Frankensafe: Team 2

Team Members (left-to-right on picture, above)	*Class No.*	*Lab Div*
Katrina Leffel	6689-L	2
Jordan Cecil	9410-C	1
Htoo Thein	6775-T	2
Michael Monaghan	9557-M	2

Report/Functionality Grading Criteria	*Points*
Originality, creativity, level of project difficulty	20
Technical content, succinctness of report	10
Writing style, professionalism, references/citations	10
Project functionality demonstration	20
Overall quality/integration of finished product	10
Effective utilization of microcontroller resources	10
Significance of individual contributions*	20
Bonus Credit Opportunities	*Bonus*
Early completion	0.5%
PCB for interface logic	2%
Poster (required for Design Showcase participation)	1%
Demo video (required for Design Showcase participation)	1%
Design Showcase participation (attendance required)*	1%

**scores assigned to individual team members may vary*

Grading Rubric for all Criteria (Including Bonus)	*Multiplier*
Excellent – among the very best projects/reports completed this semester	1.0 - 1.1
Good – all requirements were amply satisfied	0.8 - 0.9
Average – some areas for improvement, but all basic requirements were satisfied	0.6 - 0.7
Below average – some basic requirements were not satisfied	0.4 - 0.5
Poor – very few of the project requirements were satisfied	0.1 - 0.3

TABLE OF CONTENTS

1.0 Introduction

For our mini-design project we made a digital safe that could be used in a hotel or dorm room. The safe should be small and not extremely heavy but also sturdy enough to protect its contents. The user is allowed to pick their own passcode that is at maximum eight digits long to program the safe. At any time while the safe is unlocked they should be able to clear out the passcode and return the safe to default settings. There is also a debug mode of the safe that tests the functionality of the keypad, motor, speaker and various power readings throughout the hardware. This debugging mode was added to help the user have an easier time making sure that the safe is functional and ready for use. This will also help the user understand when and why the safe isn't working and prevent the user from trying to use a malfunctioning safe. The safe utilizes a LCD display, keypad, stepper motor, magnetic reed switch, speaker, and of course a PCB and Microcontroller.

We used many of the modules on the microcontroller like the PWM, TIM, RTI, ATD, and SCI. The microcontroller is programmed to receive a password from the user to unlock the safe. The alarm will go off in the safe if too many incorrect password attempts have been entered. The reed switch is used to determine if the safe door is open and will not lock unless the safe is closed. There is a programmed master code to unlock the safe at any point for easier use at a hotel. If the user fails to unlock a locked safe in five tries, the safe automatically flushes out the user programmed passcode and permanently locks the safe until the programmed bypass code is entered. Should the safe be unplugged or lose power for any reason, it does not automatically unlock on power up and would need the bypass code to unlock it.

We split the initial design of the safe into four parts, the designing of the printed circuit board for neater, cleaner design, interface a more sophisticated LCD to scroll messages, interface a keypad for input and then interface a step motor to turn the lock. Katrina Leffel took charge of the printed circuit board, Htoo Thein did the LCD interfacing, Jordan Cecil worked on the keypad and Michael Monaghan worked on the motor functions. Eventually Htoo and Jordan collaborated on the main function of the microcontroller, the RTI, the ATD, TIM and PWM modules. Then the team as a whole worked together to debug the hardware and software and make sure the two functioned efficiently together.

2.0 Interface Design

A PCB board was utilized for this project in order to simplify the circuitry so the logic could neatly fit onto the lid of the safe as well as improve the overall reliability of the hardware. The PCB was originally designed with many components as possible inputs and outputs to the overall code. These included: LCD screen, Keypad, Stepper Motor, Closed Door Detect, Lock Detect, and Speaker, Voltage Divider input to ATD, GAL chips, and Regulated Power Supply.

The main functionality of the code was dependent on inputs and outputs from four key components: Regulated Power Supply, LCD screen, Keypad, and Stepper Motor. The Regulated Power Supply was based off of the supply found on our mounting boards for lab. It is able to take the unregulated input from the wall wart and convert it to the 5 V inputs needed for the Motor, Microprocessor, and Digital Components. Two parallel voltage regulators were used to isolate the noisy motor power from the cleaner digital power. Both the LCD and Keypad are used to prompt and interact with the user. The LCD functions exactly as programmed during our

362 Lab Experiments, with a GAL chip used as a shift register for data. We made slight addition of a ribbon cable to the LCD pins for easy connection to the PCB. The Keypad was a new component introduced to receive input lock codes from the user. It is a switch network that sends out a high signal to one of twelve output pins whenever the corresponding key is pressed. A GAL chip was used to encode the pressed key into a four bit number to send to the Microprocessor. The motor that was chosen was a 4-phase stepper motor. It was chosen due to the precise steps that could be driven to initiate lock and unlock states as well as being a low cost component with extensive documentation of use on the internet. A bent bar was attached to the motor shaft to create the locking mechanism so that clockwise or counterclockwise motion was the only thing needed to lock the safe.

On top of the main functionality, we were able to add several auxiliary features to improve the overall appeal of our project. The features we added were a Door Detect, Speaker, and Voltage Divider. The Door Detect feature runs off of a magnet switch that allows a high state when the two ends of the magnet are touching and low when they are apart. This was used as a user error check so that the safe cannot be locked while the lid was open presenting possible damages to the motor. The Speaker we used was a small simple speaker that was used as the alarm when the incorrect passcode is entered. Adding this feature gave our safe a “real world” aspect for possible burglars. One of the final auxiliary features we implemented were simple Voltage Detectors to provide power checks in our code. The three detectors utilized two resistors each that feed into the ATD converter and allowed for voltage checking on the regulated Motor, regulated Microcontroller, and raw input power lines.

Since this was a working project, several modifications were made to the PCB during our progress. As modifications and improvements to our software were made corresponding hardware changes were also implemented. This included the cutting of some copper traces as well as white-wiring some connections on the back of the board. Also, due to time restrictions, not all of the features were implemented on the first version of the PCB. This included the Lock Detect sensor which would have utilized an infrared emitter and sensor placed near the lock bar to detect a successful locking and unlocking of the safe.

If we were to implement another version of the PCB all of the cuts and white-wires would be added directly as well as a slight rearranging of the layout so that the ribbon cables and wires connecting inside the safe would be easily attached without crossing or twisting. The Regulated Power Supply would also be examined to possibly minimize heat dissipated. On top of those changes other motors would also be examined for use to deliver more power so as not to always stall out at the slightest resistance.

3.0 Microcontroller Resource Utilization

The software of the project used many of the microcontroller’s on-chip peripherals for their special modes in order to create an elegant, smart and efficient program. The software utilized the ATD, SCI, TIM, and PWM peripherals as well as the on-chip RTI to build a safe that has a two operating modes and have many added features beyond simple passcode entering, lock and unlock functions. Each port pin and peripheral used was careful considered for best functionality of the overall product.

The heart of the program is the on-chip RTI which sets all of the flags that drives the software. The safe is user driven, each operation such as saving an entered passcode or driving a

motor to lock and unlock the safe depends on the user's input into a keypad, which is interfaced with a GAL device that simply outputs four digital values to the microcontroller's port AD. The pins 0 to 3 on port AD are designated to digital inputs to read the keypad's GAL output as a four bit value. The RTI then samples this input roughly every two milliseconds and makes the necessary decisions based on existing conditions of the safe. Two milliseconds were chosen so that it would be fast enough to detect the fastest change in values fed in by the keypad when a user presses a button. In order to make the safe as easy as possible to use by a consumer, the RTI must not only transform the four digit inputs into an appropriate four bit value but must also set the appropriate operation flag based on previous flags set and existing conditions. The RTI not only samples the data input from the keypad interfaced through a GAL chip, it also samples the digital data from the door sensor. The door sensor is a magnet plus reed switch that sends a digital value 0 for open and 1 for close. This digital input is fed into port AD's pin 7 and the RTI samples this data as well and makes the appropriate decision. A value read by the RTI is deemed valid to set flags for the main program when it is different from the previous value read by the RTI, this is to erase the problem of the RTI sampling the same value more than once because of its fast sampling time. For example, if the user presses the key '5', a value of 0101 will be sent by the GAL device to pins 0 to 3 on port AD for as long as the key is pressed. The RTI then will read a value of five as the input more than once because the average person presses a key for longer than two milliseconds. In order to eliminate repetitive five's, the RTI will compare the current value read to the previous value and set flags only if the values are different.

Though pins 0 to 3 and pin 7 on port AD are utilized for digital input, pins 4, 5 and 6 are utilized in ATD mode to check the power at various points on the PCB. As per the course notes, this means enabling the ATD, bit 7 of register ATDCTL2 and setting ATDDIEN to 0x8F to set all but channels 4, 5, and 6 and digital inputs and DDRAD as 0 as all channels on port AD are input channels. The ATD runs without an interrupt flag, in non-FIFO mode with 8 bit resolution and samples every two ATD clock periods. Many of these settings are the default setting after reset as they work very well for our purpose, to check the power supply on command and simply give a value for the power at various points on the PCB. Because there is no need to continuously scan the input, it is set into single conversion sequence mode but will sample across three successive channels starting with channel 4. The non-FIFO mode means that the results will be stored in result registers 0, 1 and 2. The ATD module checks the voltage for the motor, door sensor and speaker.

The TIM module works together with the PWM module to create an 800 Hz square wave to the speaker inside the safe when enough tries have been reached. We chose 800 as it is both audible and loud enough to be heard when the safe is closed. The TIM initially disabled and only enabled if the user failed to unlock the safe in five tries. The PWM utilizes port T's channel 0 to output a square wave. The PWMCLK is set to clock A, with a pre-scalar of two and a clock period of 255.

The SCI module is used to display messages on the LCD via a GAL shift register. As per the reference pages for the SCI module and course notes, the SCI had a baud rate of 9600 and was set for program-driven operation. Pin 4 on Port B was set for output mode and Pins 2, 3, and 4 plus port T were used to drive the LCD.

4.0 Software Narrative

The software portion of the project handles all of the core functionality of the safe. This includes, but is not limited to, the ability to enter and reset passcodes, the locking and unlocking of the safe, setting up the hotel override passcode, interfacing of the keypad and LCD display, and the implementation of the debugging mode. The code is organized in a user-driven fashion, in which the setting of various flags determines the current state of the safe at any given time. Depending on the current state, the user is prompted with specific instructions or messages that take the form of either static or scrolling text on the LCD display. A major goal of the project was to make the safe as user-friendly as possible, so special care was taken to make sure the code would not break if keys were pressed at inappropriate times.

The main function of the code is an infinitely cycled flag loop. The program starts in a beginning state in which either the user can enter the special debugging mode or input a new passcode for the safe for standard operation. Error checking is done to ensure that the passcode is the proper length. If the passcode is entered successfully, the safe is locked and can only be opened with either the newly input code or the predefined hotel code. The code first checks to see if the combination entered matches the user-input code and then checks to see if it matches the hotel code. If either case fails a variable that stores the number of tries is incremented by one. If the number of tries reaches five, the alarm will sound and only the hotel code can be used to open the safe. If the user successfully opens the safe, they can either choose to relock it (assuming the safe door is closed) or change the current passcode, which resets the safe back to the beginning state. If the debugging mode is entered, the user will be taken through a series of steps to check that all of the core parts of the safe are functioning. First the user will be asked to press the keys in a specific order to ensure the keypad is working. Then the safe will attempt to lock and unlock itself. Afterwards, the speaker will play the alarm sound. Finally, power supply will be tested at three points on the PCB. If any of these four steps are not completed successfully, the safe will enter a special lockdown state. Once in the lockdown state no further actions can be taken, as the safe is not functioning properly.

5.0 Packaging Design

The “base” of the safe was a standard lock box purchased from Walmart. It was chosen due to its compact size which could be easily utilized in a dorm room or hotel closet. Another factor for choosing a lock box as the safe was its durability as well as original functionality. The metal allowed for secure fastenings for all of the needed components as well as the “first defense” against possible burglary.

To connect and mount all of the necessary components multiple screw holes were made using a drill press. A slightly larger hole was needed for input from the power supply as well as even larger holes for the ribbon cables to be fed into the inside of the safe. Most of our components included mounting holes already but those that didn’t were also drilled with appropriate spacing to ensure they could be mounted onto the safe. To protect components and maintain aesthetic appeal, plastic standoffs were used to slightly lift the necessary parts off the safe.

All of the necessary components for user interaction or power input were mounted on the outside of the safe. The LCD and Keypad were mounted on top of the safe for use of user

interaction with the ribbon cables being fed under the parts and into the safe. The power input was mounted on the left side to correspond to the location of power input to the PCB. The inner circuitry is all contained in the safe. The mounted components included the Motor, Door Detect, and PCB. In order to keep it out of the way the PCB was mounted onto the lid of the safe so the ribbon cables and power input could reach. The Motor was mounted at the front of the safe lining up with the lock hook on the lid. The connection of the lock bar was formed by a Sparkfun part specific for use with motors.

Since this was a working project several modifications were necessary during the construction of the project to maintain and improve functionality. The lock hook, which was originally an eye hook, was bent to allow a larger clearance area for the lock bar. The initial metal chosen for the lock bar was a steel rod, this proved very difficult to manipulate and bent into the correct shape. In the end an aluminum rod was implemented due to its ease of cutting and bending into the desired shape. The Lock Detect sensor was also decided to be not feasible in the given time and was never attached to the safe. Had it been implemented the emitter and sensor would have been mounted near the lock hook so that the connection would be interrupted when the lock bar moves into the lock position.

Given more time our team would have implemented several improvements to the second version of the packaging design. The motor cable would be rerouted so that when the safe is open they don't cut across the open space. The extra wire on the door detect would also be better routed to ensure no slack was present for catching. Another improvement we would have liked to add would be a shield panel for the PCB. This would either cover the exposed portion of the lid creating a "nest" for the PCB to sit into and covering the wires and ribbon cables that currently just hang into the safe. Another design of this would be to completely cover the PCB to prevent user tampering. This could be either transparent or opaque to show the "brains" behind the safe.

6.0 Summary and Conclusions

In our project we had many difficulties that we had to overcome. The main problems we encountered were interfacing all of the components together, learning how each piece worked best into our design, and managing power to all of the systems. We also had to determine which aspects of our design were required and which ones were unessential. After completing the required part of the project, we were able to find which extra parts of the design would be most useful to implement.

The first part we worked on was the keypad. Since it had many outputs we decided to interface it through a Gal Device to lower the inputs to the microcontroller. This seemed like a simple task at first. However, the more we worked we found it hard to implement it through the Gal Device to the microcontroller since it wouldn't drop the voltage after being pressed. After a little work though we found that if we connected a resistors to the Gal Device inputs it dropped the voltage after being pressed. Also we were originally going to have the Gal device clocked, but we found it was unnecessary to do so, which was our first small revision to the design.

The next step of our design was to interface the LCD display to the microcontroller. We already had much of the code from previous labs to help us with this part. However, we did run into some issues with some of the messages we wanted to display being more than the 16

character slots available on each line. To fix this problem we made a scrolling display to get us up to 40 characters on a line.

This led way for us to start the core part of our program. In here we had to make a program that would receive a password from the keypad, and to unlock for that password or the hotel password which would allow us to unlock if someone forgot their password. We also had to determine all of the logic for how the interface should work. Then we had to make our first major changes to our original design. The issue we had was our LCD was connected to the PWM/TIM outputs and the motor outputs connected to the SCI outputs. We had to switch a few of the outputs around on our design and make some small changes to our PCB which we already had started which led to the next component.

The motor was out last essential part to the safe. We had a bit of difficulties trying to operate the motor through the microcontroller at first. We found that the motor ran better at half step than at full step. Also we had to determine how far we would have to rotate the motor to lock and unlock the safe. We got the motor to work, but had another issue with it. The motor would not always rotate when the function was called. After a bit of practice we found that we needed initialize the outputs to the motor so that the inductors didn't charge while the motor was not running. This seemed to have fixed the problem. If we had more time we may have found a motor that does not fully rotate which would stop any potential problems with over rotating.

After this was finished we started working on some extra features that would improve the safe. The first is door detect which was which was useful so that the use doesn't try to lock the door while the safe is open. The next part was the speaker which sounded an alarm if the user attempted too many unlocks and did not use the hotel password. These extra features slightly improved our safe and gave us the opportunity to learn how to operate more components.

In conclusion, we learned a lot from this project, and definitely became more aware of the issues with digit design. If we had more time to work on the safe we would have redesigned our PCB board so that we wouldn't have any wire connections on any parts of the board. Also we may have found a motor that doesn't fully rotate so that we would not have any potential problems with over rotating. Also we would have made a cover for the safe microcontroller in the safe so that the connections don't get loose or problems with items getting tangled in the wires.

References

D. G. Meyer. (2016). *Microcontroller-Based Digital System Design Module 3* [Online]. Available FTP: https://engineering.purdue.edu/ece362/Notes/PDF/3-Mod3_CP.pdf

Motorola, Inc. 21 Feb. 2003. *ATD_10B8C Block User Guide V02.10* [Online]. Available FTP: *https://engineering.purdue.edu/ece362/Refs/9S12C_Refs/S12ATD10B8CV2.pdf*

Motorola, Inc. 16 Apr 2004. *HCS12 Serial Communications Interface (SCI) Block Guide V02.08* [Online]. Available FTP: https://engineering.purdue.edu/ece362/Refs/9S12C_Refs/S12SCIV2.pdf

Motorola, Inc. 14 Mar 2002. *PWM_8B6C Block User Guide V01.14* [Online]. Available FTP: *https://engineering.purdue.edu/ece362/Refs/9S12C_Refs/S12PWM8B6CV1.pdf*

Motorola, Inc. 04 Feb. 2003. *SPI Block Guide V03.06* [Online]. Available FTP: https://engineering.purdue.edu/ece362/Refs/9S12C_Refs/S12SPIV3.pdf

Motorola, Inc. 11 Oct 2001. *TIM_16B8C Block User Guide* [Online] Available FTP: https://engineering.purdue.edu/ece362/Refs/9S12C_Refs/S12TIM16B8CV1.pdf

28 July 2012. *Stepper Motor 5V 4-Phase 5-Wire & ULN2003 Driver Board Arduino* [Online]. Available FTP: geeetech.com/wiki/index.php/Stepper_Motor_5V_4-Phase_5-Wire_%26_ULN2003_Driver_Board_for_Arduino

Appendix A:

Individual Contributions and Activity Logs

Activity Log for: Htoo Thein **Role:** LCD Interface, Software Overview

Activity	*Date*	*Start Time*	*End Time*	*Time Spent*
Present Initial Project Ideas and Decide on Major Task Breakdowns	3/5	3:00 PM	4:00 PM	1 Hour
Initial project checkoff; office hour approval	3/9	3:30 PM	3:45 PM	15 Min
After break check-in; reevaluation of tasks and deadlines	3/24	2:30 PM	3:00 PM	30 Min
Start programming the LCD to scroll to display messages longer than 16 characters	3/26	3:00 PM	4:00 PM	1 Hour
Programmed GAL device to send the appropriate data from the keypad --- Jordan	3/28	1:30 PM	3:00 PM	1.5 Hours
Tried interfacing the keypad to the GAL encoding --- Jordan	3/31	1:00 PM	3:00 PM	2 Hours
More attempts at interfacing the keypad to the GAL, worked on the main function	4/2	2:30 PM	5:00 PM	2.5 Hours
Worked on the software structure for the microcontroller, designated necessary peripherals for functions	4/3	2:00 PM	4:00 PM	2 Hours
Worked on connecting the keypad via the GAL to the microcontroller -- Jordan	4/6	1:30 PM	3:30 PM	2 Hours
Built the first prototype, a simple keypad and LCD display with a microcontroller	4/7	1:00 PM	3:00 PM	2 Hours
Worked on the RTI and main function for the software	4/8	3:00 PM	4:30 PM	1.5 Hours
Built a fully functional prototype on the breadboard, “Frankenstein” of the keypad working with the LCD and the microcontroller --- Jordan	4/9	2:00 PM	5:00 PM	3 Hours
Debug initial hardware prototype and a nearly complete main function for normal safe mode -- Jordan	4/13	1:00 PM	3:30 PM	2.5 Hours
Continued debugging the main function for normal safe mode --- Jordan	4/15	2:00 PM	6:30 PM	4 Hours
Re-interfaced the keypad to the GAL due to hardware/PCB changes	4/19	11:30 PM	1:30 PM	2 Hours
Worked on the code to detect the door sensor	4/20	1:00 PM	3:20 PM	2.33 Hours
Reworked the main structure of the software for a more elegant and sophisticated product	4/21	1:00 PM	3:00 PM	2 Hours
Finished software with motor functions added.	4/21	6:00 PM	8:00 PM	2 Hours
Added code for the ATD module for power checks and TIM and PWM for speaker	4/23	2:45 PM	4:30 PM	1.75 Hours
Debug the main software for functionality	4/24	4:30 PM	7:00 PM	2.5 Hours

with the final version of the safe				
Added code for the entire debugging module	4/24	8:30 PM	12:30AM	4 Hours
Went to lab ahead of team to debug the debugging module code and ATD function	4/25	10:00 AM	11:30 AM	1.5 Hours
Finalized and debug the entire software, wrote the code for TIM and PWM modules in order to have the speaker added, general debugging of the hardware and safe design	4/25	11:30 AM	6:30 PM	7 Hours
Worked on poster, flowchart	4/25	9:30 PM	10:30 PM	1 Hours
Worked on essay portions, more software flowcharts	4/28	1:00 PM 6:30 PM	2:00 PM 10:30 PM	1 Hour 4 Hours

Written Summary of Technical Contributions: Htoo Thein

My initial portion of the project was to simply interface a more sophisticated LCD display to the microcontroller, so that it can scroll and such. This task turned out relatively simple and took only a few hours so I began working on the main function while the rest of the group continued working on their respective tasks. Because the main functionality of the software is the user entering an input from the keypad and then the microcontroller initiating the proper protocols and displaying the appropriate messages on the LCD display. To get working on this, I began helping on interfacing the keypad to the microcontroller through a GAL device and made significant contributions. We set the GAL to send 1111 (15) when no buttons were pressed, each digit key was simply assigned their values and the (*) key was assigned 1100 and (#) key assigned (1101). These four digital outputs from the GAL device are sent to the port AD pins 0 -3. After interfacing the keypad to the microcontroller, I then wrote the code for the RTI to sample the data and set the necessary flags for our main function. Jordan Cecil and I worked closely together as we strived to debug much of the main software to get it fully working with a simple keypad and LCD interfaced to the microcontroller. We built a hardware prototype with breadboards to continue working and debugging while the PCBs were being finalized.

I also wrote the code and initialized all the values for the ATD converter that samples three points on the PCB and check that the motor, door sensor and overall PCB were receiving the appropriate supplies after a voltage divider. I wrote the code for the door detect, which functions as a digital input in port AD pin 7, the sensor sending a digital 1 or 0 depending on state and then I utilized the RTI to sample that value, much like how the RTI samples the values sent by the keypad GAL device. I also wrote the code for the entirety of the debugging mode, which takes the user through various hardware components (keypad, motor, and speaker) and check three points on the PCB to ensure a fully operating PCB. I also wrote the code for the TIM and PWM module to produce a square wave for the speaker when necessary.

Activity Log for: Katrina Leffel **Role:** PCB and Hardware Design

Activity	*Date*	*Start Time*	*End Time*	*Time Spent*
Eagle Schematic Tutorial	2/20	8:00 PM	8:30 PM	30 Min
Present initial project ideas and decide on major task breakdowns	3/5	3:00 PM	4:00 PM	1 Hour
Initial project check off; office hour approval	3/9	3:30 PM	3:45 PM	15 Min
PCB Design - Choose parts and schematic	3/15	1:00 PM 3:30 PM	2:00 PM 4:30 PM	2 Hour
PCB Design - Begin board layout and wiring	3/16	2:00 PM 4:00 PM	3:00 PM 5:00 PM	2 Hour
PCB Design - Continue board layout and wiring with modifications to schematic as needed	3/17	11:30 AM 2:00 PM	12:30 PM 3:00 PM	2 Hour
Error checking and size/layout checking	3/18	2:00 PM	3:00 PM	1 Hour
Final checks and ordering	3/19	2:30 PM	3:30 PM	1 Hour
After break check-in; reevaluation of tasks and deadlines	3/24	2:30 PM	3:00 PM	30 Min
Board inputs and wire preparations and lock box design and cuts	3/27	2:00 PM	3:00 PM	1 Hour
Soldering board 1 - power supply, pull-down resistors, and GAL chip nests	4/2	3:00 PM	5:00 PM	2 Hours
Soldering board 1 - MP standoff and connecting pins for ribbon cables	4/6	3:00 PM	4:00 PM	1 Hour
Code test for keypad and LCD with board 1 Soldering board 1 - rework: cuts and white-wires	4/7	3:00 PM 6:00 PM	4:00 PM 7:00 PM	2 Hours
Begin work on code for motor control	4/13	3:00 PM	4:00 PM	1 Hour
Soldering board 1 - motor chip standoff and current draw testing with rework Continue work on motor control code	4/15	1:30 PM 3:00 PM	2:30 PM 4:00 PM	2 Hours
Code test for keypad and LCD with board 1 reworks (additional rewire after meeting)	4/19	11:30 AM 9:00 PM	1:30 PM 9:30 PM	2.5 Hours
Addition and testing of motor code	4/20	1:30 PM	3:00 PM	1.5 Hours
Testing board 1 with lock box and addition of auxiliary functions	4/21	6:15 PM	8:00 PM	1.75 Hours
Report Work Soldering of boards 2 and 3	4/24	12:00 PM 3:00 PM	12:30 PM 8:00 PM	5.5 Hours
Testing of boards 2 and 3 and report work	4/25	4:30 PM	7:00 PM	2.5 Hours
Debugging of auxiliary code and components; report work	4/26	11:30 AM	5:00 PM	5.5 Hours

Written Summary of Technical Contributions: Katrina Leffel

My main role on the team was packaging design and PCB lead. From the start of our project I knew I really wanted to use a PCB to hold the main logic of our program because of its compact nature and better reliability than typical breadboard wirings. As I was laying out the PCB I also found it helpful to have a general idea where in our final project everything would be oriented to have clear connections to all off board components. Due to this I also prepared the final safe to be used in our project.

To design the PCB I used the Eagle software. First I setup the schematic including all the connections for possible accessories I thought we may use in our safe. This included ribbon cable connectors for the Keypad and LCD, GAL chip standoffs, power regulator design, connectors for our MP and motor chips, door detect, lock detect, and speaker inputs. The board layout was designed with the positions of all the external components in mind. One of the hardest parts of the board layout was the wiring of the Keypad and LCD ribbon cable connections. In the end several reworks were needed in order to fit everything on the board easily. While soldering the first board I went through and current tested as each component was added to make sure there weren't any components overdrawing current. During work on the first board there were also several changes that were made to the design and wiring of the board that required cuts and white-wires. The original redesigns were done with an exact knife and wires running across the back of the board. Once we had tested the first board and agreed upon the revisions to make I started work on the second and third boards. Since the testing and redesign layouts were already complete these boards took less time and had better layout of wires and cuts. During work on the third board I ran out of through hole 10 ohm resistors so the pull-down resistors for the Keypad were changed to surface mount resistors. This was a form of soldering that I hadn't done before and I was grateful for the learning experience.

Since I knew the most about the PCB design, I was in charge of error checking the boards anytime something went wrong with our programs. We were also testing our code on a breadboard that simulated the connections on the PCB to help figure out if it was a hardware error of software error. Several of the hardware errors included wrongly sized capacitors or poorly soldered joints. One of our biggest struggles was getting the speaker to work at an audible volume and correctly function in the PCB. The cause of the problem ended up being a missing ground wire after a redesign of the speaker circuit.

The safe design was based on a typical lock box found at Walmart. Since the board layout was already known, adjustments to the safe were made based on the needed connections to external components. All of the holes were made with a drill press and measured so that components could fit on and through the safe. One of the hardest parts of the safe design was figuring out how to get power from the wall cable to the inside of the safe. This was necessary because the PCB was to be mounted on the lid of safe to ensure it wasn't damaged during motor operation but could also be displayed. After mounting the motor and motor hook, I bent the motor arm from a rod of aluminum so that the locking mechanism was formed.

Once the boards and safe were constructed my role changed to assisting with the motor control. Since a 4-Phase stepper motor was used, the control only required the changing of four data lines from high to low. The hardest part of this task was figuring out the correct order of phase excitation, but the motor was relatively easy to control once this was complete.

Activity Log for: Jordan Cecil **Role:** Keypad Interfacing / Code Debugging

Activity	*Date*	*Start Time*	*End Time*	*Time Spent*
Present Initial Project Ideas and Decide on Major Task Breakdowns	3/5	3:00 PM	4:00 PM	1 Hour
Initial project check off; office hour approval	3/9	3:30 PM	3:45 PM	15 Min
After break check-in; reevaluation of tasks and deadlines	3/24	2:30 PM	3:00 PM	30 Min
Began initial work on interfacing the keypad via GAL chip	4/6	1:30 PM	3:30 PM	2 Hours
Cleaned up keypad functionality and helped debug initial code	4/8	3:00 PM	4:30 PM	1.5 Hours
Completed the first functioning prototype "Frankenstein"	4/9	2:30 PM	5:00 PM	2.5 Hours
Fixed minor hardware problems of "Frankenstein" and helped debug normal safe operation	4/13	1:30 PM	4:00 PM	2.5 Hours
Spent entire time debugging the code for normal safe operation; pretty much complete	4/15	2:00 PM	6:30 PM	4.5 Hours
Worked out new keypad outputs after PCB changes and began work with door detect	4/19	11:30 AM	2:30 PM	3 Hours
Tried learn how to implement flash memory	4/20	7:00 PM	9:30 PM	2.5 Hours
Helped to debug problem with the door detect code	4/21	1:00 PM	3:00 PM	2 Hours
Streamlined code, added more comments for easier navigation	4/24	1:30 PM	4:00 PM	2.5 Hours
Helped debug code with new peripherals; fixed a problem with the speaker	4/24	4:30 PM	6:00 PM	1.5 Hours
Helped with the last major debug of the code with all peripherals in place	4/25	11:30 AM	3:00 PM	3.5 Hours
Added summary of project and photos to poster	4/26	6:00 PM	7:00 PM	1 Hour
Filmed video for project	4/28	2:30 PM	4:00 PM	1.5 Hours

Edited video for project and worked on report	4/28	6:30 PM	10:00 PM	3.5 Hours

Written Summary of Technical Contributions: Jordan Cecil

At the start of the project, I was responsible for interfacing the keypad with the microcontroller. This involved a small amount of coding to program a 22V10C GAL chip and setting up the appropriate port pins on the microcontroller. I had to learn how our specific keypad worked, as this can vary across different models. Trying to figure out how the keypad button outputs were ordered on the ribbon cable was one of the tougher parts, as they were not ordered numerically as I assumed they would be. I performed several tests just to ensure that the keypad was functioning properly by hooking it up to a DMM and checking the output voltage when the keys were pressed. Once I was confident that the keypad was working, I helped construct the first prototype "Frankenstein" that was used to do most of our early tests before our PCB's arrived. This contained only the most basic components of our safe: the microcontroller, keypad, LCD display, and the GAL chips.

The keypad interfacing didn't take very long, so I began branching out into other areas of the project. I assisted in the early stages of setting up the motor and learning how it functioned, but most of my focus went towards the software component of the safe. I helped to debug nearly all of the code used in the project. This involved extensive work especially as our project became more advanced, requiring more states, flags, and variables. It was not uncommon that small changes to the code led to unexpected, larger problems with the safe. Before the PCB's arrived, I had to remake "Frankenstein" a few times, so we could continue testing the code. Using this first prototype was a pretty huge pain, because the ribbon cable output of the keypad made it difficult to wire up on a breadboard. Connections were loose and often came undone, leading to minor problems during tests.

After the PCB's were ready, I had to reconfigure the GAL code for the keypad in response to last minute changes to the board. I continued debugging the main code as more peripherals were added to the project. I worked with the speaker a lot, trying to get the PWM and TIM initializations correct in order to output sounds at the appropriate frequencies. I helped troubleshoot a problem in which it wasn't outputting any sound at all, suggesting that there was an issue with the PCB layout. I also helped extensively with the coding and restructuring of the "debug" operation of the safe.

I wrote sections of the report, and was responsible for editing the video for our project. I wrote the summary and added the photos of all the prototypes to the poster. I was also importantly responsible for coming up with the Frankenstein theme that eventually made its way into our project's name.

Activity Log for: Michael Monaghan **Role:** Peripheral Utilization

Activity	*Date*	*Start Time*	*End Time*	*Time Spent*
Present Initial Project Ideas and Decide on Major Task Breakdowns	3/5	3:00 PM	4:00 PM	1 Hour
Initial project check off; office hour approval	3/9	3:30 PM	3:45 PM	15 Min
After break check-in; reevaluation of tasks and deadlines	3/24	2:30 PM	3:00 PM	30 Min
Worked on programming the Gal 22V10 Devices	4/2	2:30 PM	5:00 PM	2.5 Hours
Attempted to interface the keypad through the Gal 22V10 Device	4/7	1:30 PM	3:00 PM	1.5 Hours
Worked on debugging keypad	4/13	3:30 PM	4:30 PM	1 Hour
Got the motor to rotate using the wave generator	4/15	3:30 PM	4:30 PM	1 Hour
Programmed the microcontroller to rotate The motor	4/16	1:00 PM	6:00 PM	5 Hours
Made functions to rotate the motor CW and CCW for roughly 90 degrees	4/22	12:00 PM	3:00 PM	3 Hours
Testing and attempting to configure the PWM for the alarm	4/25	4:30 PM	7:00 PM	2.5 Hours
Debugging, helping with PCB, and report	4/26	12:00 PM	5:30 PM	5.5 hours
Printing Poster	4/27	4:30 PM	5:30 PM	1 Hour
Report/Video	4/28	2:30 PM 10:00 PM	3:30 PM 1:00 AM	4 Hours

Written Summary of Technical Contributions: Michael Monaghan

My primary task on the team was to work on the 28BYJ-48 – 5V Stepper Motor. I spent some time on spring break researching how the motor is run, and tried to figure out how I could implement it on the microcontroller. I also spent some time during the beginning stages of the team helping get the keypad to interface through the Gal Device to the microcontroller. I also helped program the Gal Device and initial code. Then later I spent most of my time getting the motor to run through the microcontroller. The lastly spent time debugging our code, working on the report/poster, and got to help make some minor changes on the PCB.

During the first few meeting we set roles for the team and went over the general idea of how we wanted to design the project. When we started working on the project, I spent a bit of time helping programming the Gal Devices and setting up circuits to run the keypad to the microcontroller. This is mostly what I did the first few meetings.

Later on I started working on the stepper motor. I did some research on it before meetings to figure out the best way to run the motor on our microcontroller. Then I helped get it to run using two oscilloscopes to generate a waveform. After that I spent most of my time writing a program to run the motor through the microcontroller. This was a difficult task because I needed to make proper delays between steps in the motor and find which inductors to charge to get a full rotation. A few of the problems I ran into was getting the right delay from changing state, and figuring out that this motor ran better on half step than full step. After that I made another function to rotate the motor in reverse. This seemed simple at first, however, we ran into a problem with the motor stalling every once in a while in reverse. The problem was that the initializations weren't correct for the motor. Lastly we hard to determine how many steps we would have to take for the motor to lock the safe and unlock.

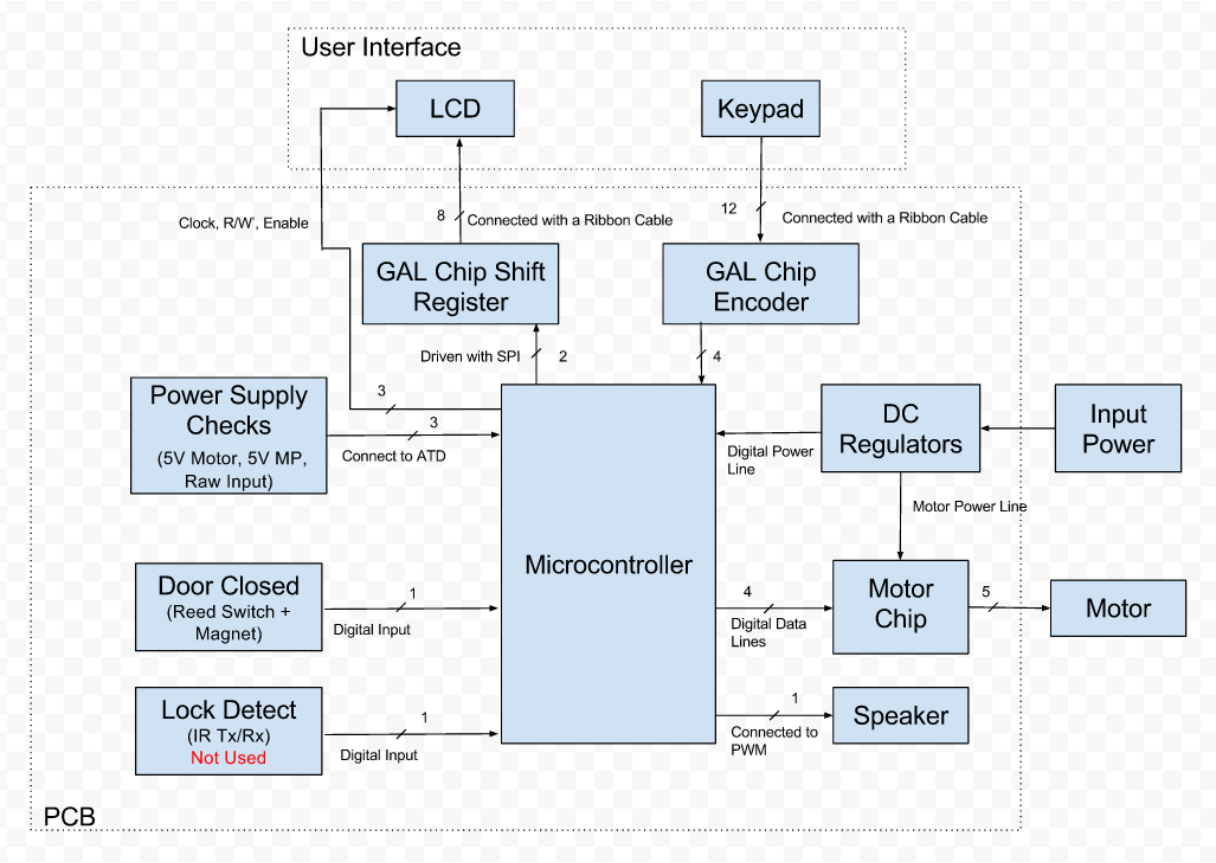
When this was done I started helping with the other components again. I spent a bit of time testing the speaker to find a good frequency for the alarm. Also since it was run through the PWM, I found a few of the initializations so that we could create that frequency. Then I helped with some more debugging of the ATD to measure the right voltage across our PCB.

I also got a little practice helping to solder a few capacitors and resistors onto the PCB, and remove a few as well. I thought this was a good experience and will help me know the basic knowledge for senior design. At the last meeting and in my spare time, I wrote part of the poster, and finished some of the report. I also added some pictures and notes to appendix D, and references most of the material we used for our data sheets.

Appendix B:

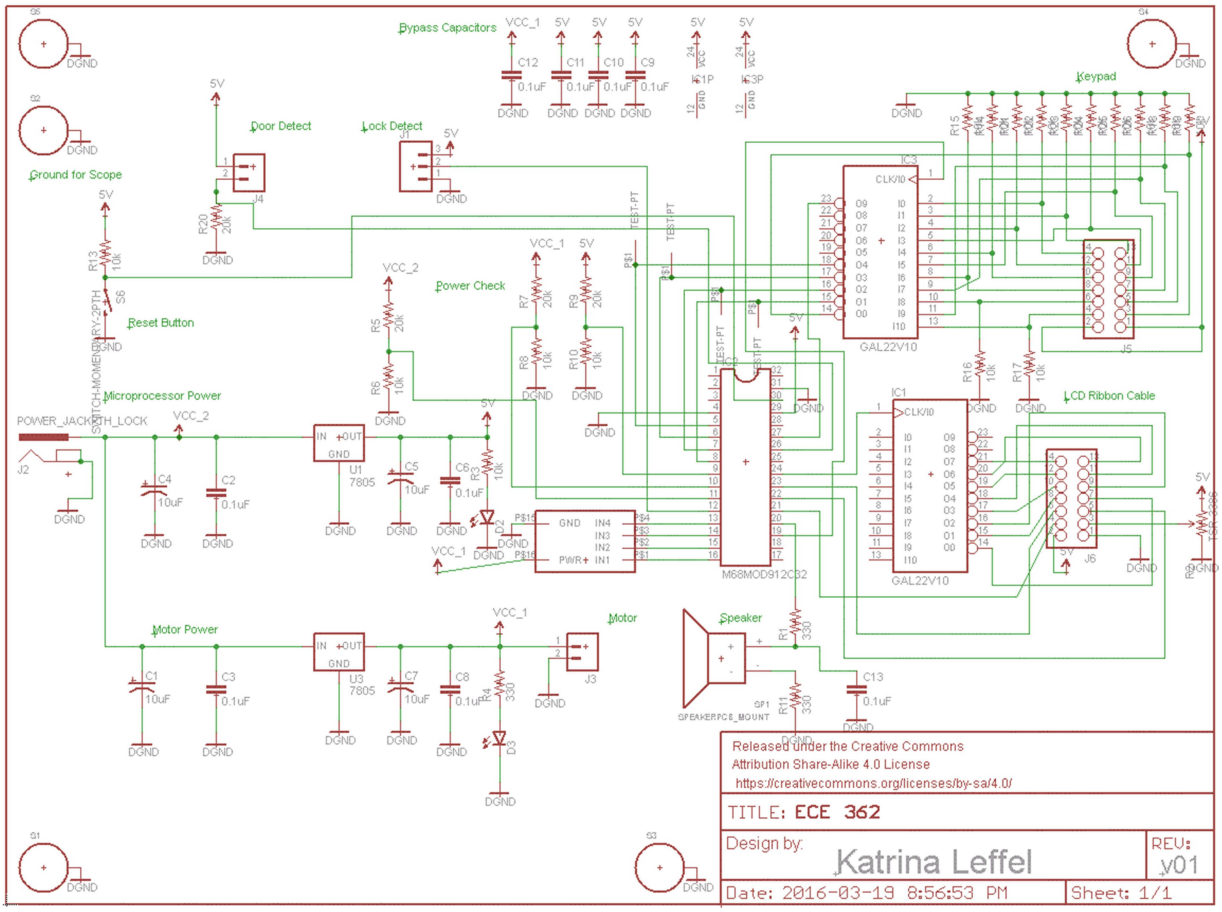
Interface Schematic and PCB Layout Design

Functional Block Diagram



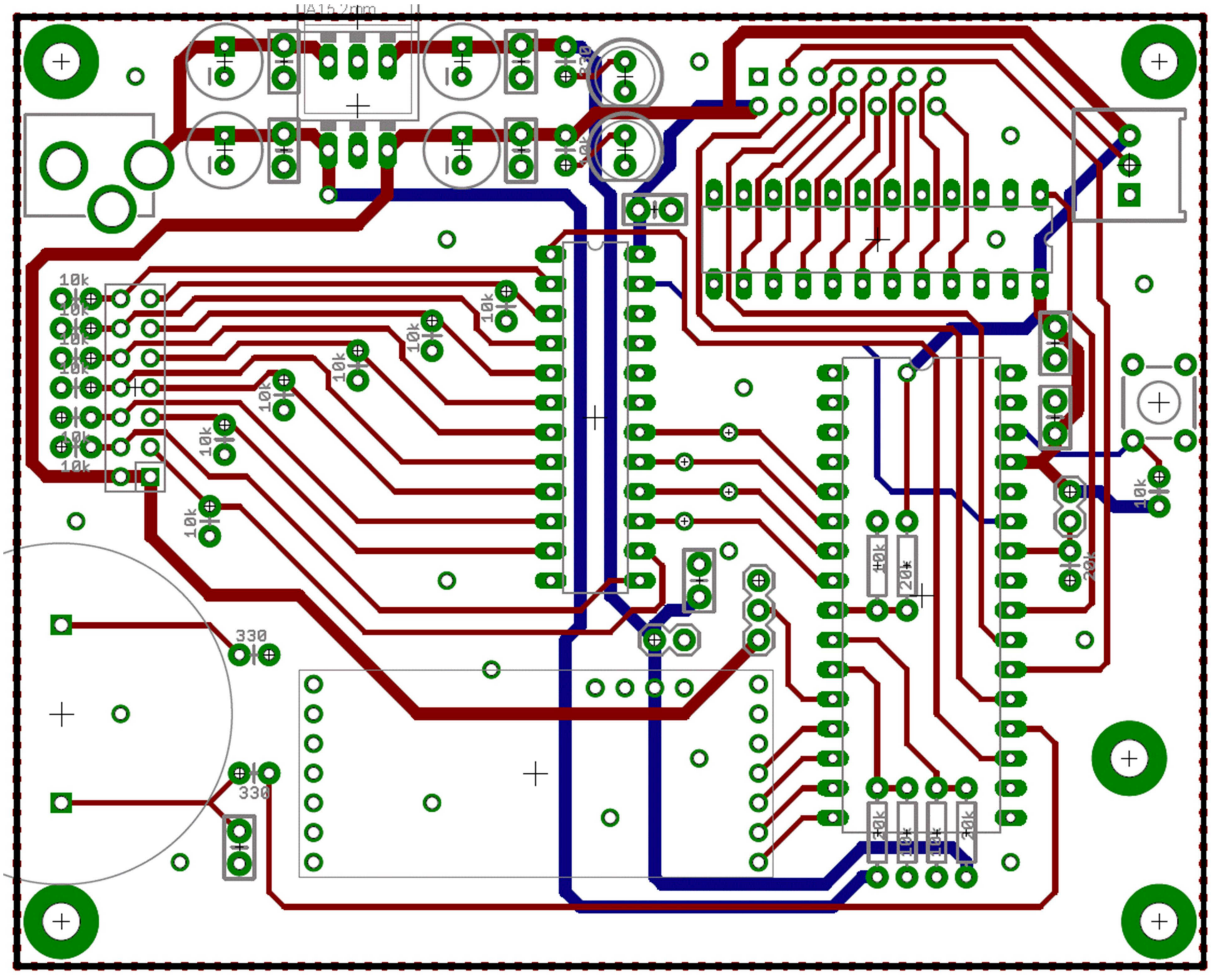
By: Katrina Leffel

Original Schematic



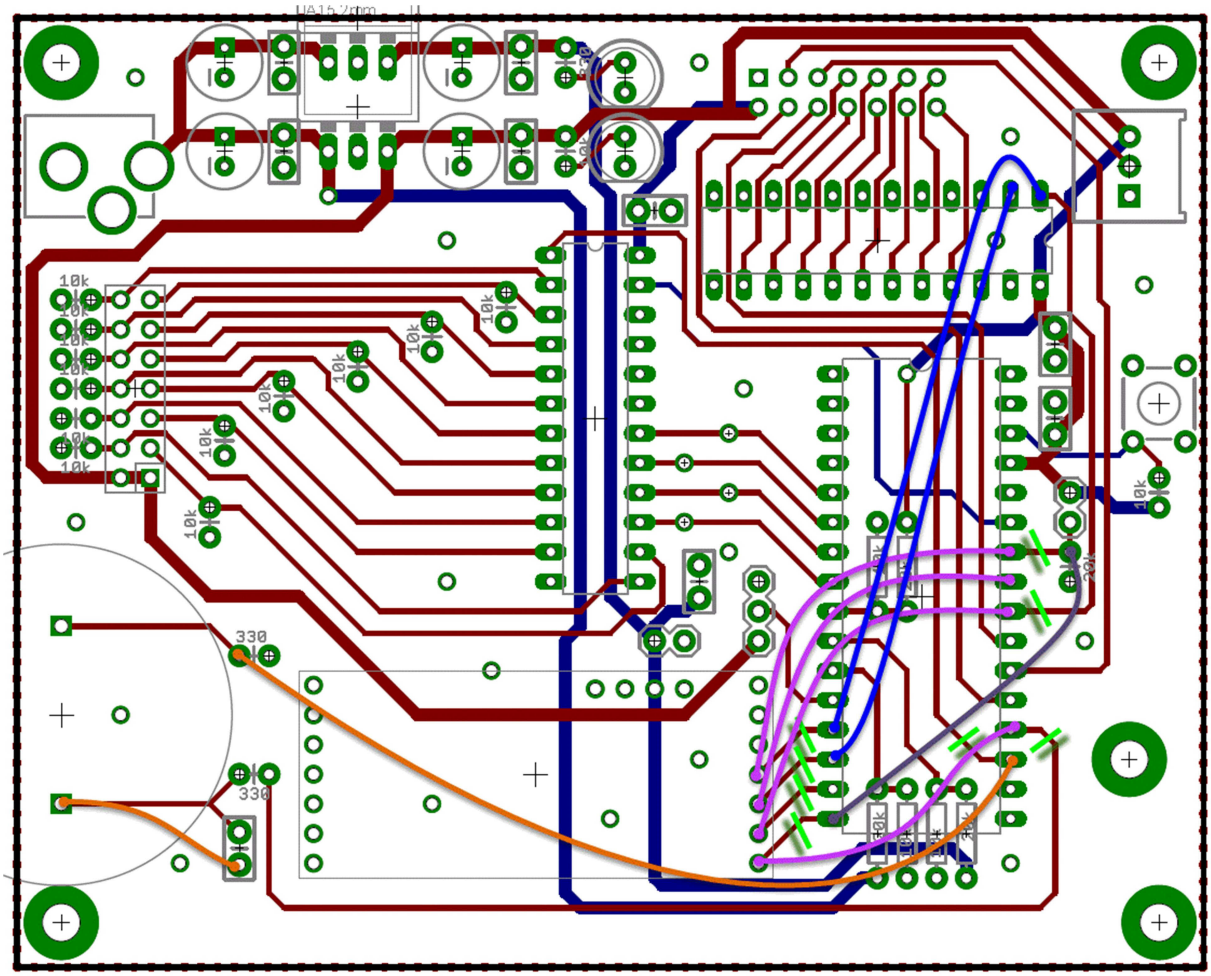
By: Katrina Leffel

Original Board Layout



By: Katrina Leffel

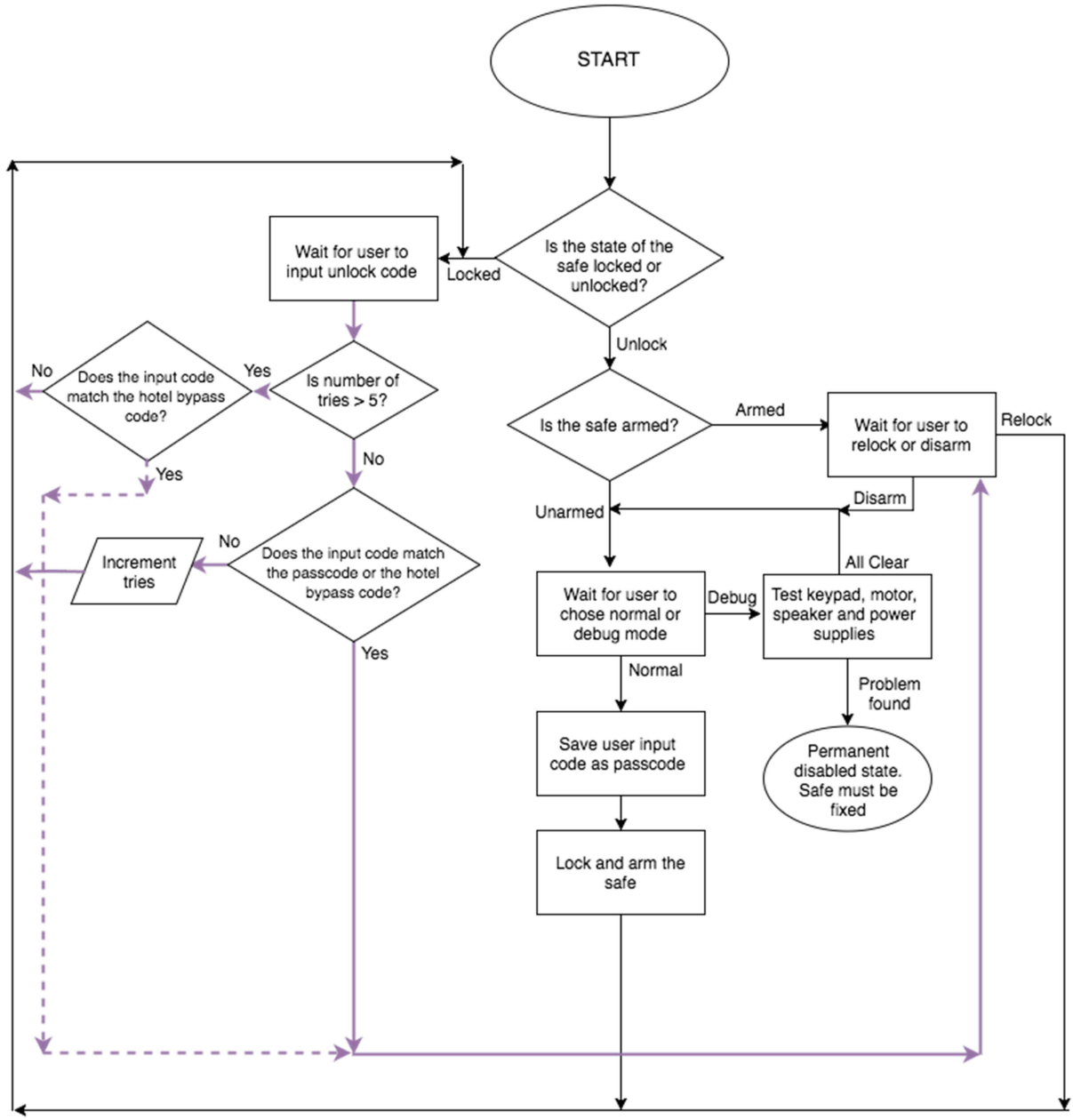
Board Layout with Cuts and Rewires



By: Katrina Leffel

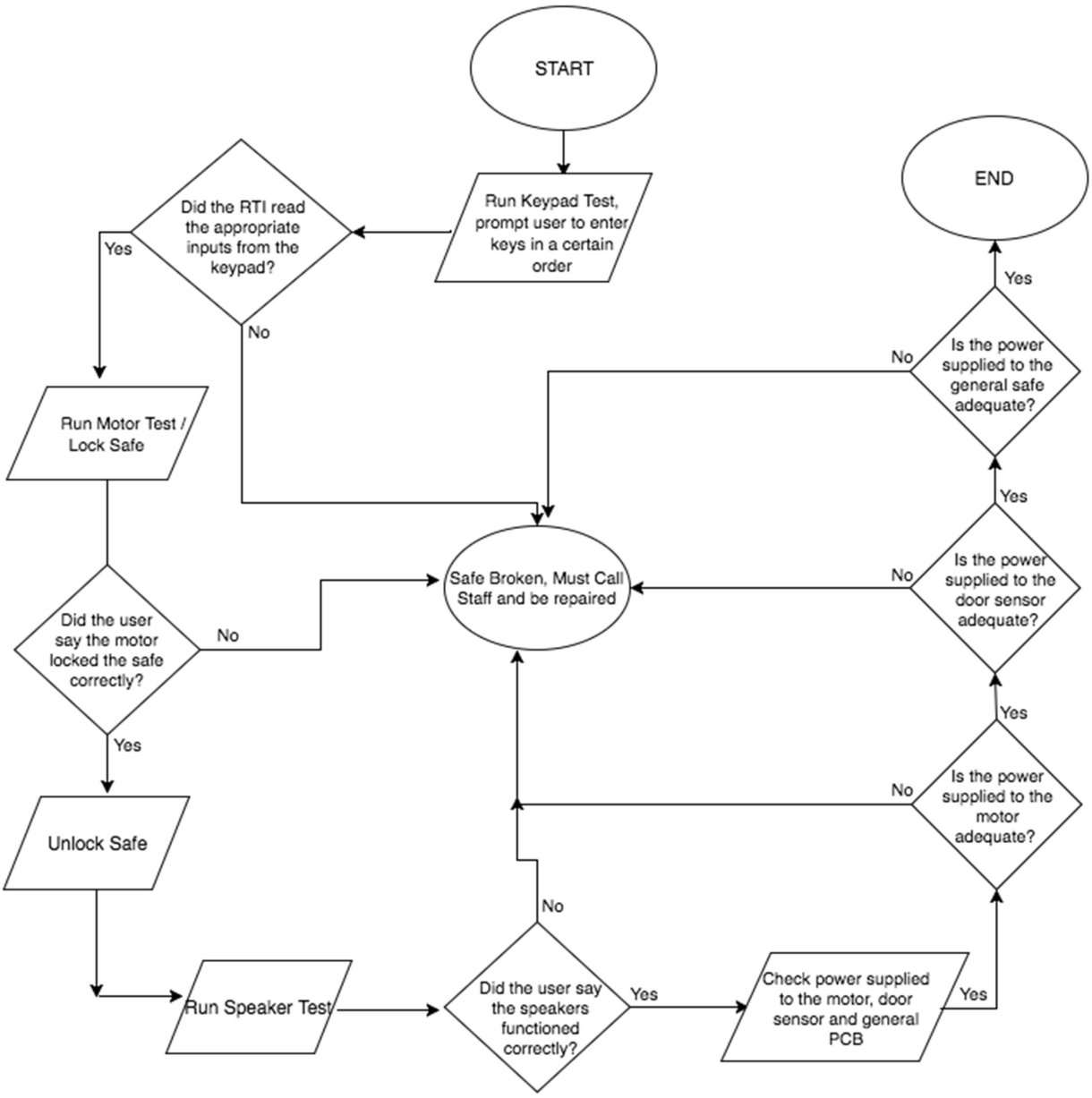
Appendix C:
Software Flowcharts

General Software Flowchart:



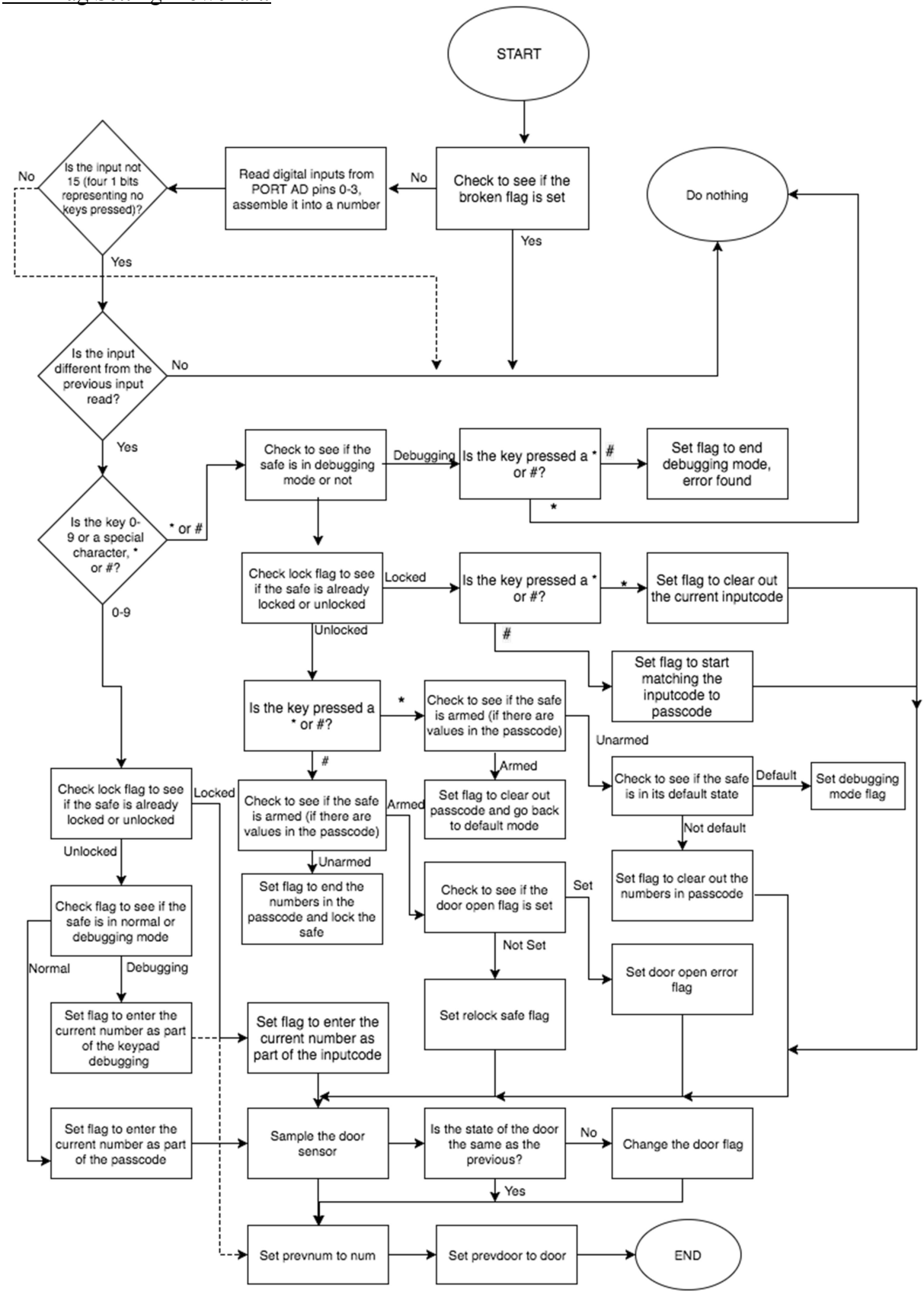
By: Htoo Thein

Debugging Mode Flowchart:



By: Htoo Thein

RTI Flag Setting Flowchart:



By: Htoo Thein

Main Function Pseudo Code:

passcode is the code the user programs to lock the safe, set to default value of {-1,-1,-1,-1,-1,-1,-1,-1}
inputcode is the code the user enters to unlock the safe once it is locked {-1,-1,-1,-1,-1,-1,-1,-1}
hotelcode is the code that always unlocks the safe {-1,-1,-1,-1,-1,-1,-1,-1}

```
 if (default mode) {
    scroll the welcome message prompting users to start entering their passcode or select to run
    debugging mode
}
if(the flag to enter the number sent by the keypad into the passcode is set) {
    if(the passcode has exceeded 8 digits) {
        display error message, clear out all the numbers added so far
    }
    else {
        add the number to the array storing the passcode
        display appropriate numbers entered so far
   }
}
if (the flag to lock the safe is set) {
   end the input of numbers in the passcode array, set flag to run the motor for locking
}
if(the flag to enter the number sent by the keypad into the inputcode is set) {
   if(the length of the inputcode has exceeded 8 digits) {
     display error message, clear out all the numbers added so far
   }
   else {
      Add the number to the array storing the inputcode
      display the appropriate numbers entered so far
    }
}
if(the flag to start matching the inputcode to the passcode is set) {
   for(go through the maximum length of passcode and inputcode: 8) {
      compare each digit in passcode to inputcode, if an unequal value is found clear the flag to
      match
   }
   if(the inputcode didn't match the passcode) {
      for(go through the inputcode and hotelcode) {
         compare each digit of both codes, if an unequal value is found, clear the flag to match
      }
   }
   if(the inputcode failed to match both the passcode and hotelcode) {
      increment the number of tries, display the error message, clear out the input code
      if(the number of tries is five) {
          set the passcode to the hotelcode so that the inputcode must match the hotelcode to
          unlock the safe now.
          display the appropriate warning message and turn on the alarm (speaker).
      }
   }
   else {
```

```
        call the unlock the safe function, display proper message, set tries back to 0.
    }
   clear out the flag and set the inputcode back to default
}
if(clear out the inputcode flag is set) {
    clear out the inputcode and display the appropriate message
}
if(clear out the passcode flag is set) {
    clear out the passcode and display the appropriate message, set back appropriate values to
    their initial values
}
if(the lock safe flag is set) {
    display the appropriate unlocking message, call the lock safe function
}
if(the flag to run the debugging mode is set) {
    display appropriate message, call the test keypad function
    if(the keypad function returned a failed keypad) {
        display error message, set the broken flag
    }
    else {
        test the motor by calling the lock function. Ask the user to test the safe.
        if(the user says the safe locked properly) {
            unlock the safe
        }
    }
    if(the user said the safe did not lock properly) {
        display error message, set the broken flag
    }
    else {
        rest the speakers, ask the user to verify that it is functioning properly
        if(the user said the speakers were broken) {
            display error message, set the broken flag
        }
    }
    if(the user said the speakers were functioning) {
        call the power check function
        verify that the power to the motor, door sensor and overall PCB are all satisfactory
        if(the power is not satisfactory) {
            display error message, set the broken flag
        }
    }
    if(the power is satisfactory) {
        display safe is functioning message, go back to default mode
    }
}
```

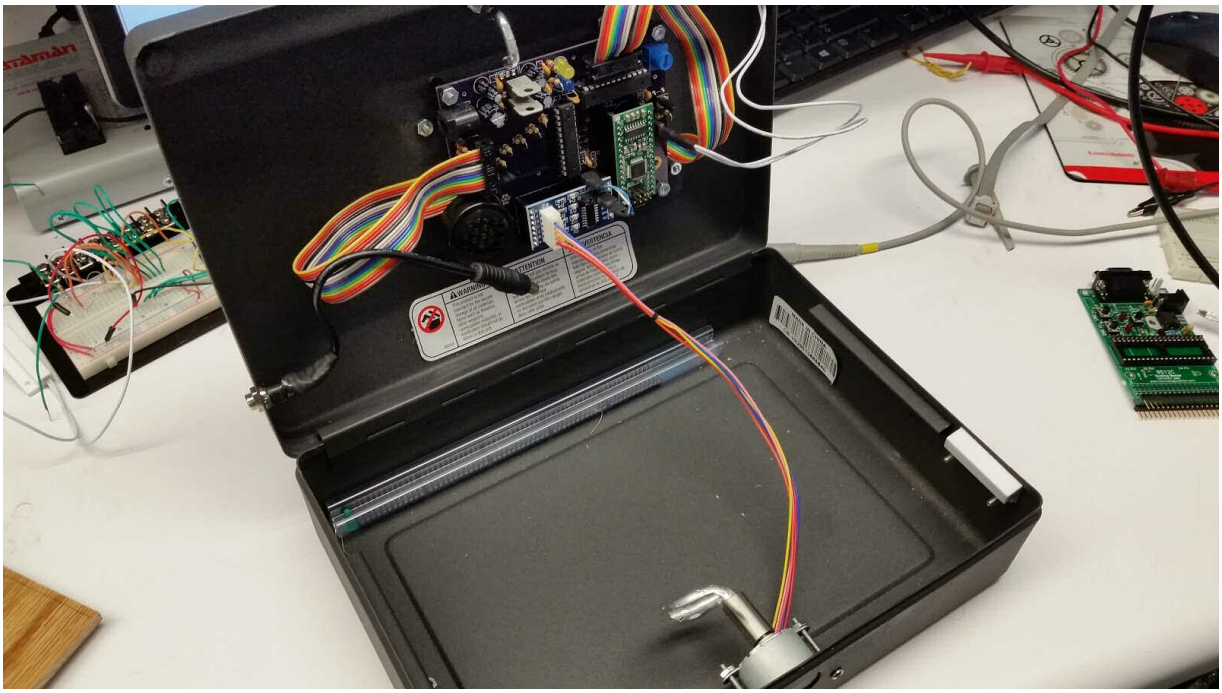
By: Htoo Thein

Appendix D:
Packaging Design

Parts List:

- Walmart Lock Box - $10
- Keypad (EBay) - $4
- Stepper Motor (EBay) - $3
- LCD (EBay) - $5
- 2x Ribbon Cables (EBay)- $4
- PCB (OSH Park) - $25
- 2x GAL Chips - $6
- Microprocessor - $30
- Various Screws
- Various Resistors
- Various Capacitors

Rough Cost: $87



This picture is of the inside of the safe. As can be seen the PCB is connected to the top of the lid, and the power is connected to the left side of the lid. The motor is connected to the front of the safe facing inward. Also the door detect sensor is connected to the right side of the case and lid.

This picture is of the outside of the safe. In the center the keypad is drilled into the safe and a larger slit is placed at the left side for the keypad cable to run to the PCB right beneath it. The LCD display is placed to the right side of the top of the lid. It is also drilled into the safe, and at the back side a larger slit is placed for the cable to run to the PCB right beneath it. Lastly in the front center a hook is inside the safe for the motor to connect to.