SAN FRANCISCO STATE UNIVERSITY

SCHOOL OF ENGINEERING

Final Project Report: Home Security System

ENGR 478-04

Fall 2020

By

Amir Modan, #918234621

amodan1@mail.sfsu.edu

Date report submitted: Dec. 20, 2020

Table of Contents

Introduction: When we were first tasked with implementing an Embedded System using the TM4C123 Microcontroller, we wanted to develop a low-cost, yet practical device that solves an issue common in many societies, Home Burglary. Because there are already several professional security systems on the market today with prices starting at over a hundred dollars, we wanted our security system to be extremely cost-effective so that anyone, especially poorer communities in which burglary is most rampant, can utilize our technology. Therefore, we will use several low-cost, yet accurate peripherals that will help detect signs of burglary as well as preventing it, all being controlled by the Tiva Board that is the primary focus of this course. The Tiva Board will also interface with a Raspberry Pi in order to add more advanced features to the Security System.

Hardware Design:

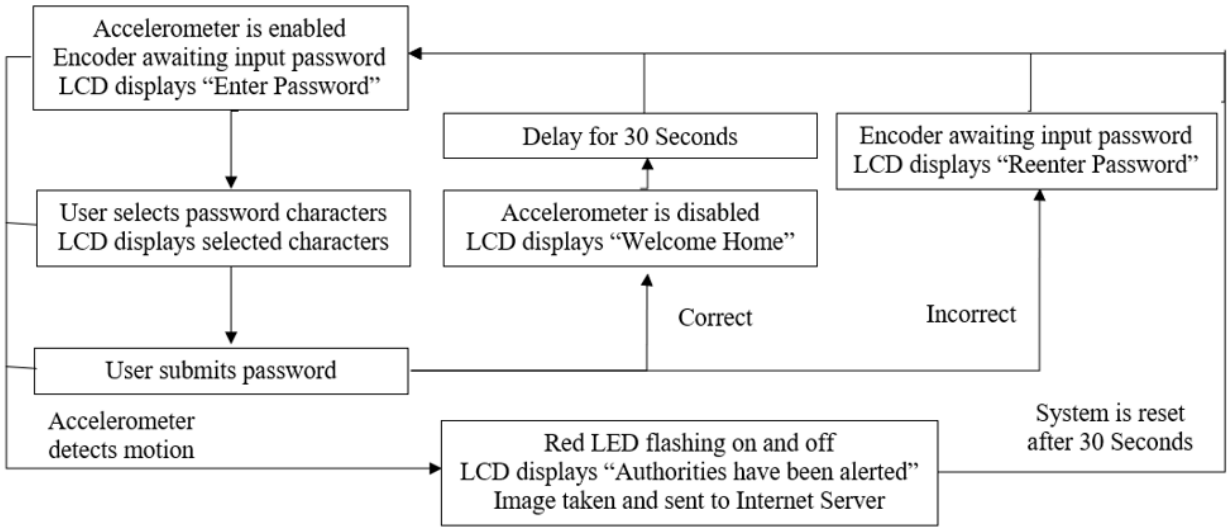

Figure 1 – Design Flow

The main component that will be used in this project is the ADXL345 Accelerometer which is capable of detecting force in the x, y, and z axis. This component, along with the rest of the system, will be mounted directly on a container holding one's valuable belongings. The purpose of this component will be to detect whether someone is attempting to open the chest on which the system will be mounted. We will program the Tiva Board to trigger a response once motion in any axis surpasses its respective threshold. Due to the centrality of the Accelerometer, we will be placing the device closest to the Tiva Board in order to minimize the possibility of noise distortion. Although we will be mounting the security system on a chest for the purposes of demonstration, the 6 axis capabilities of the ADXL345 allow us to mount the system on any apparatus including doors and windows.

Of course, we would want to disable this system whenever the user wishes to access the contents of their chest. This would require the system to be capable of verification of the user as well.

Therefore, we have decided to implement a password-based system that will disable motion detection upon receival of the correct password.

In order to gain access, one would first need to enter a password using a Rotary Encoder to browse through and select characters. We have chosen to utilize an encoder over a potentiometer as the encoder is capable of continuously rotating in either direction, allowing us to cycle through a set of characters with ease. Because absolute position is irrelevant when designing this module, we will opt for the incremental capabilities of our encoder to de/increment the character value each time the shaft is turned. The encoder also contains another built-in button which will be used to submit the selected characters. Two additional buttons will be used to insert/delete individual characters, allowing the user to submit passwords consisting of multiple characters. All components used for password entrance will be placed furthest away from the accelerometer to minimize the chances of the user accidentally making contact with the accelerometer which would likely set off the system.

The characters will be displayed on an LCD1602 for the user to see. Entering the right password will disable the accelerometer for 5 minutes, giving the user time to enter the home without setting off the system. Entering the wrong password would cause the LCD screen to display a warning message and prompt the user to reenter the password. Attempting to open the chest without entering the correct password will cause an on-board LED to flash on and off. A message will also be displayed on the LCD that authorities have been alerted, solely for scaring off the burglar since cellular capabilities will not be added to this device.

However, the system will send a message to the internet via the wireless module built into the Raspberry Pi, allowing a user to monitor the status of their home from anywhere using a web application. A USB Camera will also be connected to the Raspberry Pi, taking a picture whenever a breach has been detected by the Tiva Board. This picture will be accessible through the online server. If one wishes to view the current status of the Security System, the camera will also take a picture on command of the user.

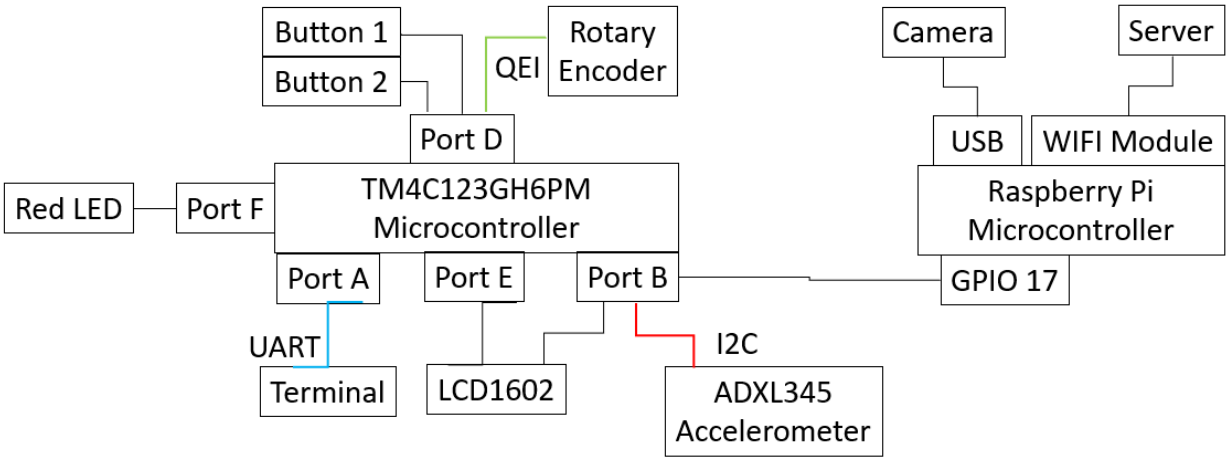

Figure 2 - System Architecture

Software Design:

We have decided to tackle this project module-by-module. This means that we have written and tested separate C programs for each hardware module, then integrated them all into one software system that behaves according to our design plan.

The main component of our system, the ADXL345 Accelerometer, was programmed using the I2C (Inter-Integrated Circuit) serial interface. I2C allows a master to send data to and from a slave using two bi-directional lines, Serial Data Line and Serial Clock Line, each assigned to their own pin on the microcontroller. Three functions have been written that allows a master to read from a slave address, write to a slave address, and conjoin two 8-bit values into one 16-bit value. This conjunction is necessary as motion values for x, y, and z vectors are stored in two separate addresses on the ADXL345. According to the data sheet for the ADXL345, we will have to first configure the ADXL345 by writing the value 0x2D to the slave, which will prompt it to begin sampling data. We can then use our read and conjoin functions as well as a timer interrupt to sample the motion vectors of the ADXL345 at a frequency of 1Hz. If any of the x, y, or z motion vectors are past their respective thresholds (set at the beginning of the program), the security response will ensue. However, the security response can be temporarily disabled if the user successfully validates their identity by entering the correct password. The logic for this module was implemented in the main function rather than in a separate I2C interrupt due to its central role in the security system.

The validation component of the security system is largely dependent upon a rotary encoder which we have programmed using QEI (Quadrature Encoder Interface). Two GPIO pins will be used to detect whether the shaft of the encoder is rotating clockwise or counterclockwise. The TivaWare Library contains functions that determine the position and direction of rotation based on the phase difference between these two pins. If a large change in position is detected, the selected character will increment or decrement once based on the direction of rotation. A small delay was also inserted between each sample to ensure characters aren't skipped when browsing through a selection of characters.

The third pin of the encoder acts as a button which will be used to submit characters when prompted for a password. Two more generic buttons will be used to either insert a single character into the password buffer, or to delete any existing characters from the buffer. All three of these buttons will be programmed using a positive edge-triggered interrupt. When any of these buttons are pressed, an interrupt will occur and will be handled by a single GPIOPort D_Handler function. Here we will first create a small delay to eliminate switch debounce. Then we check which of the three buttons are pressed so we can clear the trigger flag for that pin. Pressing the encoder button will cause a Boolean variable *entered* to become true, causing execution to exit the text editing state and compare the current buffer of entered characters to the actual password. Pressing one of the other two buttons will either enter the current character into the buffer or delete the last character in the buffer if not empty.

An LCD1602 screen will be used to display the current buffer of characters entered by the user. Other messages will also be displayed on the LCD screen to inform the user of an in/correct password, warn the user of a security breach, etc. Seven GPIO Pins are required to program the LCD screen correctly. Four of these pins will be used to supply the LCD screen with data including several possible characters. The other three pins will be used as a register select pin, a read/write pin, and an enable pin. We have written several helper functions that will automatically configure these pins whenever the functions are called. Functions for printing a character or printing a string will be used extensively in our main function.

An output pin indicating the status of the security system will be connected to an input pin on the Raspberry Pi to establish communication to the Raspberry Pi's Web Server. This pin will go high whenever motion in any direction has surpassed the threshold, indicating a breach in security. A resistor will be placed in between these two pins to ensure that a short does not occur in case the Raspberry Pi's GPIO pin has accidentally been configured as an output pin as well.

Unlike the C programs we have written for the TM4C, we will be using Python and HTML to configure the Raspberry Pi. Three separate Python Scripts will be written for the Raspberry Pi. One script will take a picture using the connected USB camera whenever the incoming breach signal goes high. This picture will be saved into the Raspberry Pi's server storage. Because we want the picture to be taken only once, we must use an edge-triggered interrupt to take a picture on every rising edge of the breach signal. Despite using a different language, the theory behind setting up an interrupt is quite the same, by initializing an interrupt and specifying a handler function to execute. However with the use of Python's GPIO libraries for the Raspberry Pi, this process takes only a few lines. In the handler function, we will run a Bash Command telling the Raspberry Pi to take a picture and save it to 'Burglar.jpg'.

The other two scripts will be for uploading text and files to the Raspberry Pi's online server using Python's built-in library 'HTTP Server'. While we are using Python to describe the flow of our program according to GPIO pins, the server's aesthetics will be determined by embedded HTML code within the python scripts. One script will be used for uploading text to the server whereas the other script will be used for uploading images. Additionally, the two scripts will be using two different ports on the Raspberry Pi, the reason being that the two require different encoding; While HTML can be encoded as text, images must be encoded as binary executables, otherwise one of the two will not be encoded and displayed properly. We will design the text port to display whether or not a security breach has been detected depending on the value of the breach pin, easily achieved using an if-else statement. Additionally, the background color of the server site will change accordingly. Links will be displayed underneath the system status prompting the user to either check the most recent burglary image or to test the camera. While the former option simply loads and displays 'Burglar.jpg', the latter option will take a picture in real-time, save it to 'Camera.jpg', and display that picture.

Experimentation: While building the security system, we came across several problems that hindered the performance of the system, stemming from both hardware and software flaws. One such problem arose from the power that we were supplying to the LCD screen. Because the LCD screen was intended to communicate only with the TM4C, we decided it would be appropriate to supply the screen with the supply pin on the TM4C. Little did we know that the LCD1602 required a 5V source to function properly, not a 3.3V source like the one supplied by the TM4C. The lack in voltage resulted in the LCD screen displaying low contrast, making it harder for the user to read.

Figure 3 - LCD1602 supplied with 3.3V

After checking the voltage specifications in the device's datasheet, we deemed it appropriate to opt for a 5V source for the LCD peripheral only. While the TM4C cannot supply this voltage, the Raspberry Pi is capable of supplying both 3.3V and 5V sources, alleviating us from having to acquire a separate power source. Because the pins communicating with the LCD screen are 5V tolerant, we do not have to step down the voltage at each pin either, though this could've been achieved using voltage dividers for each pin. The difference can be seen in Figure 4, with the LCD screen displaying a sharper contrast.

Figure 4 - LCD1602 supplied with 5V

Another problem that we faced was the encoder sending multiple rotation signals as once for one rotation. This is due to the bouncing of input signals as the shaft is turned, similar to how a push-button works. Like a push-button, this problem can be solved by inserting a small delay between each sampling of the encoder to debounce the signals. If one wanted to achieve this using hardware, an RC circuit can also be used to act as a lowpass filter which would stop the higher frequencies being sent due to signal bouncing.

Validation: While observing the behavior of the LCD screen during execution of the system does tell us whether the system is behaving correctly, it is not ideal for debugging as only 32 characters be displayed at a time. In order to facilitate the testing stage of our system, we set up a UART interface solely for monitoring the values of any I/O Pin at any time. Unlike the LCD screen, we can display as many characters as we want on a terminal window to thoroughly monitor any register in the system.

In order to do so, we have to establish UART communication between the terminal and the TM4C using two Port A pins for transmission and receival of characters. "Print" functions from the UART library are used to print the value of whatever variable or Boolean expression we provide it with. Below is an example of how we validate the functionality of the password feature.

Figure 5 - Password Validation

We can see different stimuli being applied to the password buffer, with each character comparison being read-out in real-time. When we had issues regarding false passes/fails of the password, this feature helped us immensely in pinpointing the bug in the program. For instance, we initially did not clear the password buffer each time the password was checked. This meant that any characters we entered into the buffer would remain there even after it is checked, allowing for a user to continuously deactivate the system after entering the correct password only once. Fortunately, we detected this behavior by analyzing the comparisons being made and addressed the problem by setting all characters in the password buffer to ' ' before prompting for input.

Additionally, we monitor the breach pin going from the TM4C to the Raspberry Pi to ensure that it is behaving correctly. We want the security system to send a response at any stage of the design flow (except when deactivated) regardless of whether we are entering characters into the buffer. This behavior is seen above, with the security breach occurring even while we are entering characters into the password buffer. Furthermore, the I/O register indicating a breach in security goes high each time this occurs. If any problem were to occur regarding the correct display of system status, we could monitor the status of the register to determine whether a bug was present in the TM4C's programming, or the Raspberry Pi's.

Results: After validating the Security Response at each stage of the design flow, we are able to say with confidence that our Security System performs the expected tasks with a high degree of accuracy. When the system is at rest, we observe the behavior shown in Figure 6.

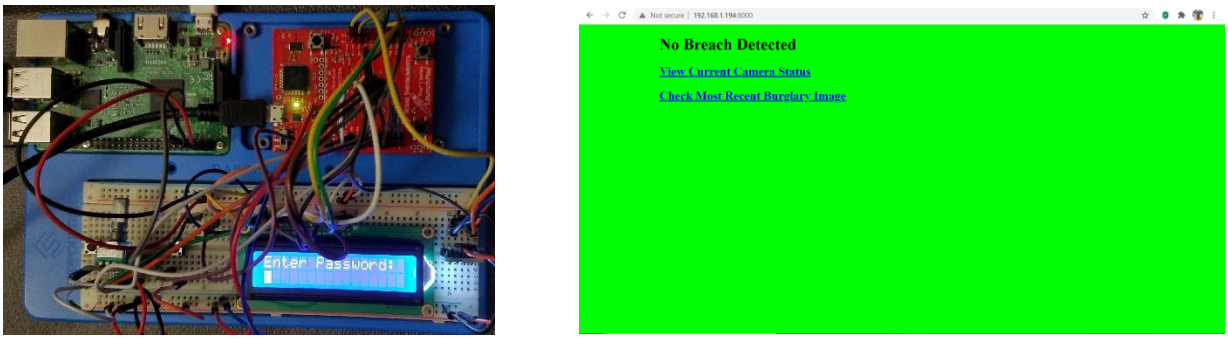

Figure 6 - LCD Display (Left) and Website (Right) when Security Response is inactive.

Whereas a system detecting a large enough motion vector would exhibit results shown in Figure 7.

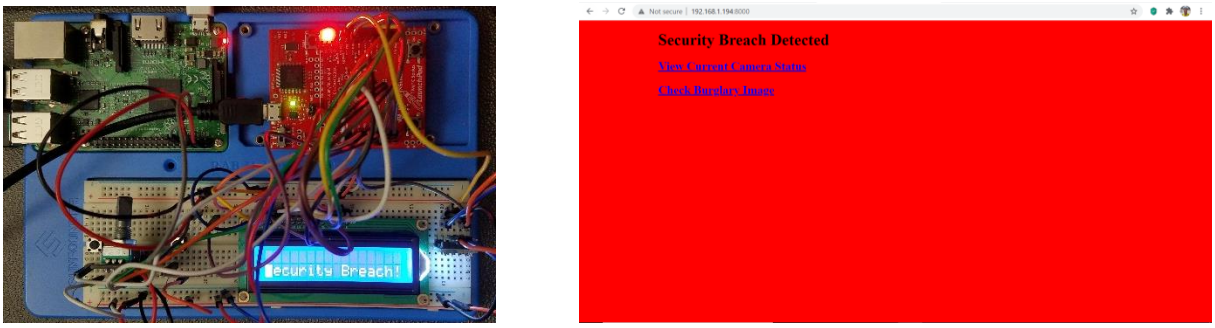

Figure 7 - LCD Display (Left) and Website (Right) when Security Response is active.

Clicking either link also displays the desired image with a timestamp as shown in Figure 8.

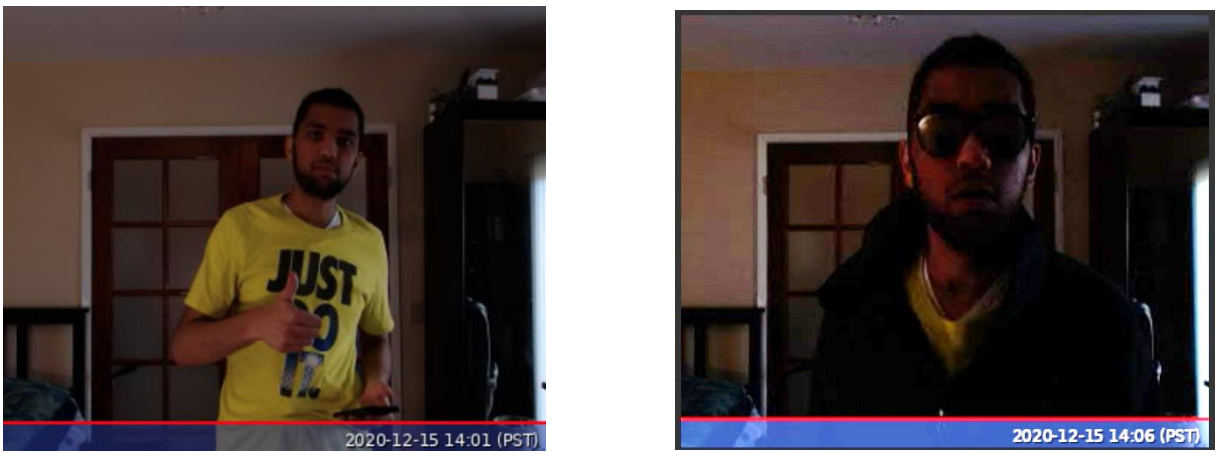

Figure 8 - Camera Check (Left) and Burglar Evidence (Right)

Discussion: While our project is currently an excellent Security System for its manufacturing price, there are still many features that we intend to add. First and foremost, our web server needs more security protocols as the server and the camera is accessible to anyone who knows the IP Address of the Raspberry Pi, creating a security risk in the very system we created to increase security. For this, we intend to shift our server implementation to the much more versatile LAMP (Linux, Apache, MySQL, PHP) server that will allow us to place security measures on the website itself. Using LAMP will also allow us to create a database in which we can store every image taken rather than keeping only the most recent.

In order to better facilitate communication between the two microcontrollers, we intend to set up an I2C interface which can send multiple bits of data on a single data bus rather than the basic digital I/O interface we are currently using. With this data bus, we plan to add features to remotely disable the system, change motion sensitivity, and change passwords, all from our soon-to-be secure server.

Finally, we intend to add more peripherals to the security system itself, improving the local response so the home's inhabitants can be alerted if present. We can do this by adding a loudspeaker to the system and implementing a Digital-to-Analog conversion which would emit a sharp sound.

Conclusion: We have now learnt the basics behind Designing with Microprocessors by building our very own Embedded System using both the TM4C and Raspberry Pi Microcontrollers. We have gained experience using several different peripherals covering a wide range of functions, from motion detection to character display. We have also familiarized ourselves with different modes of data communication including Digital GPIO, UART, I2C, and USB. Finally, we have applied different techniques commonly used in Embedded Systems such as Edge-Triggered Interrupts and Periodic Interrupts. Judging from the low cost of manufacturing our project, we have proven that an effective home security solution does not have to be more than a hundred dollars. Our system has the potential to bridge the gap between the underserved communities and a safe and secure lifestyle.

References:

1. ADXL345 Data Sheet
2. LCD1602 Data Sheet
3. TM4C Data Sheet
4. TivaWare Peripheral Driver Library
5. Cheung, Henry; How to control Raspberry Pi GPIO via http web server