THOMPSON RIVERS UNIVERSITY

CENG 3010 – Digital Systems Design

Design Activity:

Digital Controller for a Programmable Locking Safe

Luka Aitken (T00663672)

Toma Aitken (T00663680)

Brandon Breithaupt (T00591195)

December 2, 2022

Table of Contents

List of Figures

List of Tables

1 Introduction

Every year, there are about on average 2.5 million burglaries that occur around the world, with 66% of these being break-ins. Of that, homes without any security system are more likely to be targeted 3 times more than homes with a security system [1]. To prevent important and valuable items from being taken, using a safe is beneficial but issues can occur. These types of issues of safes can range from how reliable the safe can be, the ease of use, and the type of security options the safe contains. Most safes are reliable enough to keep items safe but can tend to wear down over time every time the safe is used to store items. When the safe becomes unusable, it can cause items to be easily stolen or trapped due to the safe breaking down, and would cost more money to pay to replace the safe. With each safe being able to keep items safe, the safe itself should be easy to use, but many people use older safes to save money which can lead to these users not knowing how to operate the safe. This can eventually make the person want to not use the safe or make them waste more money on a different safe. Finally, with safes having limited reliability and ease to use, each safe has different security measures that can be both beneficial but also disadvantageous. These disadvantages can be from the safe only having one combination that is easy to break, no countermeasures such as one-time use codes and a limited number of tries, and a timer to either keep the safe open for a duration or to be only used during a certain time period in a day [2]. With all these flaws that can lead to break-ins and loss of items, using a programmable safe is the key to preventing this and that's where our programmable safe comes in.

Figure 1.1 - Image of Someone breaking into a safe [12]

To begin with our programmable locking safe, we will come up with different possible solutions that will require minimal effort and ease of use for users, while following the project requirements and constraints. The task was to come up with a solution that is programmable to have enhanced security features for users' personal items that they want to keep safe. The safe should be interactive with the user allowing the users to perform measures to keep belongings safe.

Through this report, we will discuss the problems we faced with the designing and building of the programmable locking safe. We will also discuss other solutions that weren’t used and how they inspired the final solution. Additionally, the environmental, societal, safety and economic considerations are other key factors that will be discussed throughout this report.

2 Design Problem

2.1 Problem Definition

According to security expert, R. Edwards, the average loss from a robbery of homes is worth about $3000, which can include cash and expensive items. With that, about 50% of stolen goods are irreplaceable or have sentimental value towards the owner which can cause emotional and mental stress [3]. Programmable safes can provide protection over sentimental items and prevent items from being either damaged or stolen, reduce the need for more security, prevent sentimental items from being damaged by fire, and give relief to the item's owner by reducing the stress of the owner [4].

2.2 Design Requirements

2.2.1 Functions

- F-1: The system should allow users to open the door on a keypad or/and remote device.
- F-2: The system should use the 7-segment display to display information when the safe is being used.
- F-3: The system should have a reset function for inputting a new password. The time limit on how long it will sit idle will be based on a clock.
- F-4: If the system goes over 3 attempts when entering an incorrect password, an alarm will go off. The alarm will also go off if the door is open for too long.
- F-5: The door should lock when it is either closed, the lock button has been pressed on a remote device, or after the time limit.
- F-6: The door should unlock when a correct password has been implemented.

Performance	8
Safety	9
Simplicity	7
Maintenance	7
Affordability	8

Table 2.2.1 - Importance Chart

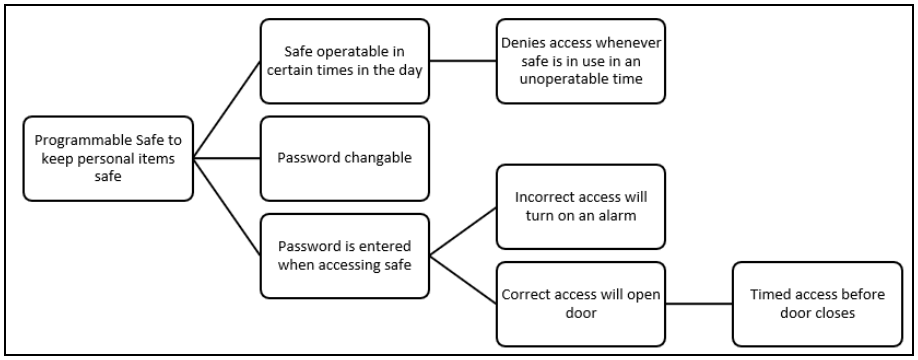


Figure 2.2.1 - Function Tree

2.2.2 Objectives

- O-1: The system should be easy to use for new users.
- O-2: The system should be able to keep the user's items safe.
- O-3: The system should use a sequential machine to run all possible outcomes.
- O-4: The system should be able to change the password when needed to.
- O-5: The system should be cost-effective.

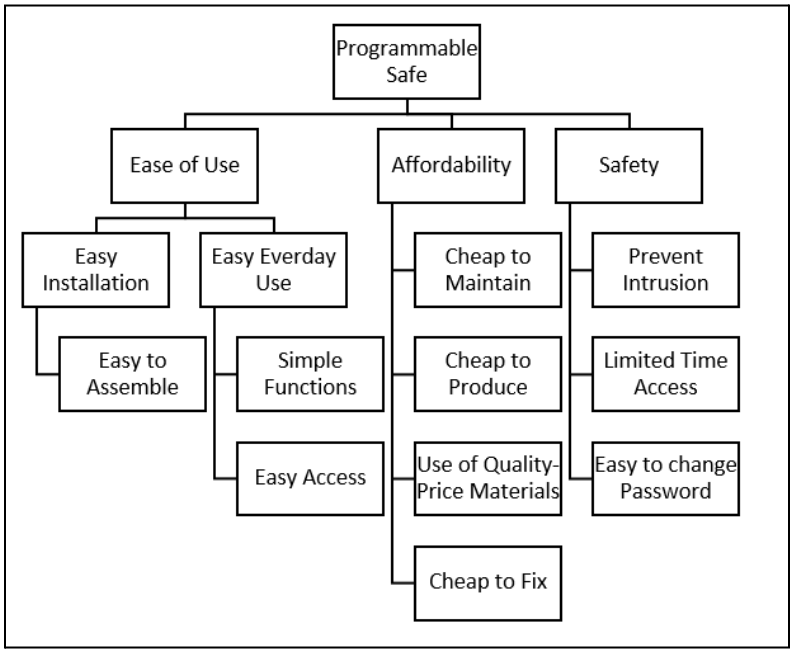


Figure 2.2.2 - Objective Tree

2.2.3 Constraints

- C-1: The door of the safe should only open when the correct password is entered.
- C-2: The safe should automatically lock after a certain time frame.
- C-3: The safe should use one or more sequential machines.
- C-4: The Safe should be cost-effective.

2.2.4 Functional Requirements

- FR-1: User inputs password through keypad to unlock safe door.
- FR-2: User closes the safe door and is able to lock the safe.
- FR-3: User is able to reset the password once the safe has been unlocked.

2.2.5 Non-Functional Requirements

- NFR-1: Keeps personal belongings in the safe.
- NFR-2: Gives output whether password was correct or incorrect.
- NFR-3: Coloured LED lights to indicate the state of the safe such as locked or unlocked.

2.2.6 Scope and Complexity

- SC-1: The system will be implemented through one Digilent Basys 3 FPGA board.
- SC-2: The system will be made in VHDL code.
- SC-3: The system will use a keypad to allow users to enter a password to open the safe.

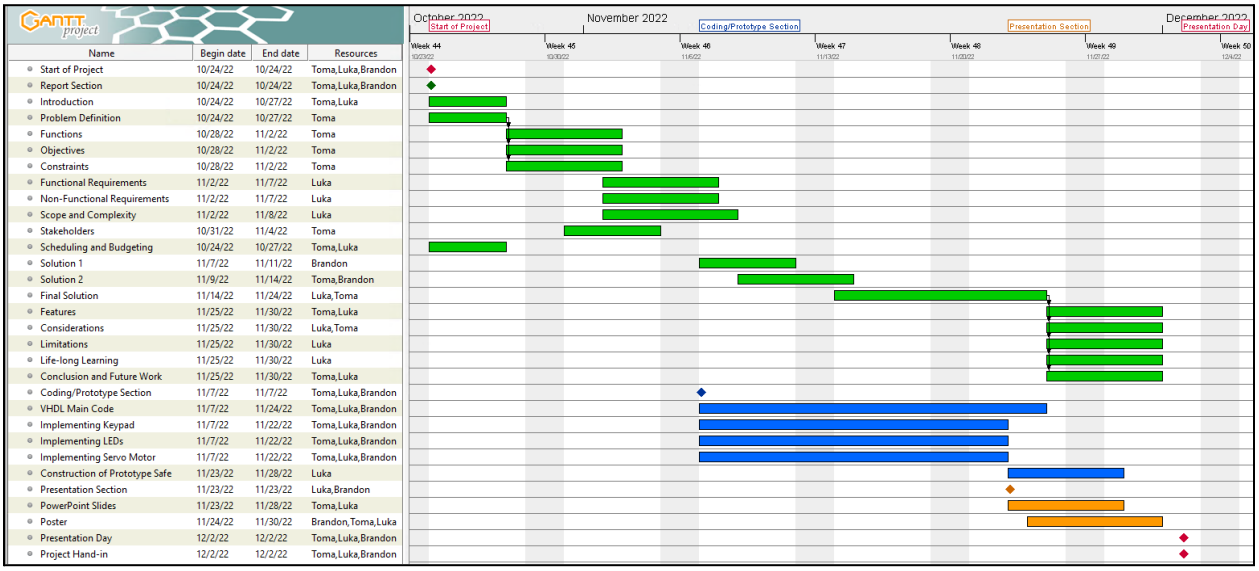
3 Stakeholders

Stakeholders	Required Needs
User / Owner of the safe	• To keep important and valuable items safe from unwanted users or environments. Such as unwanted guests, burglars and house fires.
Programmers creating the code	• To create a code that can prevent break-ins into the safe. • To create a code that can't be hacked easily.
Programmable Safe Company	• To generate money from users who require a safe to protect and keep important or valuable items safe. • To generate money from users to improve the programmable safe.

Table 3.1 - Stakeholder Table

4 Scheduling and budgeting

4.1 Milestones



Name	Begin date	End date	Resources
Start of Project	10/24/22	10/24/22	Toma,Luka,Brandon
Report Section	10/24/22	10/24/22	Toma,Luka,Brandon
Introduction	10/24/22	10/27/22	Toma,Luka
Problem Definition	10/24/22	10/27/22	Toma
Functions	10/28/22	11/2/22	Toma
Objectives	10/28/22	11/2/22	Toma
Constraints	10/28/22	11/2/22	Toma
Functional Requirements	11/2/22	11/7/22	Luka
Non-Functional Requirements	11/2/22	11/7/22	Luka
Scope and Complexity	11/2/22	11/8/22	Luka
Stakeholders	10/31/22	11/4/22	Toma
Scheduling and Budgeting	10/24/22	10/27/22	Toma,Luka
Solution 1	11/7/22	11/11/22	Brandon
Solution 2	11/9/22	11/14/22	Toma,Brandon
Final Solution	11/14/22	11/24/22	Luka,Toma
Features	11/25/22	11/30/22	Toma,Luka
Considerations	11/25/22	11/30/22	Luka,Toma
Limitations	11/25/22	11/30/22	Luka
Life-long Learning	11/25/22	11/30/22	Luka
Conclusion and Future Work	11/25/22	11/30/22	Toma,Luka
Coding/Prototype Section	11/7/22	11/7/22	Toma,Luka,Brandon
VHDL Main Code	11/7/22	11/24/22	Toma,Luka,Brandon
Implementing Keypad	11/7/22	11/22/22	Toma,Luka,Brandon
Implementing LEDs	11/7/22	11/22/22	Toma,Luka,Brandon
Implementing Servo Motor	11/7/22	11/22/22	Toma,Luka,Brandon
Construction of Prototype Safe	11/23/22	11/28/22	Luka
Presentation Section	11/23/22	11/23/22	Luka,Brandon
PowerPoint Slides	11/23/22	11/28/22	Toma,Luka
Poster	11/24/22	11/30/22	Brandon,Toma,Luka
Presentation Day	12/2/22	12/2/22	Toma,Luka,Brandon
Project Hand-in	12/2/22	12/2/22	Toma,Luka,Brandon

Figure 4.1 - Gantt Chart

4.2 Responsibilities

Step	Project Initiation	Toma	Luka	Brandon
1	Writing Report	A/R	R/C	I
2	Building Physical Model	C	R	C
3	Making code for Keypad	C	R	C
4	Making code for System	C	C	R

I = Inform these group members about steps.

C = Consulted with these group members.

R = Responsible for completing that step.

A = Accountable to complete that step.

Table 4.3.1 - Responsibility Assignment Matrix

4.3 Budgeting

Table 4.3.2 - Budgeting List		
Item Name	Quantity Used	Cost
FPGA board [5]	1	~$242.00 (Given by Lab Technician)
Pmod KYPD: 16-Button Keypad [6]	1	~$25.00 (Given by Lab Technician)
Pmod ESP32 bluetooth adapter [7]	1	~$47.00 (Given by Lab Technician)
Scrap Cardboard	~3	Free

5 Solution

5.1 Solution 1

For our first solution, our approach was to make a design using multiple components that would be implemented on the Basys 3 FPGA board. The main component for the locking safe is the Digilent Pmod keypad which would allow users to enter the password to open the safe door. The keypad would allow users to easily and simultaneously enter a password without confirming each number entered. If the password entered matches the password set in the VHDL code, it opens the safe door by using a servo motor to unlock the door and light up a green LED to confirm that the password was correct. If a password was entered wrong, the safe would light up a red LED indicating that the password was incorrect. The user would get a total of three attempts before the system will lock the user out for a certain amount of time. The user would also be able to reset the password once they have entered the correct password, where they would then be able to enter the new desired password. For purposes of the development of the safe, there would be a master password that can be entered to open the safe which would be intended for the use of the manufacturers. If the locking safe has been opened for too long, a buzzer would go off to indicate the alarm of the safe. Once the user is done with the safe and wants to lock the safe, the user would close the door and press the “C” button to activate the servo motor to close and restart the program.

The reason why this solution was not selected was due to the coding for the keypad itself. The coding for the keypad was unable to store the memory of each digit that was entered for the password. We were unable to figure out how to enter a digit simultaneously without having to confirm each digit and due to time constraints we decided it was best to move on to a new solution.

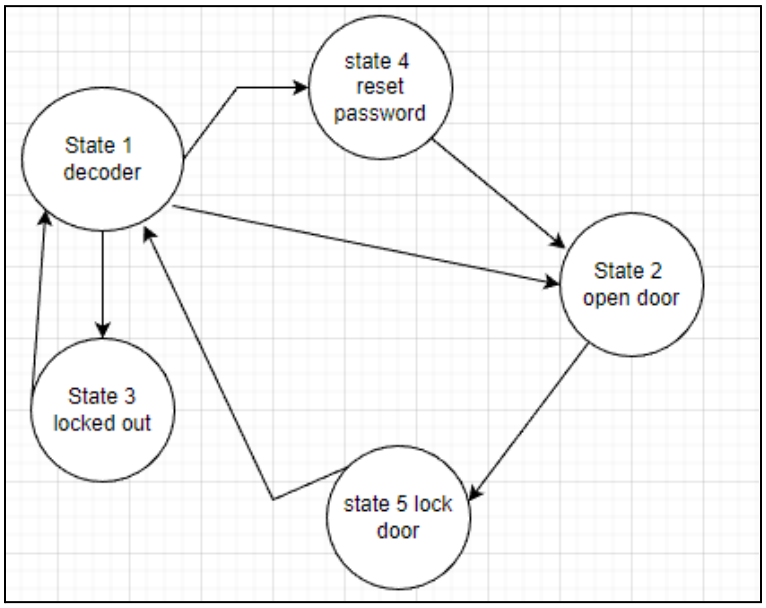


Figure 5.1.1 - Solution 1 State Diagram

5.2 Solution 2

For our second solution, our approach was to take the same ideas as from the first solution and build upon it as well as figure out how to save the digits entered from the keypad and display all digits entered at the same time. With the keypad, the users would be able to input a value to display the current digit. To move to the next digit the user would have to press the “E” button on the keypad in order to move along the four digit seven segment display. If the password was correctly entered, a green LED would activate and the servo motor would open. If the user entered the incorrect password a red LED would activate and a buzzer would go off indicating that the wrong password was entered. The user would have three tries, and each try a red LED would activate. Once the third red LED is activated, the user would be locked out from trying to open the safe again until a certain period has elapsed. Once the user has correctly entered the password and has entered the safe, there will be an option for the user to change the password if it is desired. There will also be a master password for developers or manufactures to get in the safe for the purpose of testing out functionality of the safe. When the safe is opened, there will be a time limit for how long the safe can be opened for until a buzzer would go off signaling the alarm of the safe. When the user is done using the safe, you close the door of the safe, and press the “D” button on the keypad indicating “done”. Once the “D” button is pressed it will turn on the servo motor into the close state so that the door is locked and the LEDs will reset to the off state.

Reasons why the second solution was not selected was due to the keypad not collecting and saving the input values that a user would press. For the first digit, the user would press a number that would appear on the far left digit. Once you pressed the “E” button to move to the next digit, it would remove the first input from the display and output the “E” value on the second digit. This was not feasible as our goal was to have all digits displayed on the four digit seven segment display so that the user is able to view the enter inputs that they have entered.

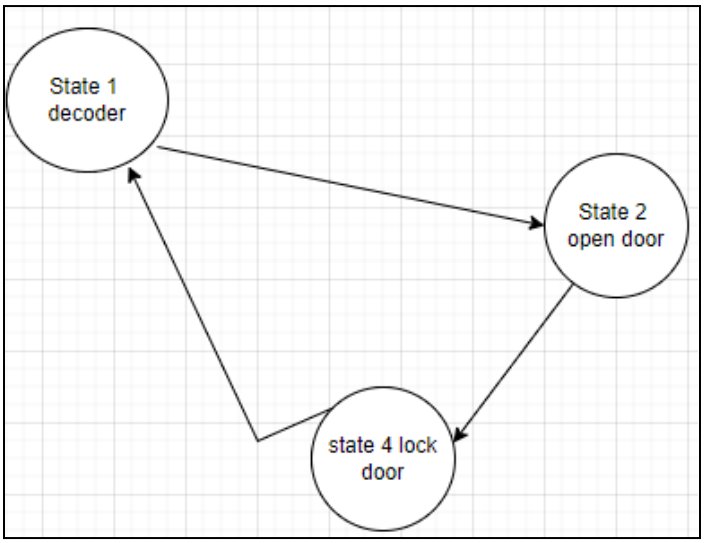


Figure 5.1.2 - Solution 2 State Diagram

5.3 Final Solution

For our final solution, we came up with a design where the user would enter the password using a keypad connected to the FPGA board. Our system displays 1 digit at a time. When the user needs to change the other digits, they would use two switches indicating from digit 1 to digit 4. Our program used cases to allow the system to easily change from one digit to another such as 11, 10, 01, and 00. If the user tries to open the door without entering a password or with an incorrect password, the safe will turn on an alarm meaning that the password is wrong or someone was tampering with the safe. If the user enters the password correctly, the program will allow the user to open the safe and once they do, a timer is started. The timer has a set duration until an alarm is started due to the safe being open for too long, which can lead to items being taken away. If the user closes the door before the timer is up, the system will go back to its original state asking for a password. With each state, there are LEDs indicating if the password is correct, the door is open and the alarm is triggered.

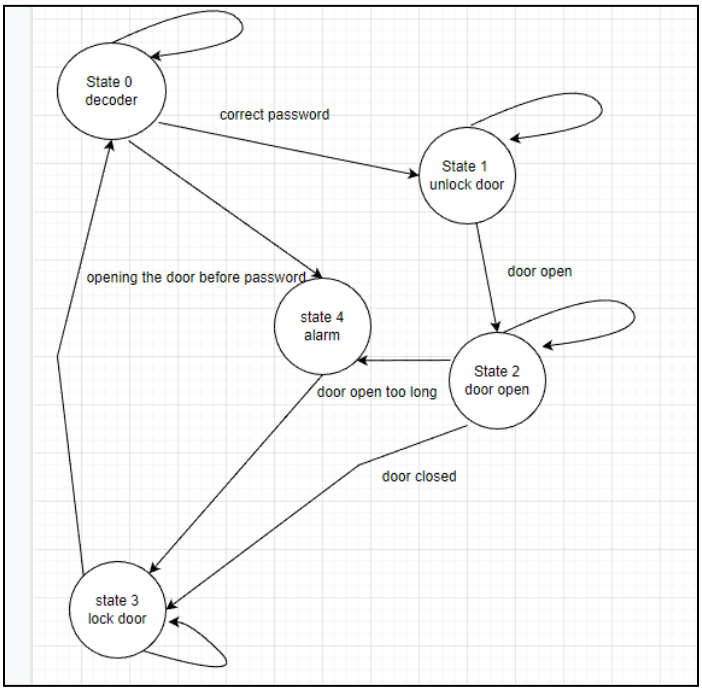


Figure 5.1.3 - Final Solution State Diagram

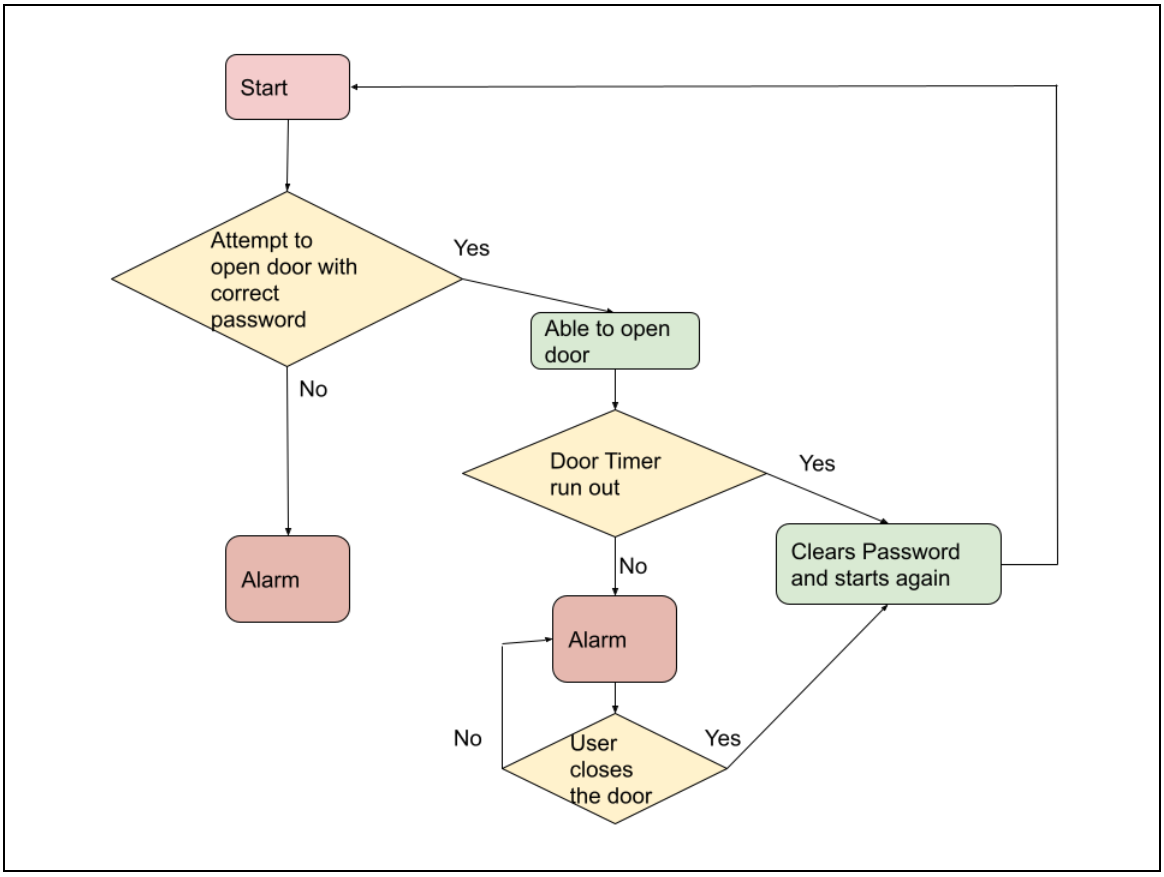


Figure 5.1.4 - Flow Chart of Final Solution

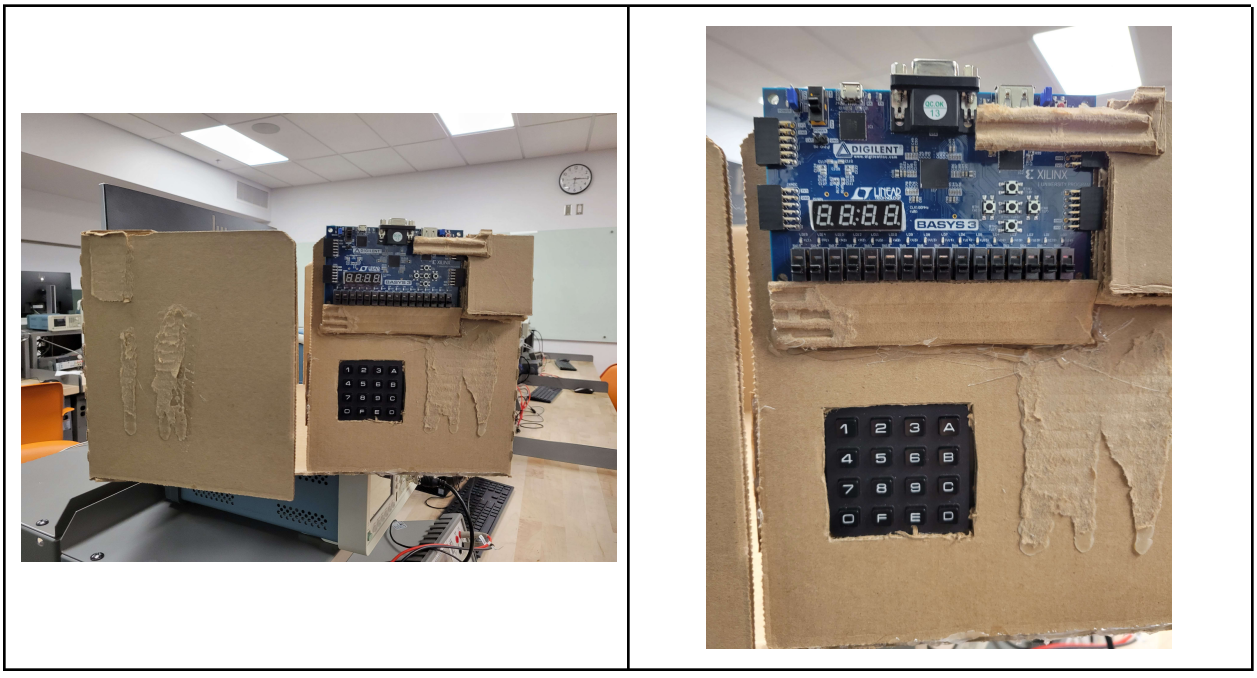


Figure 5.1.5 - Final Solution Prototype Safe

		Solutions					
		Solution 1		Solution 2		Final Solution	
Criteria	Weight	Score	Partial Score	Score	Partial Score	Score	Partial Score
Security	0.40	7/10	0.280	6/10	0.240	8/10	0.320
Safety	0.35	6/10	0.210	6/10	0.220	7/10	0.245
Simplicity	0.25	6/10	0.150	8/10	0.200	7/10	0.175
Sum	1.00		0.640		0.660		0.740

Table 5.1 - Decision matrix chart for the considered alternatives

5.3.1 Features

For our features, the programmable locking safe will use a keypad to allow users to enter a password. To enter a password, a user must use the two most right switches on the FPGA board. These two switches can be represented in binary digits for choosing which display you will be starting on with the 4 digit 7 segment display. The leftmost 7 segment displays the “11” case, next it would be “10”, then “01”, and the rightmost digit would be “00”. Current password is “1234” as shown in Figure 5.2.1.

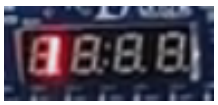
First Digit (“11”)	Second Digit (“10”)	Third Digit (“01”)	Fourth Digit (“00”)
			

Figure 5.2.1 - Image of each digit with their positions and switch combination

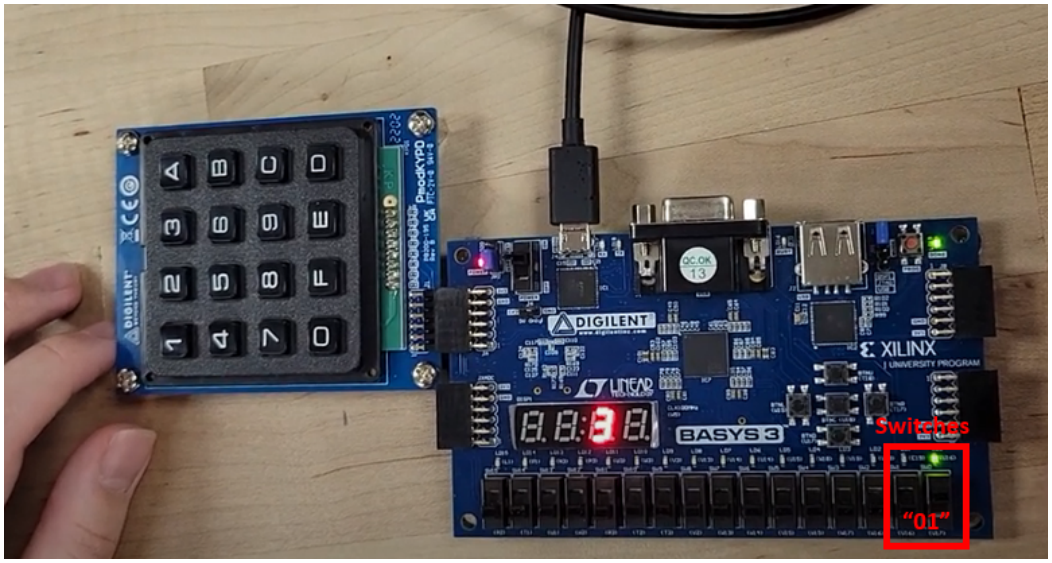


Figure 5.2.2 - Image of FPGA showing digits and switches

When users are entering a password, any button located on the keypad can be used as an input for the password. To prevent other unwanted users from viewing your password, we have an

enhanced security measure that will only allow users to view one digit at a time for optimal security.

When the user has correctly entered a password, the door will be unlocked which will be represented as the third most right LED on the FPGA board. If the door is open for more than five seconds, the timer in the program will cause the fifth most right LED to start blinking. This indicates that the safe is in the alarm state and that the user needs to close the door with the third-most right switch. If a user tries to open the door with the incorrect password, the alarm state will be set off and cause the fifth-most right LED to start blinking.

Correct Password	Door is open	Alarm is trigger
		

Figure 5.2.3 - Image of LEDs of Password, Door and Alarm

5.3.2 Environmental, Societal, Safety, and Economic Considerations

When designing our programmable locking safe, we took into consideration the safe to make sure that it was environmentally friendly, safe to use, and inexpensive. We also wanted to take into consideration what elements we included to make our safe much easier to use and understand.

For our environmental considerations, we used a simple sequential machine to make the system faster and less harmful to the environment. If the program used unnecessary lines of code and states to perform tasks for the programmable locking safe, it could cause more harm to the environment by consuming more energy and leading to a bigger carbon footprint. Using simpler, but effective states and lines of code will produce less energy meaning a reduced carbon footprint [8]. With our physical prototype, the entire safe structure is made out of cardboard which can be recycled easily.

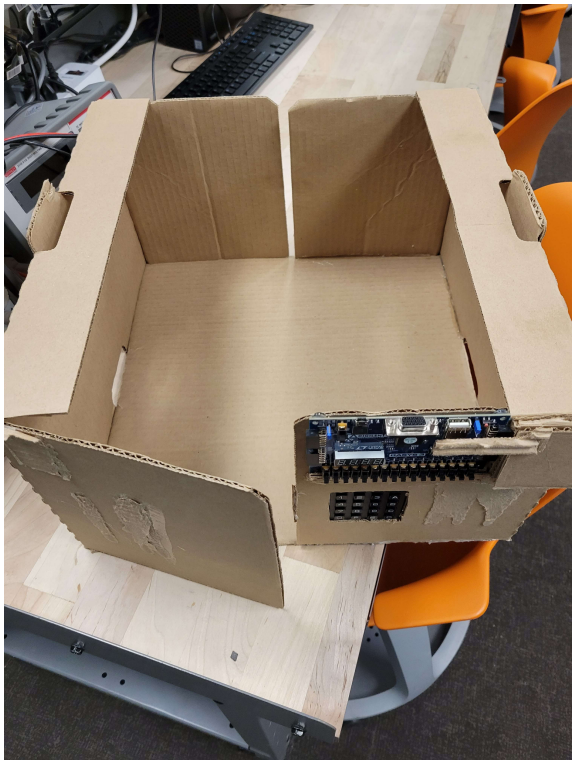


Figure 5.3.1 - Cardboard Prototype

For our societal considerations, we know that users want to keep their belongings safe. Our safe is built strong and tamper-proof. We have equipped the safe with an alarm in case of any type of tampering with the safe door without a password.

For our safety considerations, we implemented a timer for the door whenever the user opens the safe. The safe would have a time limit before the alarm state turns on letting the user know that the safe door is still open and must be closed to protect the user's belongings. To protect the user password, each digit is shown one at a time so that if any unwanted people are looking at your password, they will not see the full combination.

For the economic considerations, we are using the minimal amount of external components that are required to make a fully functional locking safe while keeping the safe's integrity high. We plan on making the safe inexpensive and cost-effective due to the minimal components used. This result will lower the manufacturing cost of building a locking safe as well as give users an affordable option when looking for a programmable locking safe.

5.3.3 Limitations

When building and developing our programmable locking safe, there are always limitations. Many of our limitations were due to time constraints. This is because we spent a lot of time figuring out and trying to understand how the keypad functions and how we can save a user's input value in our program. Due to these time constraints, we were unable to do a test bench to test the different input and output values within our program. If there was more time, we

would be able to use a test bench to see all possible inputs and outputs. There were also other features that we wanted to perform such as the user resetting the password to their choosing. Additionally, we wanted to use external colour LEDs to represent different states within our program such as when the door is locked or unlocked. With the door, we wanted to use a servo motor to act as the locking mechanism that would display our physical prototype locking instead of an LED. With our physical prototype safe, the safe itself was made out of cardboard which is not a realistic safe and is easily accessible to break-ins. For our design purposes, environmental considerations, and cardboard being easily accessible, we used cardboard as our main safe structure.

5.3.4 Lifelong Learning

When building the program for our programmable locking safe, we learn a lot of information on how to create a decoder for the PMod keypad. When first learning about the PMod keypad, we had to figure out how our keypad works and how it connects with the Basys 3 FPGA board. This took a while as most examples of how to decode the keypad were using Verilog rather than VHDL. Once we figured out a way to decode it, we integrated it into our system which allows us to enter any button on the keypad, and it would convert it into binary code that can work with the seven-segment displays. After getting these numbers, we were able to compare them with a decode value that can take it as a 4-bit binary number such as “0010” which represents ‘2’ on the keypad.

6 Team Work

6.1 Meeting 1

Time: October 15, 2022, 2:00 pm to 3:30 pm

Agenda: A list of topics to be discussed in meeting 1

1. Review project requirements and constraints.
2. Brainstorm possible solutions.
3. Decide what each team member wants to work on.

Team Member	Previous Task	Completion State	Next Task
Luka	N/A	N/A	Brainstorming Ideas
Toma	N/A	N/A	Brainstorming Ideas
Brandon	N/A	N/A	Brainstorming Ideas

Meeting Minutes: A list of decisions made in meeting 1

1. All team members will do individual research on topics for the project and report back on possible ideas in the next team meeting.
2. Next meeting distribution of tasks for the report will be given out to each team member.

6.2 Meeting 2

Time: October 22, 2022, 4:00 pm to 5:00 pm

Agenda: A list of topics to be discussed in meeting 2

1. Share research on topics for the project.
2. Distribution of tasks for the report will be given out.
3. Create a full list of tasks for the entire project.
4. Discuss possible solutions.

Team Member	Previous Task	Completion State	Next Task
Luka	Brainstorming Ideas	95%	Create a Gantt chart. Begin introduction of the report. Research about the project.
Toma	Brainstorming Ideas	95%	Begin introduction and problem definition of the report. Research about the project.
Brandon	Brainstorming Ideas	95%	Research about the project.

Meeting Minutes: A list of decisions made in meeting 2

1. Luka and Toma will begin writing the introduction of the report..
2. Toma will begin writing the problem definition of the report.
3. Luka will begin making a Gantt chart with a list of tasks to be done.
4. All team members will do research about the project.

6.3 Meeting 3

Time: November 5, 2022, 5:00 pm to 6:00 pm

Agenda: A list of topics to be discussed in meeting 3

1. Review the introduction and problem definition of the report.
2. Reviewing and adding to the functions, objectives, constraints, functional, non-functional, scope and complexity in the design problem section of the report.

Team Member	Previous Task	Completion State	Next Task
Luka	Create a Gantt chart. Begin introduction of the report. Research about the project.	85%	Begin the stakeholders and scheduling and budgeting section of the report.
Toma	Begin introduction and problem definition of the report. Research about the project.	90%	Begin the stakeholders and scheduling and budgeting section of the report.
Brandon	Research about the project.	95%	Continue to add to the functions, objectives and constraints.

Meeting Minutes: A list of decisions made in meeting 3

1. Each team member will go through a report and make any necessary edits.
2. Next meeting will begin the discussion of the code and deciding what components will be implemented to the FPGA board.

6.4 Meeting 4

Time: November 15, 2022, 4:00 pm to 6:00 pm

Agenda: A list of topics to be discussed in meeting 4

1. Review the functional, nonfunctional, scope and complexity sections in the report.
2. Discuss the components that will be used for the project.
3. Begin writing the code for the locking safe system.

Team Member	Previous Task	Completion State	Next Task
Luka	Begin the stakeholders and scheduling and budgeting section of the report.	90%	Begin and research ideas for the project code. Come up with a possible solution.
Toma	Begin the stakeholders and scheduling and budgeting section of the report.	90%	Begin and research ideas for the project code. Come up with a possible solution.
Brandon	Continue to add to the functions, objectives and constraints.	90%	Begin and research ideas for the project code. Come up with a possible solution.

Meeting Minutes: A list of decisions made in meeting 4

1. Luka will sign out the FPGA board, keypad, and bluetooth module from the lab technician.
2. Next meeting will be the beginning of the code.

6.5 Meeting 5

Time: November 19, 2022, 3:00 pm to 5:00 pm

Agenda: A list of topics to be discussed in meeting 5

1. Begin the implementation and coding of the keypad to the FPGA board using Vivado.

Team Member	Previous Task	Completion State	Next Task
Luka	Begin and research ideas for the project code. Come up with a possible solution.	90%	Work on the implementation of the keypad.
Toma	Begin and research ideas for the project code. Come up with a possible solution.	90%	Work on the implementation of the keypad.
Brandon	Begin and research ideas for the project code. Come up with a possible solution.	90%	Work on the main system.

Meeting Minutes: A list of decisions made in meeting 5

1. Luka and Toma will continue to figure out how to code and implement the keypad.
2. Brandon will figure out the design for the main code to control all components.
3. Brandon will make the state tables and state graphs for the 3 possible solutions.

6.6 Meeting 6

Time: November 23 , 2022, 11:00 am to 12:00 pm

Agenda: A list of topics to be discussed in meeting 6

1. Continuing working on the keypad implementation and code.

Team Member	Previous Task	Completion State	Next Task
Luka	Work on the implementation of the keypad.	70%	Build a prototype safe out of cardboard. Begin writing the final solution.
Toma	Work on the implementation of the keypad.	60%	Implement the main system with the keypad. Begin writing the final solution.
Brandon	Work on the main system.	70%	Implement the main system with the keypad.

Meeting Minutes: A list of decisions made in meeting 6

1. Once the keypad code has been implemented into the main system, team members will decide what other components will be added if there's still time.
2. Toma and Luka will continue to write in the report about the final solution and topics in relation to it (features, considerations, limitations, life-long learning).

6.7 Meeting 7

Time: November 29, 2022, 3:00 pm to 8:30 pm

Agenda: A list of topics to be discussed in meeting 7

1. Keypad code will be implemented into the main system.
2. If time is still available, team members will try adding more external components to the FPGA board.

Team Member	Previous Task	Completion State	Next Task
Luka	Build a prototype safe out of cardboard. Begin writing the final solution.	90%	Begin writing the conclusion. Begin final testing of prototype. Make Poster
Toma	Implement the main system with the keypad. Begin writing the final solution.	90%	Begin writing the conclusion. Begin final testing of prototype. Create PowerPoint Slides.
Brandon	Implement the main system with the keypad.	90%	Begin final testing of prototype.

Meeting Minutes: A list of decisions made in meeting 7

1. Next meeting will be for final testing and finishing the report of the project
2. Toma will begin the PowerPoint for presentation.

6.8 Meeting 8

Time: December 1, 2022, 5:30 pm to 6:30 pm

Agenda: A list of topics to be discussed in meeting 8

1. Finish programmable locking safe prototype.
2. Go over last editing on report, PowerPoint, and poster

Team Member	Previous Task	Completion State	Next Task
Luka	Begin writing the conclusion. Begin final testing of prototype. Make Poster	90%	Presentation/Hand-in Project.
Toma	Begin writing the conclusion. Begin final testing of prototype. Create PowerPoint Slides.	90%	Presentation/Hand-in Project.
Brandon	Begin final testing of prototype.	90%	Presentation/Hand-in Project.

Meeting Minutes: A list of decisions made in meeting 8

1. All project contents (Technical Report, PowerPoint Slides, Poster) will be handed in on December 2, 2022.

7 Conclusion and Future Work

In conclusion, we achieved many considerable designs on how to create a programmable locking safe that can keep valuable items safe from unwanted people. First off, we came up with a couple of solutions that could be used to make the best possible programmable locking safe, while following all the constraints. Our objective was to build a program that is easy to use and keeps items safe by using sequential machines.

From our research, the biggest challenges for safes are the things that could go wrong. Some such incidents could be home intrusions and break-ins that can lead to normal safes from being broken into and losing valuable items. With a higher frequency of these possible events, one in every six homes is targeted every year. Furthermore, as these events occur, the results can lead to the loss of important items and damages in homes that can cost lots of money to repair or replace [9].

Figure 7: Image of storing items [11]

With the help of programmable locking safes, it can reduce the number of unpleasant incidents when using programmable locking safes. Using these types of safes, can reduce the chance of home intrusions and break-ins as well as overall improve the security of important items. Also, it can save money by reducing the chances of important items being taken from unwanted people and preventing the item's owner from being replaced [9] [10].

For recommendations on how the process will continue in the future of improving the programmable locking safe, we would include more security when operating the safe. This can be such as whenever a user entered the password wrong three times in a row, which could trigger the alarm and a buzzer to indicate the alarm instead of an LED. Also, we would want to implement a BlueTooth device to allow the user to control the safe's functionality on a phone app for easier access or to change the settings of the safe. Some such functions could be opening and closing the safe door, notifications of the safe's activities, and changing the password of the safe in case unwanted people know the password.

8 References

1. S. Stasha. “27 Alarming Burglary Statistics For 2022.” Policy Advice. https://policyadvice.net/insurance/insights/burglary-statistics/ (Accessed October 23, 2022)
2. C. Strauss. “Advantages of Electronic Safe Locks.” Great Valley Lockshop. https://www.gvlock.com/blog/electronic-safe-locks-advantages-disadvantages/#:~:text=They%20are%20easy%20to%20use,to%20use%20in%20the%20dark (Accessed November 5, 2022)
3. R. Edwards. “8 Surprising Home Burglary Facts and Stats.” safewise. https://www.safewise.com/blog/8-surprising-home-burglary-statistics/ (Accessed November 5, 2022)
4. “Benefits of Having a Safe at Home.” Safe Trolley. https://www.safetrolley.com/benefits-having-safe-at-home/ (Accessed November 13, 2022)
5. “410-183.” Digi-Key Electronics. https://www.digikey.ca/en/products/detail/digilent-inc/410-183/4970275 (November 23, 2022)
6. “410-195.” Digi-Key Electronics. https://www.digikey.ca/en/products/detail/digilent-inc/410-195/3902802 (Accessed November 23, 2022)
7. “410-377.” Digi-Key Electronics. https://www.digikey.ca/en/products/detail/digilent-inc/410-377/8599302 (Accessed November 23, 2022)
8. S. Podder. “How Green Is Your Software?.” Harvard Business Review. https://hbr.org/2020/09/how-green-is-your-software (Accessed November 23, 2022)
9. A. Mustafa. “Do Home Safes Really Protect Your Valuables?.” Canadian Security Professionals. https://www.cspalarms.ca/blog/home-safety/do-home-safes-really-protect-your-valuables/ (Accessed November 30, 2022)
10. “The Pros and Cons of Electronic Door Locks.” Locksmith Monkey. https://locksmithmonkey.com/the-pros-and-cons-of-electronic-door-locks/ (Accessed November 30, 2022)
11. “Residential Locksmith Services.” Quickie Key Locksmith. https://quick-key.net/blog/5-common-items-to-store-in-a-home-safe-why/ (Accessed December 2, 2022)
12. N. Pedersen-McKinnon. “Banks raid customers’ redraw savings.” Yahoo Finance. https://au.finance.yahoo.com/news/banks-raid-redraw-savings-210026675.html (Accessed December 2, 2022)
13. “Pmod KYPD.” Digilent Reference. https://digilent.com/reference/pmod/pmodkypd/start?redirect=1 (Accessed December 2, 2022)

9 References

9.1 Additional Information

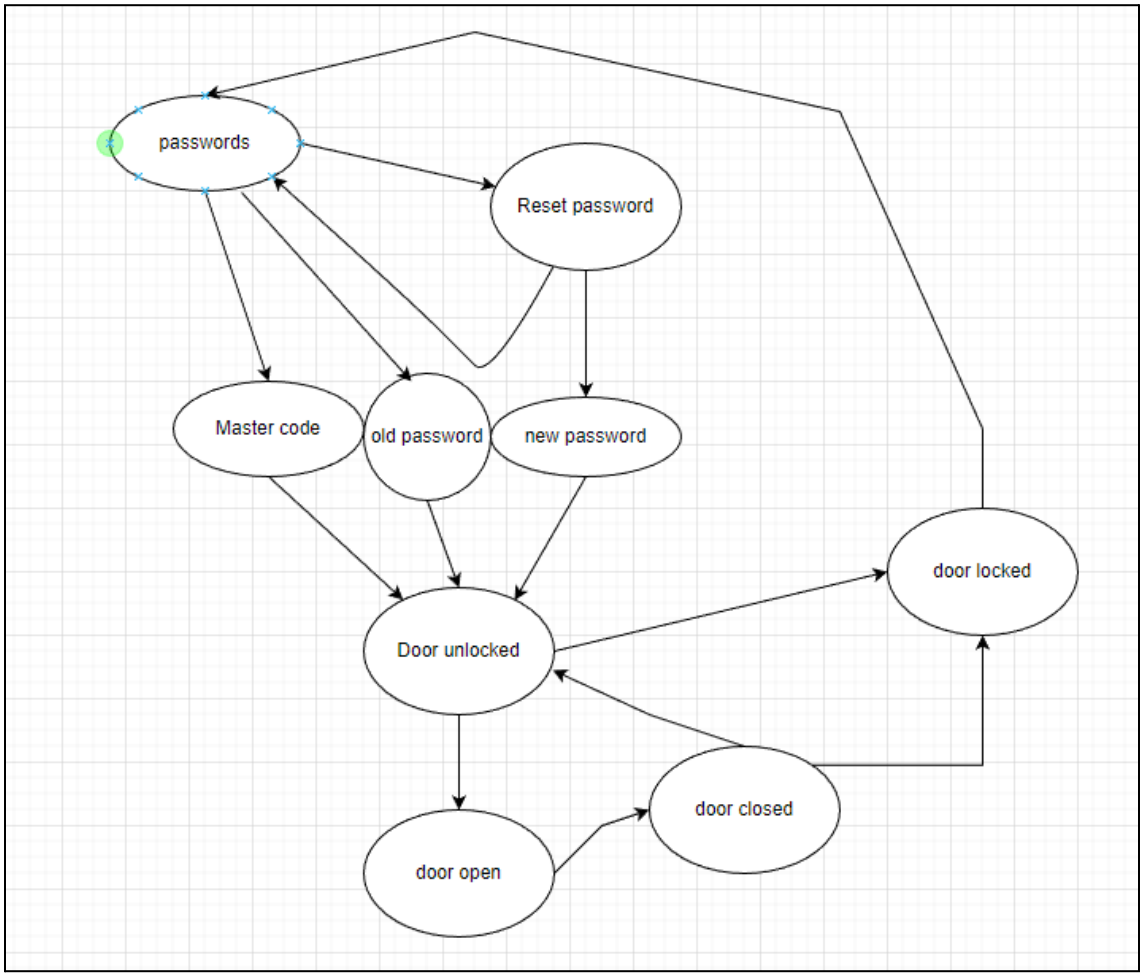


Figure A.1 - First State Diagram for First Design Idea

Youtube Link of the Prototype Working: https://www.youtube.com/watch?v=PupkJDQFcT8

9.2 VHDL Code of Programmable Locking Safe

The base code that was used was an open source code provided by Diligent through examples on how to use the keypad with a FPGA board [13].

Table A.1 - Code For Main File

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity PmodKYPD is
  generic ( ClockF: INTEGER := 100000000);
   Port (
                    clk : in  STD_LOGIC;
                    JA : inout  STD_LOGIC_VECTOR (7 downto 0); -- PmodKYPD is designed to be
connected to JA
        an : out  STD_LOGIC_VECTOR (3 downto 0);   -- Controls which position of the seven
segment display to display
        seg : out  STD_LOGIC_VECTOR (6 downto 0);
        digit_count: in std_logic_vector (1 downto 0); -- digit to display on the seven segment
display
        led: out std_logic;
        door: in std_logic;
        ledDoor: out std_logic;
        alarm: out std_logic
       );
end PmodKYPD;

architecture Behavioral of PmodKYPD is

component Decoder is
         Port (
                  clk : in  STD_LOGIC;
       Row : in  STD_LOGIC_VECTOR (3 downto 0);
                  Col : out  STD_LOGIC_VECTOR (3 downto 0);
       DecodeOut : out  STD_LOGIC_VECTOR (3 downto 0));
         end component;

component DisplayController is
         Port (
                   DispVal : in  STD_LOGIC_VECTOR (3 downto 0);
        anode: out std_logic_vector(3 downto 0);
```

```
        segOut : out  STD_LOGIC_VECTOR (6 downto 0));
          end component;

signal Decode: STD_LOGIC_VECTOR (3 downto 0);


signal pos1: std_logic_vector (3 downto 0);
signal pos2: std_logic_vector (3 downto 0);
signal pos3: std_logic_vector (3 downto 0);
signal pos4: std_logic_vector (3 downto 0);

signal pass1: std_logic_vector (3 downto 0);
signal pass2: std_logic_vector (3 downto 0);
signal pass3: std_logic_vector (3 downto 0);
signal pass4: std_logic_vector (3 downto 0);

signal ticks: integer;
signal nreset: std_logic;
signal Second: integer := 0;
signal State, nextState: integer range 0 to 4;

begin
          C0: Decoder port map (clk=>clk, Row =>JA(7 downto 4), Col=>JA(3 downto 0),
DecodeOut=> Decode);
          C1: DisplayController port map (DispVal=>Decode, anode=>an, segOut=>seg );

pass1 <= "0001";
pass2 <= "0010";
pass3 <= "0011";
pass4 <= "0100";

process(State)
begin
   case State is
      when 0 =>
      nreset <= '0';
      led <= '0';
      ledDoor <= '0';
      alarm <= '0';
      case digit_count is
         when "00" =>
            an <= "1110";
```

```
            pos4 <= Decode;
          when "01" =>
            an <= "1101";
            pos3 <= Decode;
          when "10" =>
            an <= "1011";
            pos2 <= Decode;
          when "11" =>
            an <= "0111";
            pos1 <= Decode;
          when others =>
            an <= "1111";

       end case;
       if ((pos1 = pass1) and (pos2 = pass2) and (pos3 = pass3) and (pos4 = pass4)) then
nextState <= 1;

      else
         if (door = '1') then
          nextState <= 4;
          else
           led <= '0';
           nextState <= 0;
          end if;
       end if;

       when 1 =>
          nreset<= '0';
          led <= '1';
          if (door = '1') then
          nextState <= 2;
          else
          nextState <= 1;
          end if;
       when 2 =>
          ledDoor <= '1';
          nreset <= '1';
          if ( second = 5) then
            nextstate <= 4;
            nreset <= '0';
          elsif (door = '0') then
          nextState <= 3;
          else
```

```
        nextState <= 2;
        end if;
    when 3 =>
        led <= '0';
        ledDoor <= '0';
        alarm <= '0';
        nreset<= '0';
        pos1 <= "0000";
        pos2 <= "0000";
        pos3 <= "0000";
        pos4 <= "0000";
        nextState <= 0;

    when 4 =>
        nreset <='1';
        if (Second mod 2 = 0) then
        alarm <= '1';
        else
        alarm <= '0';
        end if;
        led <= '0';
        if (door = '0') then
         nextState <= 3;
        else
        nextState <= 4;
        end if;
    when others => null;
    end case;
    end process;

process(clk)
begin
   if clk'EVENT and clk = '1' then
   State <= nextState;
   end if;
      if rising_edge(clk) then
         -- If the negative reset signal is active
         if nreset = '0' then
            ticks   <= 0;
            Second <= 0;
         else
            -- True once every second
            if ticks = ClockF - 1 then
```

```
            ticks <= 0;
            Second <= Second + 1;
          else
            ticks <= ticks + 1;
          end if;
        end if;
      end if;
end process;
end Behavioral;
```

Table A.2 - Code for Keypad Decoder (Component to Main File)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Decoder is
   Port (
                              clk : in  STD_LOGIC;
       Row : in  STD_LOGIC_VECTOR (3 downto 0);
                             Col : out  STD_LOGIC_VECTOR (3 downto 0);
       DecodeOut : out  STD_LOGIC_VECTOR (3 downto 0));
end Decoder;

architecture Behavioral of Decoder is

signal sclk :STD_LOGIC_VECTOR(19 downto 0);
begin
         process(clk)
                  begin
                  if clk'event and clk = '1' then
                           -- 1ms
                           if sclk = "00011000011010100000" then
                                    --C1
                                    Col<= "0111";
                                    sclk <= sclk+1;
                           -- check row pins
                           elsif sclk = "00011000011010101000" then
                                    --R1
```

```
                if Row = "0111" then
                        DecodeOut <= "0001";          --1
                --R2
                elsif Row = "1011" then
                        DecodeOut <= "0100"; --4
                --R3
                elsif Row = "1101" then
                        DecodeOut <= "0111"; --7
                --R4
                elsif Row = "1110" then
                        DecodeOut <= "0000"; --0
                end if;
                sclk <= sclk+1;
        -- 2ms
        elsif sclk = "00110000110101000000" then
                --C2
                Col<= "1011";
                sclk <= sclk+1;
        -- check row pins
        elsif sclk = "00110000110101001000" then
                --R1
                if Row = "0111" then
                        DecodeOut <= "0010"; --2
                --R2
                elsif Row = "1011" then
                        DecodeOut <= "0101"; --5
                --R3
                elsif Row = "1101" then
                        DecodeOut <= "1000"; --8
                --R4
                elsif Row = "1110" then
                        DecodeOut <= "1111"; --F
                end if;
                sclk <= sclk+1;
        --3ms
        elsif sclk = "01001001001111100000" then
                --C3
                Col<= "1101";
                sclk <= sclk+1;
        -- check row pins
        elsif sclk = "01001001001111101000" then
                --R1
                if Row = "0111" then
```

```
                                DecodeOut <= "0011"; --3
                        --R2
                        elsif Row = "1011" then
                                DecodeOut <= "0110"; --6
                        --R3
                        elsif Row = "1101" then
                                DecodeOut <= "1001"; --9
                        --R4
                        elsif Row = "1110" then
                                DecodeOut <= "1110"; --E
                        end if;
                        sclk <= sclk+1;
                --4ms
                elsif sclk = "01100001101010000000" then
                        --C4
                        Col<= "1110";
                        sclk <= sclk+1;
                -- check row pins
                elsif sclk = "01100001101010001000" then
                        --R1
                        if Row = "0111" then
                                DecodeOut <= "1010"; --A
                        --R2
                        elsif Row = "1011" then
                                DecodeOut <= "1011"; --B
                        --R3
                        elsif Row = "1101" then
                                DecodeOut <= "1100"; --C
                        --R4
                        elsif Row = "1110" then
                                DecodeOut <= "1101"; --D
                        end if;
                        sclk <= "00000000000000000000";
                else
                        sclk <= sclk+1;
                end if;
        end if;
end process;
```

Table A.3 - Code for 7-Segment Display (Component to Main File)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity DisplayController is
   Port (
                        --output from the Decoder
                        DispVal : in  STD_LOGIC_VECTOR (3 downto 0);
                        --controls the display digits
                        anode: out std_logic_vector(3 downto 0);
                        --controls which digit to display
          segOut : out  STD_LOGIC_VECTOR (6 downto 0)
          );
end DisplayController;

architecture Behavioral of DisplayController is

begin
          with DispVal select
                    segOut <=  "1000000" when "0000", --0
                                        "1111001" when "0001", --1
                                        "0100100" when "0010", --2
                                        "0110000" when "0011", --3
                                        "0011001" when "0100", --4
                                        "0010010" when "0101", --5
                                        "0000010" when "0110", --6
                                        "1111000" when "0111", --7
                                        "0000000" when "1000", --8
                                        "0010000" when "1001", --9
                                        "0001000" when "1010", --A
                                        "0000011" when "1011", --B
                                        "1000110" when "1100", --C
                                        "0100001" when "1101", --D
                                        "0000110" when "1110", --E
                                        "0001110" when "1111", --F
                                        "0111111" when others;

end Behavioral;
```

Table A.4 - Constraint File for all Code

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
        set_property IOSTANDARD LVCMOS33 [get_ports clk]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports {digit_count[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {digit_count[0]}]
set_property PACKAGE_PIN V16 [get_ports {digit_count[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {digit_count[1]}]
set_property PACKAGE_PIN W16 [get_ports {door}]
        set_property IOSTANDARD LVCMOS33 [get_ports {door}]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {led}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led}]
set_property PACKAGE_PIN U19 [get_ports {ledDoor}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ledDoor}]
set_property PACKAGE_PIN W18 [get_ports {alarm}]
        set_property IOSTANDARD LVCMOS33 [get_ports {alarm}]

#7 segment display
set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]

set_property PACKAGE_PIN U2 [get_ports {an[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]

#Pmod Header JA
#Sch name = JA1
set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
#Sch name = JA2
set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
#Sch name = JA3
set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
#Sch name = JA4
set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
#Sch name = JA7
set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
#Sch name = JA8
set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
#Sch name = JA9
set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
#Sch name = JA10
set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```