Final Project Technical Documentation:

Safe

Joshua Thao, Jonathan Benusa, Jack Schumacher

ENGR 331

University of St. Thomas

Executive Summary

In this project, a set of sensors and devices were given to implement into a final design by the project members’ choice. For this project, the goal was to make a safe implemented with a set of devices and sensors. The following would have an accelerometer to sense tampering or movement of the safe. A stepper motor that would act as a locking mechanism. The input device of a numeric keypad and an LCD as a display screen that would show the code input. A capacitive touch switch that would act as a reset button for the passcode. All these chosen sensors and input devices would be integrated into a conceptually functional safe.

The planning of the project began on December 1st, and the final prototype will be finished before December 19th. Much of the project was dedicated to programming the proper functions into the input devices and initializing the sensors, all of which were put into a single code for the safe. The final week and days would be dedicated to the integration of the sensors and devices to make a functional prototype.

Background

The purpose of this project is to build a meaningful device or system being given a set of devices. This required that the device or system somehow implemented a temperature sensor or accelerometer, an actuator, an LCD or terminal window, and at least two sensors or input devices of the eleven offered. The wiring of the device and system was meant to be set within an enclosure which meant that the wires could not stick out. The objectives and functions of the system or device were left to the project group’s choice.

The constraints were decided by the team as how the safe should act ideally. This safe is to use a four-digit code to open. The safe should ideally be 12’x12’x12’ inches. Battery usage of the safe should ideally last ½ a year under normal use. The safe operates using the ARM Cortex M4 processor. The processor would allow the safe to use the given tasks to sense movement with the accelerometer, communicate with an LCD, contain a button in the inside of the safe to allow the person to set a new passcode, and open with the correct code shown by the stepper motor or stay locked if the incorrect code has been input. The accelerometer would act to detect tampering of the safe. The safe would be relatively small, an accelerometer would be necessary to sound an alarm if the safe if someone were to take or move the safe in any way. The LCD would allow the person to see the state of the safe and the input code pressed on the numeric keypad. A button on the inside would allow the person to change the set code of the safe, which would allow for any four-number combination. This function would only work after the safe has been opened. Only if the correct code is input will the safe open which is indicated by the rotation of the stepper motor and lock again as indicated by the stepper motor. The incorrect code will not open the safe, it should remain locked until the correct input is put in.

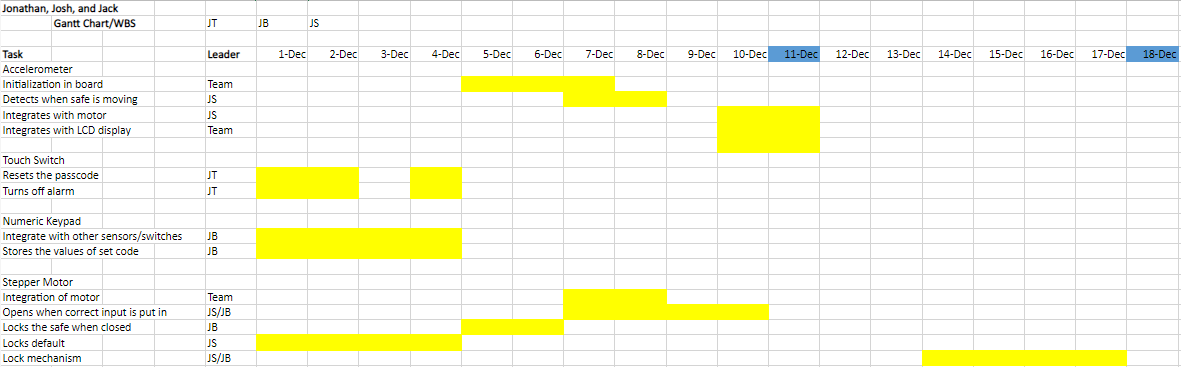
Design

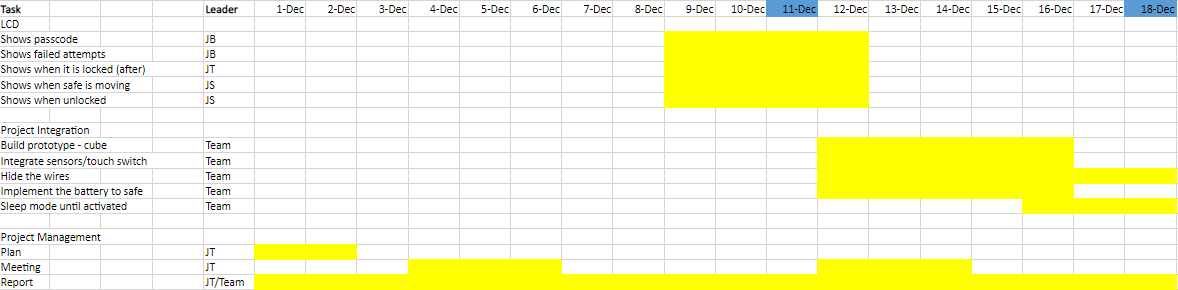
To meet the goal of creating a working safe with the necessary sensors and actuators the following design procedure was used. Once a system level block diagram had been created for the safe and then a Gantt had been made each group member began working on their respective parts of the project. The first parts of the project that were completed were the stepper motor and touch switch. To get the stepper motor to rotate half stepping was used, with port B pins 11-14 on the STM32F407 board being set as outputs and then switched from logic low to logic high in a specific order to rotate the motor either forward or backward. To achieve enough rotation to fully unlock or lock the safe a simple loop was used in the lock and unlock sequence, and the correct amount of rotation needed to lock or unlock the safe was found through simple trial and error by adjusting the length of the loop. The touch switch was set up as a pull-up switch input at port A pin 15, connected to the 3V and ground pins of the STM32F407. When pressed, the button created a high. The program used an external interrupt whenever the button was pressed, a test code would turn on a GPIO output pin of the blue LED to show that it was initialized. The code of the function would reset the code anytime it would be pressed. This would be programmed into the integrated project code, as much of the initialization of the button was to ensure that pressing it would allow for a function in the project code for creating a new four-digit combination. A DIYElectronic 10 pcs 4x3 Matrix keypad was used to input both commands and the code to unlock the safe. The keypad was setup with a non-scan method, where the safe will wait for the user to press a button when it is expecting an input. GPIOC pins 2, 4, 5, 6, 8, 9, and 13 were used. Pins 2, 4, and 13 were connected to the keypad’s C1, C2, and C3 terminals, respectively. Pins 5, 6, 8, and 9 were connected to the keypad’s R1, R2, R3, and R4 terminals respectively. When the safe first started to wait for an input the C pins were set to high outputs and the R pins were set to inputs. As soon as a button was pushed the corresponding R pins would receive a high and enter a function that set C pins to inputs and the R pin related to an output. Then, whatever C pin read a high would allow the board to interpret which key was pressed. It would then send that key’s character to update the necessary variables. The final sensor was the accelerometer which would be run using SPI1 to write and read data from the accelerometer. After initializing SPI1 with the proper specifications for interfacing with the accelerometer, pins 5-7 on port A were set as alternate function pins, with pin 5 being used for the clock, pin 6 was used for MISO, and pin 7 was used for MOSI. Pin 3 on port E was then set up as an output and was later used to read and write data from the accelerometer. Three separate functions were created to read the current data from the accelerometer, with each function reading either the x, y, or z value. Once the data had been stored in an array the functions would return the data via a variable which would be later used for setting the default position of the safe and then comparing the default position to the current position when the safe was locked and triggering an alarm routine if the default position was different from the current position. Our safe functions based on which state it is currently in. There are 3 states: state 0 for locked, state 1 for unlocked, and state 2 for changing the passcode. The code initially starts off in state 0 with the code locked. From state 0, 3 things can happen: the correct code is entered, and the safe is unlocked; the wrong code is entered, and the failed attempts counter is increased; or the alarm goes off due to 4 failed attempts in a row, or the safe being moved. When the alarm goes off, simply enter the correct code to turn off the alarm. Once the safe is unlocked the safe will enter state 1. From state 1 or 2 the alarm cannot be triggered. During state one there is a 5 second period where the touch button can be pushed to change the current passcode. If it is pressed during this time the code will enter state 2. After this 5 second period, press the \* key to lock the safe and return to the locked state. During the state 2, the user will be prompted to enter a new code. Once 4 inputs are received, the user will be prompted to enter # if they would like to save the code they entered. If they would not like to save the code, press any other key to exit without saving. The moment the user chooses to save the code or not, the safe will reenter the unlocked state.

System Level Proposed Block Diagram

Diagram

Description automatically generated





From the costs of the design:

* ARM Cortex M4 Processor (includes MEMS Accelerometer): $19.90
* LCD: $9.95
* Stepper Motor: $4.95
* Darlington Transistor: $0.623
* Touch Switch: $6.99 (in packs of 16, about $.44 each)
* Numeric Keypad Matrix: $13.41

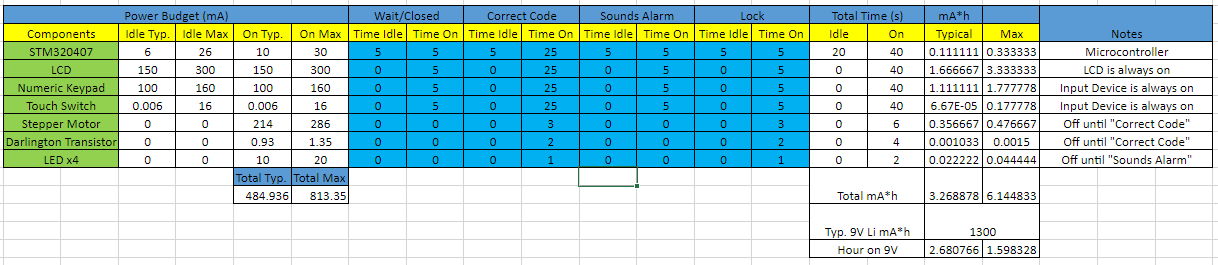
The final cost would be about $49.27.

Design Validation

To validate our design, we tested it until every outcome met our design requirements. Once everything had been wired correctly, the system was powered and turned on with the first test being whether the system was in the state of asking for the user to enter the correct code while the safe was locked.

Power Budget Analysis

The power budget analysis requires the understanding of each component that would affect the overall electrical performance of the system. The components have been determined by calculations, measurements, and datasheets provided in the references. The next figure shows the power budget analysis, along with the conceptual usage of 1300 mA of a 9V Lithium battery.



The stepper motor did not contain typical or maximum current drawings on their datasheets. The calculations were made based on the voltage and the coil resistance of the stepper motor. Another component that did not contain the current drawings on any datasheets was the numeric keypad matrix. The resistance was measured to be zero when nothing is pressed, and 30 ohms when one button is pressed. The keypad would use three volts typically and five volts maximum based on the STM32F407 voltage pins it would be connected to. Both components use Ohm’s laws to find current. Other things to note were the Darlington transistor that controls the stepper motor uses four LEDs, each which used a 20-mA current draw based on common knowledge. This is multiplied by four for the mA\*h columns since there are four LEDs connected to the transistor. The Darlington transistor uses the value based on current input (Ii) in the datasheet.

The power budget separates itself into four states. The first and default state would be the waiting or closed state. Where it waits for an input to be put in, this is either when all attempts are available, or some attempts are still available before the alarm. This state only uses the LCD and input components, the STM32F407 board could be in idle or active state. The correct code would initiate when the right input would be put in, the stepper motor and driver for the motor would turn on but for a moment to act as a locking mechanism, most of the other inputs would remain on for the function of possibly changing the four-digit passcode within the given time. The correct code state would move to the lock state when the safe needs to be closed. The alarm state would happen when the safe has been moved from the accelerometer on the board or the number of attempts have run out. This state would last until the right input was created which moves it to the correct code state. The lock state would happen whenever the safe is open and ready to lock again, which would be like the open state but only closes and does not use any other operation. This would then go to the default state.

Based on the power budget analysis, the typical amount of time the safe would consume 1300 mA of a typical nine-volt Lithium battery spans to around 2.7 hours on typical use and 1.6 hours on maximum use. This is calculated using the mA\*h for the 9V Lithium battery divided by total current of the entire system. Although realistically, the safe would not be in constant use. All the peripherals would operate only when used, which would be fair enough in terms of reaching one of the constraints of lasting for half a year. This safe would only be used to store or take out items only when necessary. The rest of the time would mean the safe would be off.

Conclusion/Discussion

From the prototype design, the functions have served much of their purpose doing what it was supposed to do. It operated properly within the states that it was in. All the input devices and sensors served their purpose within the design, all of which were able to integrate themselves into what the safe was meant to accomplish. The safe could detect when it was being tampered with, the motor could respond to the opening and locking of the safe. The numeric keypad acted as an important input device which could respond and correctly put down the necessary combination, and the touch switch was useful to reset the code when necessary. Further discussion would most likely be necessary for normal usage of the safe in how much in normal conditions the safe could last under battery power only. Otherwise, this could also alternatively be plugged into a wall. The prototype testing received power by the computer. One of the only constraints which was not achieved was the 12x12x12 in dimensions of the safe. However, due to time the methods of creating the prototype were hindered by tasks which proved to take much more time than was thought initially. Some of the input devices and sensors proved to take longer than expected. Some of the tasks went behind schedule than what was originally planned, and thus had been moved into the week where integration of the entire project was meant to take place. However, all the input devices and sensors were able to be integrated into the design and played their proper function as planned.

References

Accelerometer Data Sheet: <https://www.mouser.com/datasheet/2/389/lis3dsh-954955.pdf>

SPI for STM32F407: <https://sites.google.com/site/johnkneenmicrocontrollers/spi/spi_f407>

Darlington Transistor: <https://www.ti.com/product/ULN2003A/part-details/ULN2003AN>

Stepper Motor: <https://www.digikey.com/en/products/detail/adafruit-industries-llc/918/5629415>

Capacitive Touch Switch: <https://hobbycomponents.com/sensors/901-ttp223-capacitive-touch-sensor>

Pricing for LCD: <https://vetco.net/products/16x2-character-lcd-display-d112?gclid=Cj0KCQiAtICdBhCLARIsALUBFcF249TIswiKDE3JEfgQZhvzBzVIbfvcjkkr9Pnq3ci8MrS4JJ9_yTQaAsCYEALw_wcB>

Pricing for Capacitive Touch Switch: <https://www.amazon.com/gp/product/B08G4Z127Z>

Pricing for Numeric Keypad: <https://www.amazon.com/dp/B085HG1YD5?ref_=cm_sw_r_cp_ud_dp_5BPWWDBBGCJCJ89PRFVQ>

Board Pricing: <https://www.digikey.com/en/products/detail/stmicroelectronics/STM32F407G-DISC1/5824404?utm_adgroup=Development%20Boards&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Supplier_STMicroelectronics&utm_term=&utm_content=Development%20Boards&gclid=Cj0KCQiAtICdBhCLARIsALUBFcFG93nszzW1FI1-j6hV3p4TZIj70dubM0qpCHVMoin3brgUeLG2sMAaArIMEALw_wcB>