changes the new state is saved on a file. The states are as following: 0 - normal flux, 1 - voter authenticated, 2 - vote confirmed, 3 - recount mode, 4 - recount vote detected. At each startup the system verifies the last saved state, if it was 0 there's no recovery action to be done, if it's 1 it's necessary to roll back the conveyor belt since it's possible there's an uncounted vote in there and also it needs to have the previously authenticated voter unauthorized because they may have left the voting station after the power failure. If the state was 2 the conveyor belt rolls forward since it's possible there's a confirmed vote in there. After recovery actions from

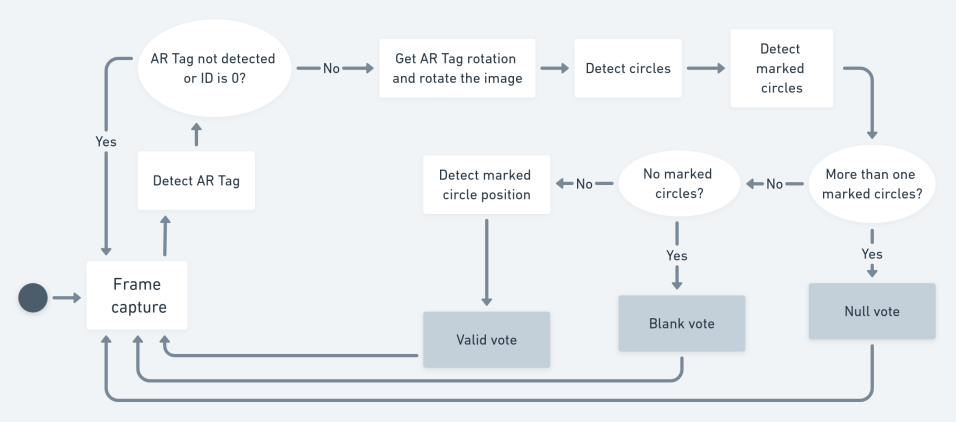

Figure 26: Final voting detection process

state 1 and 2 the new state is 0. The recovery step for state 3 is to roll back the conveyor belt and for state 4 is to roll forwards, both keep state 3 after recovery. Since the system only saves votes in the database after the election has finished it keeps them on a file during the voting process to avoid losing data during a power failure.

3.7 Recount

When the administrator press the recount button on the app after the election has finished they need to be authenticated on the voting station and insert each ballot that was on the result's box. The system checks each ballot identifier and rejects votes that were not accounted for during the election or that already were counted during the recount (duplicated ballots).

3.8 Tests

3.8.1 Fingerprint authentication

The system can handle enrolled voters and display a properly message. Sometimes the recognition take a while, but during all the project development all matches were done correctly.

3.8.2 Conveyor belt

Several tests were done to define the best conveyor belt speed, looking to mechanical limitations. Besides that, the stepper motor control was adapted to fits better the application.

3.8.3 Camera

The main problems with the camera was the illumination and focus, that complicated the vote recognition. To solve the illumination problem, the team bought a ring light and it was enough to solve the issue. Camera focus was solved with the distance adjustment between the camera and the conveyor belt. Besides that, it was necessary some tests to adjust manually the focus in the camera.

3.8.4 Numpad

The first numpad utilized by the team suddenly stopped working and was replaced by a new one.

3.8.5 Servomotor

The servomotor is used to control the lid, witch opens when the voters can insert the vote. In initial tests the correct angles were found and there was no problem in this control during all the project execution.

3.8.6 Vote detection

The tests involving vote detection and their results can be found in the Table 6. The system allows the voters to cancel the vote in case of wrong detection or change of mind.

Test	Result
Valid vote not detected	After 5 failed attempts of detection the conveyor belt changes direction and returns the vote.
Blank vote	The ring light provides a good illumination inside the box, but sometimes blank votes are not recognized as they should. In cases of non detection, the conveyor belt changes direction and the voter can repeat the voting process. If a wrong detection occurs, the voter has the option to cancel the vote.
Null vote	Votes are detected as null if more than one circle is marked. As the blank vote case, the illumination is a key to the correct detection and sometimes valid votes are detected as null ones.
Vote upside down	The AR-tag for the upside down is detected and the conveyor belt returns the vote to the voters.
Valid vote with wrong detection	The voter can cancel the vote in case of wrong detection.

Table 6: Vote detection tests

3.9 Power failure recovery

The tests simulated power failures by unplugging the voting station from the power outlet. It was repeated during different voting steps such as authentication, before and after vote detection and during recounting. The system was able to recover from the power failure and proceed the voting or recounting without losing any data.

3.10 Recount mode

The tests were done after an election scheduled end and consisted of the administrator using the app to enter the recount mode, having an administrator pass the authentication and inserting ballots. The team tested not only the ballots that were counted during the election but also duplicated and new ones. The method worked as expected rejecting these ballots and only counting the ones that had the identifier previously found in the voting time.

4 Difficulties

Throughout the development of the project there were some challenges and difficulties. With the COVID-19 pandemic still going on, the main challenge was the gathering of the group members to do some manual-labor tasks, since it was

necessary to take as much precautions as needed, always trying to keep everyone in the group safe during the execution of the project.

Alongside with all that, some other difficulties appeared during the execution. Already in the mechanical structure assembly, some problems with the lighting took a lot of the group's time to be solved. Due to the need of good lighting inside the ballot box, some original options were discarded since they did not lit up the papers enough to be easily recognized by the camera. This problem was solved by changing the original light bulb to a small ring light, which was able to fit inside the box and provide the necessary illumination for the system.

Another great issue that appeared was the complexity of a QR Code identification on the ballots. The group noticed that the processing time and the image recognition step could be simplified and improved at the same time by changing the QR Codes to AR Tags.

All those tasks took some more hours of development from the group, but that was necessary to keep the project into its best performance, both by making sure the lights inside the box were lighting up enough and by making the image processing as fast and simple as possible.

5 Results

5.1 Final Results

With all the parts of the system working and fully tested together, it is possible to say the results were in line with the expectations. The fingerprint sensor can authenticate the voters and falls back when it is not possible, which leads to the authentication through the numpad. The authentication communicates to the rest of the system that it can keep going and the vote can be inserted and processed.

Both the mobile app and the web application do their jobs in retrieving and sending data to the database, allowing the election to be held and the results and follow-up to be kept on track. Therefore, it is possible to say that the results of the project were the ones the group expected to achieve at the moment of the initial planning. The final assemble can be seen on Figure 27.

Figure 27: Final voting station

5.2 Budget

The budget of the project was very much around what the group expected to spend and it can be checked on Table 7.

Item	Quantity	Unit Price (R$)	Total Price (R$)
Raspberry Pi 3 Model B	1	350.00	350.00
Fingerprint sensor FPM10A	1	78.50	78.50
RPi Camera	1	40,00	40.40
Numpad	1	29,89	29.89
Backup Numpad	1	29.89	29.89
16x2 LCD Display	1	23.90	23.90
Servomotor	1	20.00	20.00
Step Motor	1	10.00	10.00
LED	3	1.00	3.00
Box	1	20.00	20.00
Conveyor Belt	1	20.00	20.00
Light Bulb	1	5.00	5.00
Shipping	1	50.00	50.00
Spare Components	1	200.00	200.00
Total			**880.18**

Table 7: Project budget

5.3 Schedule

Right from the beginning of the project, the group created a list of tasks that would be necessary to be done along the development and the expected amount of time each one of them would take. With that, it was possible to keep track of the group's progress while also registering all the hours of work spent. A summary of the project schedule can be checked on Table 8

Deliverable	Expected Hours	Worked Hours
Project Plan	6	8.3
Follow-up blog	5.1	1.5
Mechanical Structure Project	31	25.3
Hardware Project	92	82.2
Software Project	107	94.4
Hardware + Software Integration	57	89.5
Integration with other Software elements	60	50.5
Technical Report	27	27
Final Presentation	12	11
Total	**397.1**	**389.7**

Table 8: Project schedule summary

6 Conclusions

The project was a challenge to all the members of the group, mainly due to the difficult scenario in which Integration Workshop 3 classes had to happen. With the pandemic still going on, the team had to keep as much of the work in a remote system as possible, which led to a strict time schedule.

Due to the detailed project planning and creation of a task schedule at the beginning of the course, the group was able to take the project forward and achieve its finals goals at the end. The development of Election Pipa improved many skills of the members, and also made possible for the group to work on something everyone was interested in. It is worth mentioning the soft skills this projected developed on the team members, who had to work some of the time remotely and some in person, with people who work on different things.

The development of Election Pipa, a paper-based election system, fully integrated, with a branch of technologies involved, was a successful task. It brought a lot of knowledge to the people involved in this work, and the results were in agreement with the expectations.

References

[1] Adafruit Team. Adafruit optical fingerprint sensor. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-optical-fingerprint-sensor.pdf, 2021. [Online; accessed 27-August-2021].

[2] MongoDB Team. Mongodb. https://www.mongodb.com/, 2021. [Online; accessed 27-August-2021].

[3] RealmSwift Team. Realmswift. https://cocoapods.org/pods/RealmSwift, 2021. [Online; accessed 27-August-2021].

[4] NodeJS Team. Nodejs. https://nodejs.org/en/about/, 2021. [Online; accessed 27-August-2021].

[5] Express Team. Express. https://expressjs.com/, 2021. [Online; accessed 27-August-2021].

[6] Heroku Team. Heroku. https://devcenter.heroku.com/, 2021. [Online; accessed 27-August-2021].

[7] TinkerCAD Team. Tinkercad. https://www.tinkercad.com/, 2021. [Online; accessed 27-August-2021].

[8] Sketch Team. Sketch. https://www.sketch.com/, 2021. [Online; accessed 27-August-2021].

[9] Draw.io Team. Draw.io. https://www.draw.io/, 2021. [Online; accessed 27-August-2021].

[10] Xcode Team. Xcode. https://developer.apple.com/xcode/, 2021. [Online; accessed 27-August-2021].

[11] HK Yuen, J Princen, J Illingworth, and J Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71–77, 1990.

[12] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[13] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[14] Francisco Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76, 06 2018.

[15] Election Pipa Team. Election pipa blog. https://delirious-rhubarb-e58.notion.site/Election-Pipa-ee1d986b7caa4984a9de9ffcaecaa3d7, 2021. [Online; accessed 27-August-2021].