6-2011

A Biometric Fingerprint Scanner With a Locking Mechanism

Nathaniel Clark
Union College - Schenectady, NY

A Biometric Fingerprint Scanner With a Locking Mechanism

By

Nathaniel Elbert Clark

****

Submitted in partial fulfillment of the requirements for

Honors in the Department of Electrical Engineering

UNION COLLEGE

June 2011

ABSTRACT

CLARK, NATHANIEL A Biometric Fingerprint Scanner With a Locking Mechanism

Department of Electrical Engineering, June 2011

ADVISOR: Shane Cotter

The purpose of this capstone project was to design, construct, and test a biometric fingerprint locking system to control access to a set of lockers. This system offers a convenient alternative to traditional locks and makes lockers easier to use. Traditional locks require some type of prior knowledge, like remembering the numbers and how to enter them for a combination lock. This can make it difficult and annoying to try and use a simple locker.

Three main considerations were kept in mind during the design process to make this system more convenient than traditional locks: user friendliness, accuracy of identification, and false positive rate. The most important aspect of this project was to make a user friendly system, without challenges and questions. It was also vital to have a high identification rate while the false positive rate should be kept to a minimum, as people should not be “accidentally” granted access to other’s lockers.

To solve this problem, the project was based around a fingerprint scanner which was controlled by a microcontroller. The microcontroller was the brain of the system. It sends instructions to the fingerprint scanner, interprets returned data, and controls a LCD display so that the system was more user friendly. This report will discuss the background information needed to comprehend a fingerprint identification system, a detailed explanation of the design requirements, the design alternatives along with why components were selected, the preliminary proposed design, and the design of the final prototype.

Table of Contents

Table of Figures and Tables

1. Introduction

Traditional locks, while commonplace in today's society, are nonetheless inferior to biometric locks. To operate a traditional lock, a user must possess prior knowledge which allows them entry. For example, the standard lock and key requires a matching key, a computer requires the proper password for a specific log on ID, and a combination lock requires a series of numbers entered in the correct order and fashion. Have you ever tried to open a combination lock and forgotten the code? Have you ever forgotten where you placed your keys, or even permanently lost them? An alternative solution to the traditional lock is a biometric lock. Biometric locks use a measurement of behavioral or physical characteristics to identify people contrast to traditional locks which use keys or passwords. Popular biometric systems include fingerprint identification, facial identification, hand geometry, and speech analysis.

This project will implement a biometric fingerprint identification system for access control to lockers. This solves the problem of forgetting your combination to your personal locker and also not knowing the fashion in which the numbers must be entered. This system will essentially employ the user's fingerprint as the key to their locker. Users will enroll their fingerprint in the system via a fingerprint scanner and be assigned an empty locker. After storing their personal belongings and closing the locker, it will automatically lock itself shut. When they want to open their locker and access their belongings again, they will simply scan their fingerprint, wait for the identification process, and their locker will open. The only information the users will be required to possess is their fingerprint. Thus, the problem of traditional locks will not be present in this user friendly system.

This report discusses the method of implementing such a biometric fingerprint identification system for controlling lockers. First, the history, general topic, previous and similar

works, and the important issues and their effects on society will be discussed. Then, the design requirements of this system will be introduced including the specifications and requirements for this project. The section design alternatives covers the options and justification for the selection of the components that will be utilized in this project. Lastly, the proposed design is described. This includes a detailed design of how the project was completed as well as the corresponding timeline.

2. Background

2.1 History

Fingerprint identification has been around for a couple of centuries. For the majority of this time, fingerprints were used to help identify criminals. Dr. Henry Faulds and William Herschel were some of the first people to use fingerprints as a method of identification. Edward Henry invented a system in 1897 which classified fingerprints into categories based on their patterns and assigned numerical values to these pattern types. In the early 1900's the United States military began using the Henry Classification system to indentify service members. More recently, fingerprints have been used quite often for security access, criminal identification, deceased identification [1].

2.2 Fingerprints as Biometrics

Fingerprints have many properties that make fingerprint identification a good biometric. Because fingerprints are essentially universal (i.e., most people have fingerprints) this system can be used by almost anybody. Another advantage of fingerprints is their uniqueness. Not only does everyone have a fingerprint, but all fingerprints are distinct. Fingerprints even vary from finger to finger on the same person. In general, fingerprints can easily be recorded and the identification process is quick, efficient, and most importantly accurate. Ethically speaking,

fingerprint use has become commonly accepted by society as a method for identification and there are no expected problems for this system especially since it is being designed for personal usage. In general, fingerprints provide an accurate, easy, and accepted method for personal identification.

2.3 Fingerprint Identification

A biometric fingerprint lock relies upon a modified version of an automatic fingerprinting identification system (AFIS). An AFIS system, or any system similar to AFIS, classifies fingerprints into categories based on their geometric properties. Separating fingerprints into different categories is useful as it can speed up the time of fingerprint matching. If you could split all fingerprints up into a number of categories, you could search a fraction of them (just the ones in the same category) rather than all of the known fingerprints. There have been studies which produced algorithms that divide fingerprints into 5 or 6 categories that result in 90 percent accuracy. However, it is considered hard to divide fingerprints into more categories and retain a higher percentage of accuracy. In general fingerprints can be classified into 6 different types: arch, tented arch, left loop, right loop, whorl, and twin-loop [2]. A picture of these different classes of fingerprints can be seen in figure 1 below:

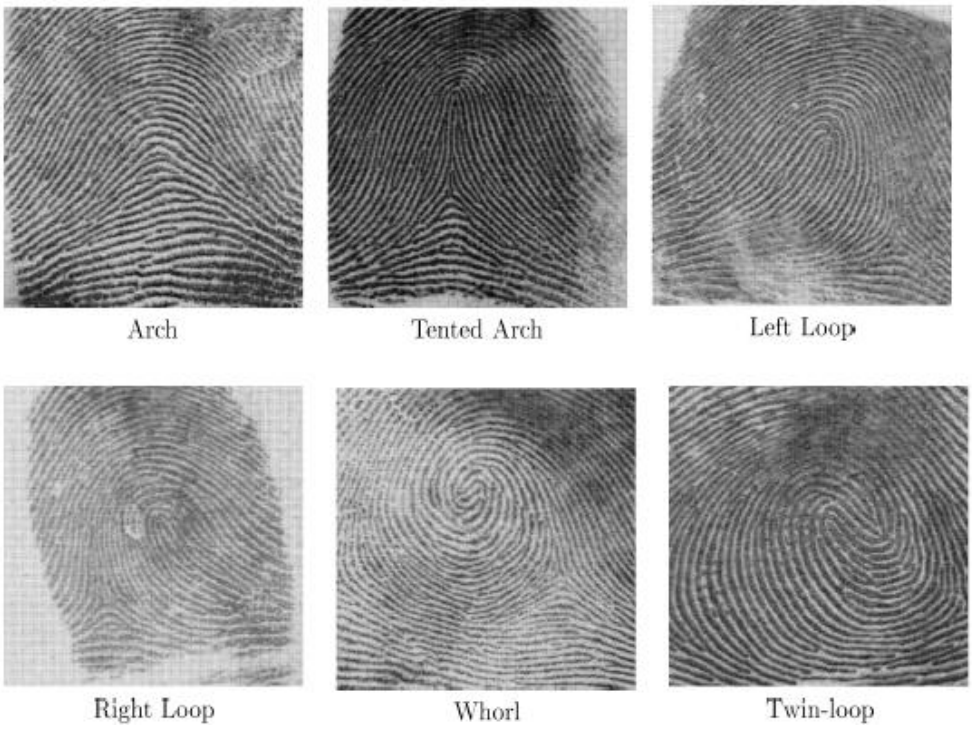

Figure 1: The Six Different Types of Fingerprints [3]

AFIS systems use many techniques to analyze biometric fingerprint data. Most programs and computers can not operate directly with a raw fingerprint; thus features must be extracted to summarize the print information. The original grayscale fingerprint is reduced to lines made of black dots on a white background, and this is termed "thinning". The physical ridges on the fingertip form various patterns, or minutiae, which can be analyzed. The two main fingerprint minutiae are ridge endings and bifurcations, seen in figure 2 below:

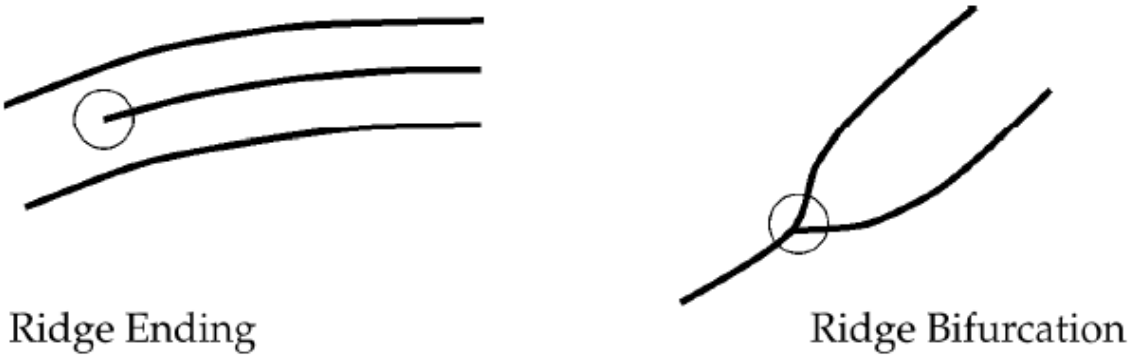

Figure 2: Ridge Ending and Ridge Bifurcation [3]

A ridge ending is where the ridge stops and a ridge bifurcation is where the ridge diverges, or splits. Thus you can use thinning to reduce an image to just an array of points that are located at these minutiae. It is known that fingerprints are distinct to each person and even identical twins have different fingerprints [4]! Similarly, the location and pattern of minutiae are unique for each fingerprint. Using this process, contrast, brightness, and noise can largely be eliminated. The process of acquiring a fingerprint, thinning it, and detecting and plotting the minutiae can be seen in figure 3 below:

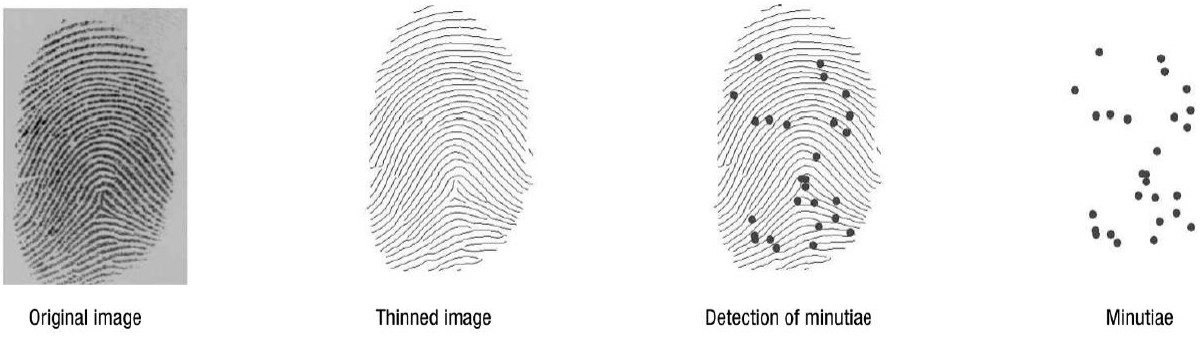

Figure 3: Plotting the Minutia of a Fingerprint [3]

In a typical system, the minutiae data is approximately 1000 times smaller than the data from the original print. It should be noted that multiple minutiae images from the same finger will not

always be the same. Of course, general fingerprint images of the same finger are not always the same either, so this is somewhat expected. The next step is for the AFIS system to compare the minutiae to a database in search of a match. Depending on the type of system, it will then grant or deny access, or identify an unknown individual.

2.4 Related Work

Currently, there are some existing biometric fingerprint door locks that start around $330 [5]. This is economically unsuitable for personal usage. Because this system will have the capabilities to be used by a small group of people this will make the economic costs more bearable. It also makes this system more desirable as the costs are greatly reduced in some cases. There are also similar designs on the market for a set of lockers controlled by fingerprint access, but most were for a larger scale. That is, the number of lockers being used in the system was upwards of 50, where in this systems case the number of lockers would be less than 10 as this will be designed for a small personal scale.

3. Design Issues

3.1 Economic

The goal of this project was to have the overall cost of the system below $250. If this project was marketed, it could be sold for profit as the production costs remains under the current existing biometric fingerprint door locks that are on the market. Ideally, the cost could be further reduced by customizing parts, enlarging the economic benefit of this project. The price range for this project places it in a reasonable economic range, not over the top of the current prices or dramatically below them.

3.2 Environmental

This project has little to no environmental impact, as it only requires two AC wall outlets to operate. Because the system draws minimal power, negative environmental impact was mitigated.

3.3 Manufacturability

This system is mainly dependent upon the fingerprint scanner. The Nitgen fingerprint scanner used in this project is available on many suppliers' websites and in the worst case it could potentially be replaced by another scanner. Some of the other components, like the MOSFET device and Arduino shield, may be difficult to find but are replaceable by other similar parts. The remaining items of this project are all relatively common and easy to obtain. It would be difficult to replace the system's parts with custom made ones, as it would be extremely challenging to construct and interface them, though this could potentially reduce the cost.

3.4 Ethical

Fingerprint data, like most other biometric data, was an ethical concern for this project. Because a fingerprint scanner is used, it is important for the users of this system to understand that their data will be kept to the fingerprint scanner's analysis chip and it will not be uploaded to the computer. Before testing was performed on this project, it was approved by the Human Subjects Review Committee at Union College [6]. It was important to note that the fingerprints from the testing were deleted upon its completion. Users in the system who would like to have their fingerprint deleted can easily do so with the help of the system's administrator.

3.5 Health and Safety

There were no health and safety concerns with this project. The final prototype was placed within a plastic enclosure so there were no exposed wires that could potentially harm the users. Assuming that the users are careful with plugging in the power cables and are conscientious around the lock, then the health and safety of the users is assured.

4. Design Requirements

4.1 Cost

Ideally this system's cost will be less than existing systems that are already on the market. Most systems already designed and implemented are not for personal use and are more applicable to a wide spread application. That being said, the fingerprint scanners can be expensive but are also the most essential component of the system. Thus, the goal of this project is to have the overall cost of the system be less than $250.00, and in this way the system is cheaper than the current models that are available. This makes the product more desirable.

4.2 Performance

There are two major considerations for the performance of the system: the accuracy of identification and the false positive rate. The more accurate the system is the more likely the false positive rate will increase. False positives occur in two situations. The less detrimental one is where a user is trying to access their belongings but is identified as a different user and thus the wrong locker is opened. The worse case is someone who is trying to hack the system and is identified as a user and given access to a locker. For these reasons the goals of this project are to limit the false positive rate as much as possible and have an understanding that users may be denied access to their belongings temporarily and have to rescan their fingerprint. That being

said, this small nuisance should be accepted by the users as this would ensure the safety of their belongings. For the sake of numerical values, we strived for an acceptance rate of at least 90 percent along with a false positive rate of less than 1 percent.

4.3 Functionality

In order for this system to be useful, it must be practical for its users. Here we must consider the speed of the system. To be functional, there should not be a long delay in between scanning a fingerprint into the system and opening their locker. The system should be as quick as possible so that the users do not become frustrated with the speed of the system. While this system is not designed for speed but more convenience and ease of use, the users should not feel that they are wasting their time. The users should believe that it is worthwhile to use this system as we do not want them returning to the traditional lock, as one of the motivations behind this project is to replace those locks!

4.4 User Friendliness

A major design consideration in this project was the level of user friendliness for the final prototype. To make the system user friendly, this system was designed to interact naturally with its users. Thus, a LCD display was used so that the user receives feedback from the system. For example, if your fingerprint was correctly scanned it prints that on the screen. This way, the users do not sit there and wonder what the system is doing. It was also important to develop a user's manual. This gives the general instructions to the users as well as how to troubleshoot basic problems.

4.5 Convenience and Acceptance

To make this system more accepted by the users, the end product has a clean and professional appearance. The professional appeal not only makes it easier to use, but gives confidence to the users that the system works well. As the system only requires two AC wall outlets, has a built in LCD screen, and does not take long to operate, it is relatively convenient and easy to use.

4.6 Safety

There should be no safety concerns involving the users of this system. Fingerprint identification is widely considered safe and there are no known health risks involved with the scanners. One consideration for the final design is the neatness. There are no protruding wires from components or sharp edges, as those could potentially harm the user.

5. Design Alternatives

5.1 Biometrics

Fingerprint data were used in this project, but other types of biometric systems were considered. There are many other types of biometrics available today like face, voice, hand geometry, retinal, etc. While they all are appealing, fingerprints usage is already accepted within today's society as a recognition process, while other biometrics can be controversial. Thus, people would not be opposed to using this system. Other strong points of a fingerprint identification system are that fingerprints are universal, unique, the analysis is quick and well established to be valid, and scanners are relatively inexpensive compared to other biometrics. For these reasons, this project implemented a fingerprint identification system.

5.2 Fingerprint Scanner

Two different fingerprint scanners were considered for this project, the Nitgen RS232 serial fingerprint scanner and the ARA-ME-01 fingerprint slide. Both devices may be seen in figures 4 and 5 below:

Figure 4: Nitgen RS232 fingerprint scanner [7]

Figure 5: ARA-ME-01 fingerprint slide [7]

The Nitgen had many pros, and the major advantage of this scanner was onboard verification capabilities. More specifically, it can capture a fingerprint in 0.2 seconds and verify in 1 second. The Nitgen communicates with external devices through serial communication at 9600 bits per second. The Arduino would talk to the device in this manner, sending instructions to the Nitgen

and receiving the results. Lastly, this scanner operates at 3.3 V at 200 mA which works nicely with the Arduino as it has an output pin for 3.3 V.

The ARA-ME-01 fingerprint slider also communicates via serial communication at 9600 bits per second. It is powered with 5 V at 60 mA, similar to the Arduino. It has built in memory that is capable of storing 120 fingerprints. However, unlike the Nitgen, the ARA-ME does not have onboard identification. Therefore, this code would have to be written, tested, and implemented separately. This would make it difficult to complete the entire system in the given time period.

In the end, the Nitgen was selected because of its powerful capabilities. Also, it had better documentation as well as a large user population who have completed similar projects. The supplier's website had detailed documentation on the Nitgen and also how to communicate with the device. In contrast, the specification sheet was the only available document for the ARA-ME as the communication documentation was not currently available. Ultimately, due to many advantages that the Nitgen RS232 serial fingerprint scanner had over the ARA-ME-01 fingerprint slide, the Nitgen was chosen as the fingerprint scanner for this project.

5.3 Microcontroller

Two microcontrollers were considered in the initial stages of this project: the Arduino Duemilanove and the 8051. The Arduino is coded using the Arduino development environment which is based on the programming language C. There is a large pool of available example codes and in general it is easy to learn this language for programming beginners. Since the Arduino acts as a means of communicating between the fingerprint scanner and the user, it is not necessary for it to have the computing power. For these reasons, the Arduino was preferred to the

8051 and was selected as the microcontroller for this system. The Arduino Duemilanove is shown in figure 6:

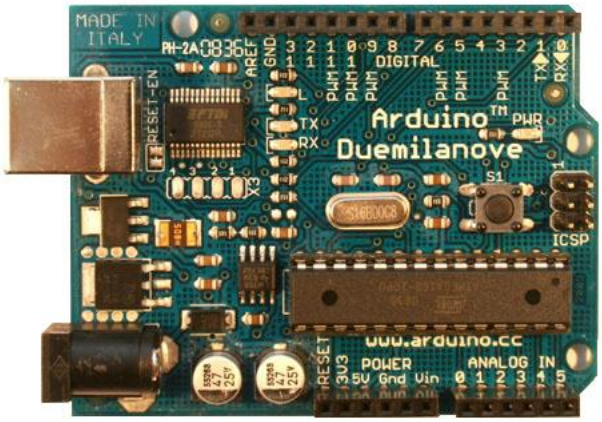

Figure 6: The Arduino Duemilanove microcontroller [8]

5.4 Power

A major concern for this project was how the Arduino and system would be powered. The Arduino has an onboard regulator which works best between 7-12 V DC and can convert that to the 5 V operating voltage. A few different designs to power the system were investigated including batteries and an AC to DC converter. Batteries would be inconvenient, unreliable, and could become expensive since the system will need to be debugged and tested so lots of batteries would be used. On the other hand, a wall adapter power supply that outputs 9 V was the ideal choice, seen in figure 8 below:

Figure 7: Wall Adapter [7]

It plugs directly into the DC jack on the Arduino and constantly supplies power to the system (assuming no power outages). Therefore, this wall adapter was purchased and incorporated into the system. The battery idea will be kept in mind as a possible back up power supply for power outages.

5.5 Minor Components

One of the main goals of this project was to design a user friendly system and a LCD display was purchased to accomplish this. The LCD has two lines with 16 characters each and operates at 5 V which can be powered off the Arduino, as shown in figure 7:

Figure 8: LCD display [7]

The LCD is controlled by serial commands sent from the Arduino. It uses the same commands as the HD44780 parallel interface chip which made it extremely easy to get the LCD up and

running properly. No other LCD displays were considered as this component is inexpensive and easy to use.

6. Preliminary Proposed Design

This section discusses the progress made with the project for the fall semester term as well as proposed work for the following winter semester.

6.1 Work Performed During the Fall Semester

This semester was extremely productive in investigating the initial stages of this system. It started by downloading, installing, and becoming familiar with the Arduino environment which is the programming language used for all Arduino microcontrollers. We began by following examples found online and in the Arduino library to do the most basic things, such as make external LEDs blink. Knowing that serial communication will be an essential aspect of this project, we used the Arduino to practice serial communication with the built in serial monitor. The next step was to communicate between two Arduinos which was completed after multiple failed attempts. One Arduino was used to send signals to the other and when the proper letter was received the other Arduino lit up an LED. This is shown below in figure 9:

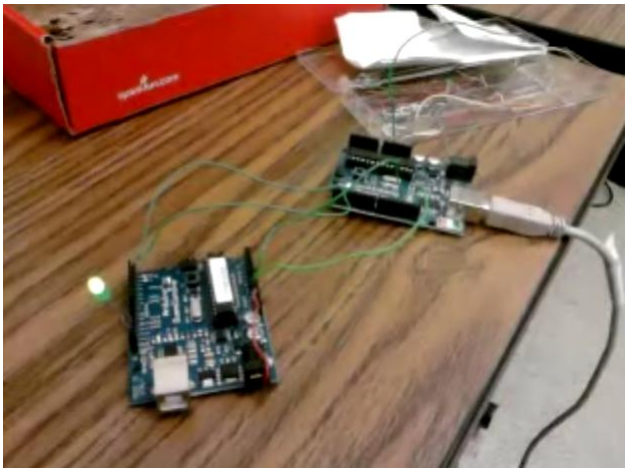

Figure 9: Serial communication between two Arduinos

A display was also ordered during this semester. The proper wires were soldered onto the LCD display so that more easily connected to the Arduino. In a relatively short amount of time the LCD was up and running and it was relatively easy to use. Some simple functions were written that could be implemented later such as printing "The locker is being opened". The LCD has 2 lines of 16 characters each. Using the built in "software serial" library of the Arduino development environment, I was able to send commands to the display.

There was also major progress made with the main identification component of the system, the Nitgen RS232 fingerprint scanner. This was ordered before the semester started but before this component could be tested a MLX to MLX cable had to be ordered. This cable is responsible for the device's power, ground, and serial communication. Once the MLX to MLX cable arrived it was cut in half and the wires had to be stripped so they could be inserted into a breadboard or Arduino. The wires were extremely small so it took lots of care and precision to strip them. Due to their size, the wires had to be soldered to a "bridge" which could be inserted into a breadboard for testing.

The first method of testing consisted of operating the Nitgen without the serial communication ports. This was done by using pins 6,7, and 8 on the connector. These should normally be set to high, 3.3V, but temporarily setting them low performs specific operations. For instance, setting pin 7 low for greater than 30 ms will allow you to enroll a fingerprint, pin 9 performs the verification, and pin 8 can be used for deleting the fingerprints. This testing can be seen in figure 10 below:

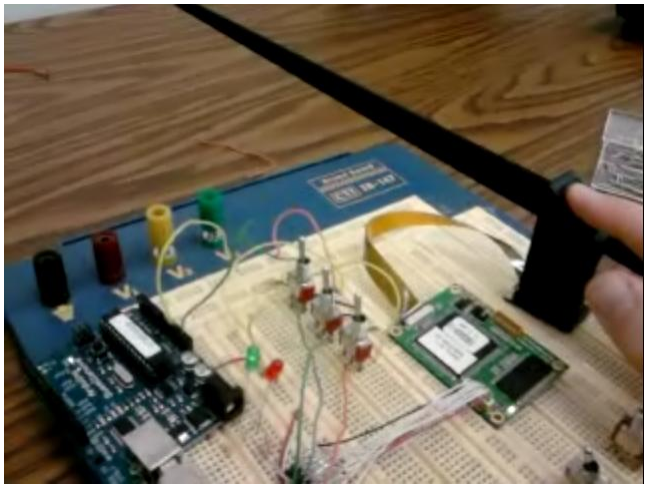

Figure 10: Testing the FIM without serial communication

After designing and testing a simple toggle switch circuit, it was connected directly to the fingerprint scanner. These toggle switches allow users to control the fingerprint scanner without serial communication. Monitoring pins 4 and 5 allow you to see if the operations were performed correctly, as they are temporarily written high to denote a success or failure. Thus, between the toggle switches and LED connected to pins 4 and 5 we verified that fingerprint scanner worked.

The last section of work done on this project during the fall semester was talking to the Nitgen RS232 fingerprint scanner using serial communication. The idea was to first verify that serial communication has been established using a PC. Because of the difficulties using HyperTerminal, it was tested using a Linux operating system. The simplicity allows for whatever

comes back from the scanner to be displayed on the screen, in other words the PC did not try to interpret any of the data. Using C we composed messages that were sent to the scanner. We first set up the communication channel. Once we received the verification that indeed the channel was set up properly we moved on and had the scanner send us its version information. We were also successful in this, and the next step was to do the same using the Arduino!

7. Final Design and Implementation

This section discusses the major successes that were made in the winter semester that led to the development of a final working prototype.

7.1 MAX232 Circuit

Before being able to attempt communication between the Arduino Duemilanove and the Nitgen fingerprint scanner, the proper serial communication channel had to be constructed. The channel is necessary because the Arduino communicates on transistor-transistor logic (TTL) levels which vary between 0 and 5 V. On the other hand, the Nitgen uses standard RS232 voltages levels which vary from +3 to +25 V and -3 to -25 V [10]. The MAX232 is an integrated circuit chip created by Maxim that is capable of converting between the TTL and RS232 levels [11]. The chip was connected using 22μF tantalum capacitors as seen in figure 11:

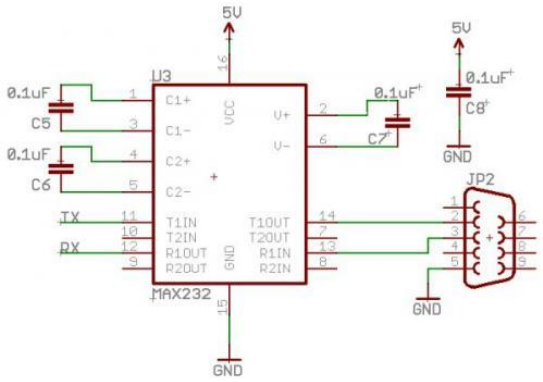

Figure 11: Pin diagram used for the MAX232 chip [9]

22μF tantalum capacitors were used because they have low leakage currents and were recommended by Professor James Hedrick at Union College. Once this circuit was built, it was tested using the Arduino, a female DB9 serial connector, and the program Hyper Terminal on the PC. The Arduino was set to continuously send print statements of "Hello Nate" to the PC. When these print statements were properly displayed in Hyper Terminal the MAX232 setup was verified.

7.2 Fingerprint Scanner

Once the MAX232 setup was verified, the first step of communicating with the Nitgen fingerprint scanner was to form a connection. After searching the forums, I was able to receive some example code which helped me understand how I could implement the desired function in the Arduino development environment. For the scanner to perform an operation, it must receive a series of bytes that have the following structure:

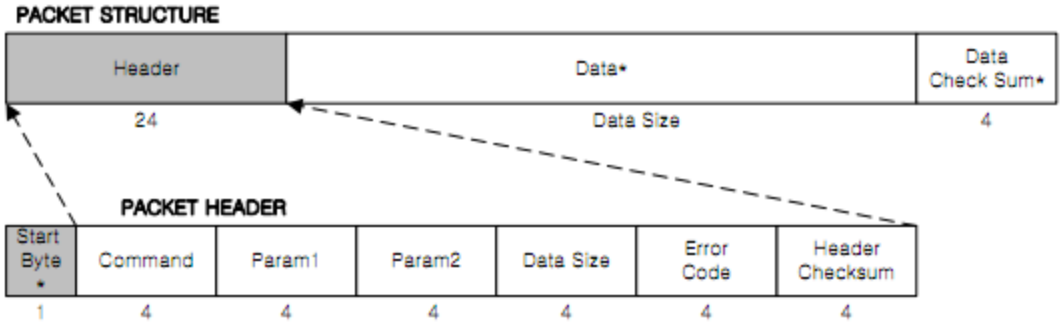

Figure 12: Structure of byes that must be sent to the Nitgen scanner [12]

All commands begin with the start byte of 0x7E, and this alerts the scanner that the incoming data is a command. The packet header varied for each of the commands based on what was being performed. After the data was sent to the scanner, the Arduino is coded to wait for a response which is also signaled by the start byte of 0x7E. Once the Arduino confirms that all of the bytes are waiting in the serial buffer, it then saves them in an array of the proper length. The code to read in the received serial data is shown below in figure 13:

```
if (Serial.available() >= 24) {
  for(i=0;i<24;i++){
    rx[i]=Serial.read();
  }
}
```

Figure 13: Arduino code to read in the serial data received from the scanner

Once the data has been saved in the array "rx" it can then be examined to determine what the scanner had performed. The 8th byte of the returned data is the most important portion as it tells you if the command was successfully performed, failed, or one of the many possible has occurred when using the scanner. To determine the proper course of action, the 8th byte is examined using switch statements. Depending on the function written, various cases report back to the user via the LCD screen. In the identification process, the cases that may occur are that their locker will be opened, the lock will remain closed, there was a failure in the process, and the fingerprint was not placed down in time. After performing all of the proper operations the

code always returns back to the welcome screen where users are instructed to select an option with one of the toggle switches. Many commands were written in the Arduino development environment including: requesting a connection, entering mater mode without a master fingerprint, leaving master mode, registering a master fingerprint, entering master mode with a master fingerprint, enrolling a new user, identifying a user, deleting a user, and checking the quality of the fingerprint image template. Most of the functions had a similar form where the packet header is altered to instruct the scanner to perform different operations. The code used for the final prototype may be seen in Appendix 1 "Arduino Final Prototype Code".

7.3 Toggle Switch

Besides coding the Arduino to talk and listen to the fingerprint scanner, it was also coded to understand when a user wanted to use the system. For that, two toggle switches were used. The switches were normally set to ground with a pull down resistor. The Arduino, with its repeating "loop" structure, is constantly reading the voltage on two of its pins. One pin corresponded to identification and the other to enrolling as new user. If the Arduino reads in that one of these pins is high, it waits 1.25 seconds and reads it again to see if is still high. This ensures that the user wants to perform the given function and that the switch was not operated by accident. Then, the Arduino calls the proper function based on the toggle switch. The code for instructing the Arduino to call the identification code may be seen in figure 14 below:

```
int A = digitalRead(identifyPin);
 if (A == 1){
   delay(1250);
   int C = digitalRead(identifyPin);
   if (C == 1){
      identify();
   }
 }
```

Figure 14: Arduino Code for the toggle switches

A similar portion of code was written for calling the enroll function. The complete code may be seen in Appendix 1 “Arduino Final Prototype Code” in the “loop” section of the code.

7.4 Arduino 4 Route MOSFET Button IRF540

Before integrating the MOSFET device into the system, it was first tested using a simple circuit with LEDs to verify its function and also the wiring schematic. The schematic used to wire the circuits may be seen in figure 15 below:

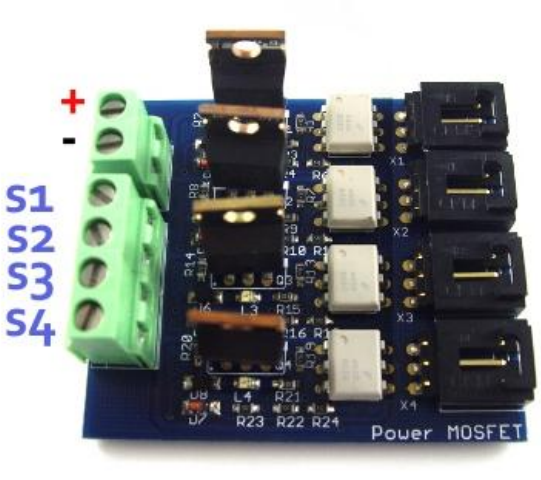

Figure 15: Wiring diagrams for the MOSFET device [13]

The power for the LEDs that were being operated using this device was connected to the positive and negative terminals. Then, the positive side of the LED was connected to the MOSFETs positive pin and the negative side of the LED was connected to the desired switch, illustrated as S1, S2, S3, and S4. For testing purposes, LEDs were connected to all of the switches and a basic Arduino program was written to turn on various LEDs, testing the capacity of all four switches.

The testing code may be seen in Appendix 2 “Arduino MOSFET Testing Code”. The MOSFET device connected to an Arduino Sensor Shield by a Common Sensor Cable. The Arduino Sensor Shield is an extension of the Arduino and is seen in figure 16 below:

Figure 16 Arduino Sensor Shield [13]

This shield allowed the Arduino to write pins high and low which switched the component’s power on and off, respectively. In this way, the lock was controlled easily.

7.5 Lock

The Lee 12 V DC electric door strike lock was a key part of this project’s design. The lock comes with two wire leads allowing it to be powered. The power leads are reversible with the positive and negative voltages of the power supply. The lock is shown in figure 17 below:

Figure 17: Lee 12 V DC electric door strike [14]

This lock remains closed when no voltage is applied. Through testing, I found that anytime that power is applied to the lock it is open. For instance, if you apply power for 1 second, the lock can be opened for 1 second. If you apply power for 1 hour, then the lock may be opened as many times as you want during that time period. A 12V wall adapter was purchased to power the lock. Instead of using a converter, the wall adapter's wire was cut revealing the positive and negative leads of the power supply. These wires were stripped down and then connected directly to the MOSFET device, saving component space and also reducing the production cost. The lock makes a faint "click" when it is opened. The lock was also tested with the MOSFET. This allowed for easy and quick control of opening and closing the lock by simply writing one pin high on the Arduino for the duration you would like the lock to be opened for. Be sure to connect the positive and negative power supply leads correctly to the MOSFET or else the lock will remain open as power is constantly applied to it.

7.6 Enclosure

To make a neat and professional final prototype, all components were placed in a plastic project enclosure box. Due to the size and amount of components of the project, a 6"x4"x2" ABS plastic enclosure was chosen, seen below in figure 18:

Figure 18: Project enclosure used for the final prototype [15]

After obtaining the enclosure, I used a ruler to lay out all of the pieces that will protrude from the box like the fingerprint scanner, two toggle switches, LCD screen, and power cords. Once I had laid out the locations where I wanted all of the components, I carefully measured out a nice design and the Machine Shop at Union College cut the box to fit all of the components. With help of Gene Davidson, I was able to secure all of the components within the enclosure. The Arduino Duemilanove and the MOSFET device were screwed down while the small breadboard, fingerprint scanner chip, the fingerprint scanner, and the LCD were all secured using double sided tape. The toggle switches were secured using nuts and washers. All power cords that were run into the box have a knot tied in them so that if they are pulled hard the circuitry in the box will not be disturbed. Overall, all of the components were secured in the box and the design is relatively durable.

7.7 Initial Prototype

After having all of the components up and running individually, they were combined to form an initial prototype. Essentially this consisted of putting all of the circuitry on a breadboard with the components placed in a box. The initial prototype is shown in figure 19 below:

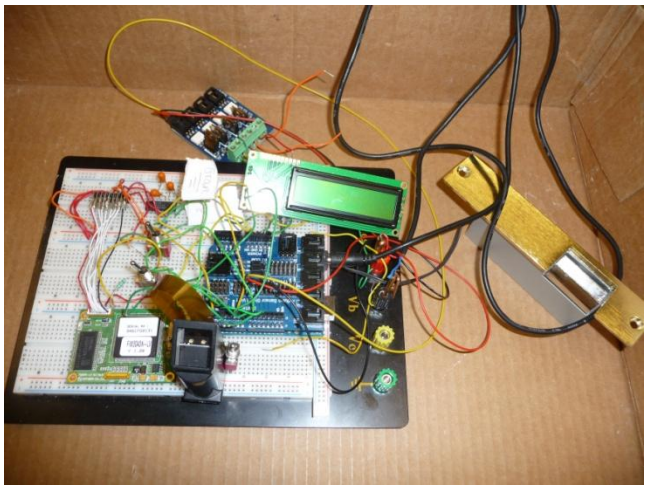

Figure 19: Initial prototype

The accuracy of the fingerprint scanner was tested on this initial prototype because we were worried that extending this to a final prototype may result in some unseen problems which would delay the testing process.

7.8 Final Prototype

Upon the completion of the testing of the accuracy of the identification process, careful circuit diagrams were made documenting all of the connections of the initial prototype. Then the box was cut, and all the components were secured within the enclosure. Next, using those diagrams I remade all of the connections using a miniature breadboard secured to the base of the box. The interior of the final prototype may seen in figure 20 below:

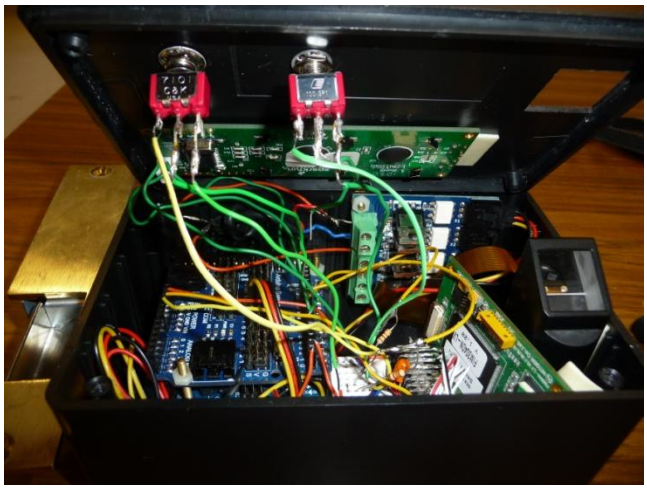

Figure 20: Looking inside the final prototype

To make the system even more user friendly, labels were added to each of the toggle switches to clearly show which was to enroll a new user and which was to identify a user. The final prototype as seen by users may be seen below in figure 21:

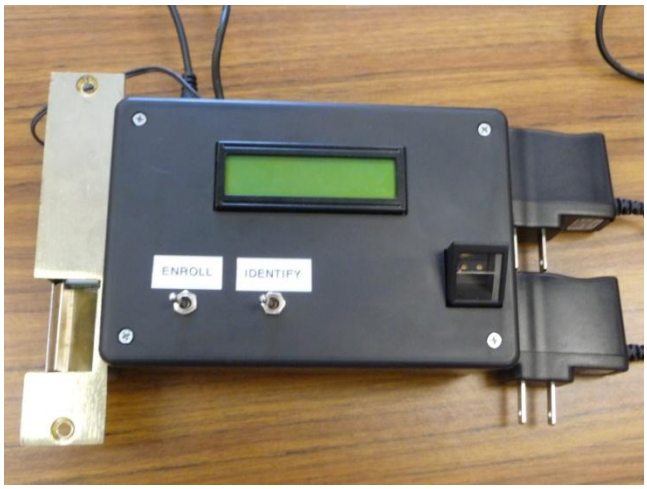

Figure 21: Final prototype

8. Performance Estimates and Results

The identification rate of the fingerprint scanner was the main evaluation of the performance of this project. Because fingerprint data were collected for the tests, an application was submitted to the Human Subjects Review Committee (HSRC) at Union College prior to

collecting the data. Upon approval from the HSRC, sixteen (16) users were enrolled in the database. After all of the users were saved in the database, they were then asked to use the system a total of ten times to attempt to identify their fingerprint. Opening the locker with the corresponding identification number was deemed a success, while not being able to open the lock was a failure. The results of this study are shown below in table 2:

	Identification Trial Number									
Participant ID	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	0	1	1	1	1	1
8	0	0	0	0	0	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1

Table 1: Results from the identification study

In table 2, a 1 represents a success where the lock was opened with the proper identification number while a 0 represents a failure where access was denied to the lock. Users were also asked to comment on the user friendliness of the system and provide suggestions for improvements which were later implemented on the final design. The overall accuracy of the identification process was found to be 96.25%. Though the false positive rate was not officially studied, it should be noted that no users were allowed to open the lock with the incorrect identification number. The approximate time from flipping the identification toggle switch to opening the lock

was approximately 9 seconds, which is comparable to opening a combination lock. For example, it took me about 12 seconds to open my school mailbox.

The two users who were not able to identify themselves all ten times gave possible explanations for the failures. The first user, who accessed the locker 9 out of 10 times, stated that they had removed their finger too quickly and this may have hindered the identification of the fingerprint. This is one problem that may reoccur with this system and will reduce the accuracy of the scanner. The other user, who had multiple failures, stated they put their finger down after the scanner's light on indicating it was waiting for a fingerprint. When they put their finger down before the scan started they were identified correctly every time. This problem was later experimented with and found that it is not reoccurring and seemed to only affect one user. Because the system was still in its initial prototype phase, many users thought that it was messy and would be confusing to use without my instructions of what each switch was responsible for. This was remedied by placing all components within an enclosure and clearly labeling each switch. Users thought that the LCD screen clearly indicated what was going on and more specifically when the scanner will be operating. I also noticed that some print statements were extraneous and could be removed for the final prototype.

9. Production Schedule

Many portions of the project began early in the semester and were continued for multiple weeks, oftentimes being put on the "side burner" for more important tasks. Overall, the most significant tasks completed this term can be summarized as constructing the MAX232 circuit, coding the Arduino to control the fingerprint scanner, building an initial prototype, testing the

accuracy of the fingerprint scanner, and creating the final prototype. The production schedule is laid out week by week for the winter semester below in table 3:

Week	Description of work
1	Began construction/research on a MAX232 circuit, experimented with coding the scanner, became more familiar with available commands
2	Continued with MAX232 circuit, attempted to code more, experimented with the MOSFET device, and researched locks
3	Finished the MAX232 circuit, connected the LCD for debugging purposes, connected to the scanner, began to attempt to enroll a user
4	Finished the enroll function, finished an identification function, wrote all the cases for both functions, have a initial prototype
5	Worked on enrolling a master with an ID number, experimented with pushbuttons, performed lots of debugging of my code, cleaned up print statements
6	Coded it so unplugging has no effect on the ID numbers, experimented with the quality of image function, inserted this function into the enroll process, began testing
7	Filled the database with 16 users, found identification to be 96.25%, received positive constructive feedback to improve the system, began preparing for the enclosure
8	The enclosure was cut, secured all components within the enclosure, remade all connections, made final preparations for the presentation/demo day

Table 2: Production schedule for the winter term

The table represents a synopsis of work performed during the winter semester. More extensive details of work done may be seen in Appendix 3 “Work Log” following this report.

10. Cost Analysis

The major components that were used in this project are listed below in table 4 by unit cost, quantity of the components used, and the total cost for each component:

Component	Unit Cost	Quantity	Total Cost
Arduino Duemilanove	29.95	1	29.95
Nitgen RS232 Fingerprint Scanner	129.95	1	129.95
Jumper Wire-MLX to MLX	1.95	1	1.95
Arduino Sensor Shield	14.98	1	14.98
Arduino 4 Route MOSFET Button	17.98	1	17.98
Arduino Common Sensor Cable	1.38	1	1.38
Lee Electric Door Strike 12V DC Lock	29.96	1	29.96
Wall Adapter	5.95	2	11.90
Project Enclosure	4.99	1	4.99
Total			243.04

Table 3: Cost analysis for all major components

Other minor components such as toggle switches, connectors, and basic wires were obtained from the ECE department and Gene Davidson. Recall from section 4.1 that the goal for this project's overall was to be less than $250.00. The total price seen above in table 4 was $243.04, just under that goal. It is possible that some of the cost of the components may be reduced. For example, the MLX to MLX jumper wire could be easily be constructed, bypassing the need for the supplier and thus reducing the overall price. It may also be possible to replace other components with custom made ones, but due to the time constraint of this project that was not feasible. It should still be noted that the overall cost of this project was under the competitive prices of other systems currently available on the market.

11. User's Manual

The following sections describe various aspects of this system and how to use them. It includes navigating the two sets of code used in this project as well as operating the final prototype.

11.1 Final Prototype Code

The code used for the final prototype was designed to print clear statements to the LCD, increasing the user friendliness of the system. It consists of various functions all of which may be edited for future use. The main functions that the final code includes of are connecting to the scanner, entering master mode, leaving master mode, enrolling a new user, checking the quality of a fingerprint scan, deleting a user, and identifying a user. While the system works well there are many small changes that may be made to this code such as using four templates, changing the duration of the fingerprint scan (how long it waits for a fingerprint to be place down), the number of the quality control, etc. The quality control was set to be 65 out of 100. This requires that the enrolled fingerprint image has at least that quality. Increasing the quality control makes it harder to enroll in the system, but also increase the accuracy of the identification process. The code itself is broken down to make it more concise and clear to understand, and in general it should be straightforward to follow.

11.2 Master Code

The master code allows for all prints to be deleted and a new master fingerprint to be enrolled into the system. To upload the master code, the project enclosure's lid must be removed so that the internal components are all revealed. The serial pins on the Arduino board must be disconnected as the Arduino development environment does not allow code to be uploaded to the board while the serial pins are connected. While having a separate master code is a slight nuisance, it increases the security of the system. The master code is not as clean cut (printing is not as professional) as the final prototype code, but in general it performs a few tasks all of which may be edited to the master's desires. This code runs automatically and is not based on toggle switches.

The code begins by verifying the master fingerprint and once in master mode it deletes all fingerprints saved on the chip, including the master fingerprint. With no master fingerprint, the system uses a separate command to enter master mode (not available if there are any master fingerprints stored in the database). Then, the new master is enrolled using the standard two template technique. Identification numbers begin at 0x30303030 where the zeros can be extracted to represent an ID of 0000. Thus, a master identification number of 1313 corresponded to 0x31333133. This was chosen to be different than the identification numbers that the users are assigned to avoid any possible confusion. Note that this identification number is extremely important as it is required to enter master mode later. That being said, if the number is changed, be sure to reflect those changes in all locations of the both the final prototype and master codes or else confusion will arise as you will not be able to enter master mode. Don't forget to change the corresponding checksums as well. If you change the code to let the system pick an identification number, it begins with 0x30303030 or 0000. Once the master code has finished running (enrolled a new master) then the final prototype code can be uploaded, the serial pins reconnected, and the lid may be re-secured. If you receive a message like "FIM not responding" or something similar, most likely the serial pins were not reconnected. The complete master code can be seen in Appendix 4 "Arduino Master Code".

11.3 Operating the Final Prototype

Using the final prototype should be relatively straightforward and clear. On the top of the enclosure there are two toggle switches labeled "Enroll" and "Identify". Flipping the enroll switch over for at least 1.25 seconds initiates the enrollment of a new user. This process begins with the verification of the master fingerprint. If the master is not present, then you will not be able to enroll a new user. This prevents anyone from coming up to the system, enrolling their

fingerprint in the database, and consequently being able to open the lock and retrieve other peoples' contents. After the master is verified, the scanner will take the first template of the new user. Once the template has been taken the quality of the fingerprint image is checked. Currently the cutoff is 65 out of 100. If the quality check is greater than that number, then the second template will be taken. If it is less than 65 than you will be forced to leave master mode, and start the enroll process from the beginning. The second template is also checked for the quality of the image. If the image is good enough the user is officially saved in the database with the next available identification number, which will be displayed on the LCD screen. If the quality check returns poor image quality, the user fingerprint must be deleted. That is because anytime the second template is taken the fingerprint is saved to the chip. But, we do not want the poor quality image to be saved, so the master is verified once again and this print is deleted from the database. Then, the user may try to enroll the user again from the beginning.

There are several things that can be done to increase the likelihood that the minimum fingerprint quality check will be achieved. First and foremost, if the scanner is dirty due to fingerprint smudges it is very likely the quality of the fingerprint images will be poor. A good method of cleaning the scanner is to wipe it down with a clean tissue after the master has been verified. Another useful tip is to use the dominant index finger. The dominant hand will be stronger and is more likely to be sturdy and steady while the scan occurs. Movement during the scan, or physically moving the scanner by brute force, generally reduces the quality of the images and sometimes produces errors. It is almost important to notice where the "good" data on your finger is. It is intuitive to align your finger with the top of the glass part of the scanner. However, your fingerprint details are farther down your finger and thus aligning the top of your finger with the absolute edge of the scanner typically provides better images as well. All of these

techniques generally result in better images, but I personally believe that some people's fingers are just better than others. For example, some people had no trouble enrolling their finger with a dirty scanner, moving the scanner, and using a non-index finger while others could not enroll their finger unless all of these cases occurred.

The other toggle switch is responsible for identifying users. The identification process is independent of the master and can be performed by anybody. Switching the toggle switch for more than 1.25 seconds causes the fingerprint scanner to take one image. It checks the database and if the person is in it, then the lock is opened. The user's identification number is also displayed on the LCD screen. If the person is not in the database (or could not be recognized) then "access denied" is displayed on the LCD. The welcome screen reappears and users may continue to try to identify themselves if necessary.

Because only the image quality of the enrollment process is controlled, it is not necessary to clean the scanner every time you want to identify yourself. While this may increase the accuracy of identification, no problems were found during the identification testing because of a dirty scanner.

Both functions of the system rely on the flip of a toggle switch. Once either switch has been flipped for at least 1.25 seconds and the code begins to run, the switch should be flipped back. Otherwise, the code will continuously repeat itself. This system is currently programmed to have sixteen users with identification numbers incrementing from 0000 to 000F but this could be changed to add more space for future users.

12. Discussion, Conclusions, and Recommendations

Lockers that are opened by traditional locks, like combination locks, can be annoying and sometimes difficult to use because they require prior knowledge. For lockers where combination locks are often used, it can be difficult to remember a series of numbers as well as how to enter them. An alternative to this type of system is one that relies on biometric data to open a locker.

This project provides a solution to the traditional locker scheme as it control access to a lock by using a fingerprint scanner. The fingerprint scanner allows users to both enroll their fingerprint in the database and attempt to identify themselves. Enrolling a new user require the master fingerprint to be verified, making the user database exclusive. However, identification is independent of the master and can be attempted by anyone. If the user is in the database then their locker is opened but if they are not then access is denied. All decisions are made by the Arduino Duemilanove. It controls the commands that are sent to the fingerprint scanner, interprets what is received from the scanner, prints to the LCD screen, and controls the opening and closing of the lock.

The final prototype was a successful biometric system that controlled access to a lock and it met most of the goals initially proposed for this project. Recall that one of the most important goals of this system was to design a user friendly prototype. During the testing process, positive feedback and constructive criticism were received and later implemented to improve the final prototype. The final prototype appeared professional and the LCD screen provided clear instructions to the users. For this system to be effective, the identification rate of the scanner had to be efficient enough to allow users a reliable way to open their locker. Through testing, we found that the scanner correctly identified the user 96.25% of the time. Keeping the false positive rate of this system to a minimum was important, as you want the locker to remain private and

exclusive to that user. While this was not specifically tested, no users were identified with the wrong identification number. The last consideration was to design a system that took comparably the same amount of time to use as the standard combination lock. The system took 9 seconds from flipping the toggle switch to opening the locker. It took approximately 12 seconds to open my mailbox which is a traditional combination locker. The greatest overall success of this project was reaching the final prototype that was an efficient biometric system that controlled a lock.

Many lessons were learned during this project. As with many engineering projects, there were lots of tradeoffs considered in the design phase. For instance, when deciding what item should be used for a component tradeoffs like cost, performance, ease of use, compatibility with other parts, and how it will be powered had to be considered. After I wrote the full program for the initial prototype I began backing up my code and saving it daily. This paid off as sometimes when small changes were made to the code it would sometimes work incorrectly, but I could revert to the previously working code to help me fix it. Before transitioning from the initial to final prototypes, clear diagrams were made of all connections. This helped a lot with constructing the final system and without these diagrams it would have been much more difficult and time consuming. Another lesson that I learned was the difficulty in making a professional final product. I did not realize how difficult it would be to measure out a project enclosure to prepare it to be cut for the components. I also did not foresee how challenging it would be to place all components within the enclosure in a neat and structurally supportive manner. Constructing the final prototype made me realize how much effort goes into making a professional product as I thought it would be easy to make it having the initial prototype already completed.

The end product of this project was a working biometric system that controlled the access to a lock. However, there are still some things that could be done to improve this system. In the original scope of this project, it was stated that the goal was to control access to a set of lockers. This could be done by using the ID numbers of the users to correspond with multiple locks. Due to the expense of locks and the difficulty of mounting them in a locker, this system provided a proof of concept that this task could be completed. Besides that, there are many small changes that could be made to the commands that control the fingerprint scanner which may improve the performance of the system. For instance, you could easily switch to four template mode (currently takes two templates), change the duration of time which the scanner waits for a fingerprint, alter the quality check during enroll, etc. All of these changes are capable of improving the system, but it should not overshadow the fact that a professional working design was constructed and works well.

13. References

[1]Global Security; http://www.globalsecurity.org/security/systems/biometrics-history.htm, 2007.
[2] K. Karu and A.K. Jain, "Fingerprint Classification", Pattern Recognition, Vol. 29, No. 3, pp. 389-404, 1996.
[3] A. Jain and S. Pankanti, "Fingerprint Classification and Matching" Handbook for Image and Video Processing, A. Bovik (ed.), Academic Press, April 2000.
[4] R.M. Bolle, J.H. Connell, S. Pankanti, N.K. Ratha, and A.W. Senior, Guide to Biometrics. Springer Science+Business Media, LLC. 2004, New York.
[5]The Keyless Lock Store; http://www.nokey.com/biomaccon.html, 2010.
[6] Union College Human Subjects Review Committee; http://www.union.edu/Resources/Academic/hsrc/index.php
[7] Sparkfun Electronics; http://www.sparkfun.com/
[8] Arduino Duemilanove; http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove
[9] Sparkfun Tutorials; http//www.sparkfun.com/tutorials/104
[10] RS-232 Signals and RS232 voltage levels; http://www.radio-electronics.com/info/telecommunications_networks/rs232/signals-voltages-levels.php
[11] MAX232 Serial Level Converter; http://sodoityourself.com/max232-serial-level-converter/
[12] Nitgen: RS-232C serial protocol for Stand-Alone Fingerprint Recognition Device
[13] Emartee: http://www.emartee.com/

[14] 12V DC Electric Strike Lock; http://www.smarthome.com/519012/Electric-Door-Strike-12VDC-220-12/p.aspx
[15] Project Enclosure; http://www.radioshack.com/product/index.jsp?productId=2062283

14. Appendices

1. Arduino Final Prototype Code

2. Arduino MOSFET Testing Code

3. Work Log

4. Arduino Master Code

Appendix 1: Arduino Final Prototype Code

```
#include <SoftwareSerial.h>
#include <EEPROM.h>

#define maxim 50
//#define txPin 2 // LCD Pin
//#define powerLCD 3
byte txPin = 6;
byte powerLCD = 5;

SoftwareSerial LCD = SoftwareSerial(0, txPin);
// since the LCD does not send data back to the Arduino, we should only define the txPin
int d = 0;
int i;
int f;
int header_rx;
int M;
int g;
int ID[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C,
0x0D, 0x0E, 0x0F};  //change back the first 2 numbers to be 0 and 1.
int s1pin = 7;
int identifyPin = 8;
int enrollPin = 9;
int powerPinI = 10;
int powerPinE = 11;
  int A = 0;
  int B = 0;
  int C = 0;
  int D = 0;
int conn;
int inc;
int quality_check = 0;

void setup()
{
  pinMode(powerPinI, OUTPUT);
  pinMode(powerPinE, OUTPUT);
  digitalWrite(powerPinI, HIGH);
  digitalWrite(powerPinE, HIGH);
  pinMode(enrollPin, INPUT);
  pinMode(identifyPin, INPUT);
  pinMode(s1pin, OUTPUT);
  pinMode(powerLCD, OUTPUT);
  digitalWrite(5,HIGH);
  pinMode(txPin,OUTPUT);
```

```
  LCD.begin(9600);
  Serial.begin(9600);
  delay(25);
  LCDon();
  delay(4000);
  delay(25);
  Serial.flush();
  ReqConn();
  if (conn == 100){
    Welcome();
    delay(2000);
  }
}

void loop()
{
  int A = digitalRead(identifyPin);
  int B = digitalRead(enrollPin);
  if (A == 1){
    delay(1250);
    int C = digitalRead(identifyPin);
    if (C == 1){
      identify();
    }
  }
  if (B == 1){
     delay(1250);
     int D = digitalRead(enrollPin);
     if (D == 1){
       clearLCD();
       firstline();
       LCD.print("Enrolling a new    user soon");
       delay(2000);
       Enroll_ID1();
     }
  }

}

void ReqConn(){
  conn = 0;
  Serial.flush();
  int rx[24];
  int bytes[] = {0x7E, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01};
```

```
  for (i = 0; i < 25; i++) {
  Serial.print(bytes[i], BYTE);
  }

  delay(2000);
  Incoming();
  if (inc == 0x0A){

  if (Serial.available() >= 24) {
    for(i=0;i<24;i++){
    rx[i]=Serial.read();
    }
  }

//Make sure to change to 0x01 but for testing lest say its 0x20 which is space
if (rx[3] == 0x01 && rx[7] == 0x01){
  clearLCD();
  firstline();
  LCD.print("Connected to FIM");
  delay(2000);
  conn = 100;

}else{
    clearLCD();
    firstline();
    LCD.print("No connection tothe scanner");
//    secondline();
//    LCD.println(" to the scanner");
    delay(2500);
    clearLCD();
    firstline();
    LCD.print("Please try again");
    delay(2500);
    }
  }
}

//Clears the LCD display of all characters
void clearLCD(){
LCD.print(0xFE, BYTE);   //command flag
LCD.print(0x01, BYTE);    //position which starts the second line
}

//Turn on the LCD Display
void LCDon(){
LCD.print(0xFE, BYTE);   //command flag
```

```
LCD.print(0x0C, BYTE);    //position which starts the first line  change to 0x0C if you don't
want a blinking cursor
}

void firstline(){
LCD.print(0xFE, BYTE);   //command flag
LCD.print(0x80, BYTE);    //position which starts the first line
}

//The following function sets the cursor to the first position on the first line
void secondline(){
LCD.print(0xFE, BYTE);   //command flag
LCD.print(0xC0, BYTE);    //position which starts the second line
}

void shiftright(){
  LCD.print(0xFE, BYTE);   //command flag
  LCD.print(0x18, BYTE);    //position which starts the second line
}

//This command will attemp to enter "master mode" based on the FP in the database designated
as masters, the masters ID is require as well.
void EnterMaster(){
  M = 0x00;
  clearLCD();
  firstline();
  LCD.print(" Verifying the     master....");
  delay(1500);
  clearLCD();
  firstline();
  LCD.print("fingerprint in 3...");
  delay(1000);
  LCD.print("2...");
  delay(1000);
  LCD.print("1...");
  delay(500);
  //  secondline();
//  LCD.print(" master FP now");
  Serial.flush();
  int rx[24];
  int f;

  int master[] = {0x7E, 0x00, 0x00, 0x00, 0x2F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x39, 0x31, 0x33,
0x31, 0x33, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC8};
```

```
  for (i = 0; i < 39; i++) {
  Serial.print(master[i], BYTE);
  }

  delay(5250);
  Incoming();
  if (inc == 0x0A){

  if (Serial.available() >= 24) {
    for(i=0;i<24;i++){
    rx[i]=Serial.read();
    }
  }
  switch (rx[7]) {
    case 0x01:
      clearLCD();
      firstline();
      LCD.print("In Master Mode");
      delay(2000);
      M = 0x0A;
      break;
    default:
      clearLCD();
      firstline();
      LCD.print("Failed to enter master mode");
//      secondline();
//      LCD.println(" master mode");
      delay(2000);
      M = 0x0F;
      break;
  }
  }

}

 void Incoming(){
  inc = 0x0A;
    for(d = 0;d < maxim;d++) {
    if (Serial.available() > 0) {
      header_rx = Serial.read();
      delay(50);
    if (header_rx == 0x7E){
      break;
  }
  else{
    clearLCD();
```

```
    firstline();
    LCD.print(" The scanner is unresponsive");
    delay(2000);
    clearLCD();
    firstline();
    LCD.print("Please reset the system's power");
//    secondline();
//    LCD.print("Please try again");
    delay(2000);
    inc = 0x0F;
    break;
  }
  }
  }

  if (header_rx == 0x7E) {
    for(d = 0;d < maxim;d++) {
      if (Serial.available() >= 24)break;
      delay(75);
    }
  } else{
    clearLCD();
    firstline();
    LCD.print(" The scanner is unresponsive");
    delay(2000);
    clearLCD();
    firstline();
    LCD.print("Please reset the system's power");
//    secondline();
//    LCD.print("Please try again");
    delay(2000);
    inc = 0x0F;
  }
  }

 void identify(){
    int f;
    Serial.flush();
    int rx[38];
    clearLCD();
    firstline();
    LCD.print("Identifying scanin 3...");
    delay(1500);
    LCD.print("2...");
    delay(1250);
    LCD.print("1...");
```

```
    delay(500);

    int identify[] = {0x7E, 0x00, 0x00, 0x00, 0x12, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x12};

    for (i = 0; i < 25; i++) {
      Serial.print(identify[i], BYTE);
      }
    delay(5250); ///Assuming it needs the same delay in case of a time out
    Incoming();
    if (inc == 0x0A){

    if (Serial.available() >= 24) {
      for(i=0;i<38;i++){
        rx[i] = Serial.read();
      }
    }
    switch (rx[7]) {
      case 0x01:
        clearLCD();
        firstline();
        LCD.print("  Opening your  locker now with");
//        secondline();
//        LCD.print("   locker now");
        digitalWrite(s1pin, HIGH);
        delay(3000);
        clearLCD();
        firstline();
        digitalWrite(s1pin, LOW);
        LCD.print("ID: ");
        LCD.print(rx[24], HEX);
        LCD.print(rx[25], HEX);
        LCD.print(rx[26], HEX);
        LCD.print(rx[27], HEX);
        delay(2000);
        clearLCD();
        Welcome();
        break;
      case 0x07:
        Slow();
        Welcome();
        break;
      case 0x12:
        clearLCD();
        firstline();
        LCD.print("Couldnt identifythe print in time");
```

```
//       secondline();
//       LCD.println(" finger in time");
       delay(3000);
       clearLCD();
       firstline();
       LCD.print("Please try again");
//       secondline();
//       LCD.println("   scan again");
       delay(3000);
       Welcome();
       break;
     case 0x0D:
       clearLCD();
       firstline();
       LCD.print("Couldnt identify");
//       secondline();
//       LCD.println("     finger");
       delay(3000);
       clearLCD();
       firstline();
       LCD.print("Please try again");
//       secondline();
//       LCD.println("   scan again");
       delay(3000);
       Welcome();
       break;
     case 0x02:
       clearLCD();
       firstline();
       LCD.print("Access denied!");
       delay(3000);
       Welcome();
       break;
     default:
       Unknown();
       Welcome();
   }
   }
 }

void LeaveMaster(){
  Serial.flush();
  int rx[24];

  int master[] = {0x7E, 0x00, 0x00, 0x00, 0x26, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x26};
```

```
  for (i = 0; i < 25; i++) {
  Serial.print(master[i], BYTE);
  }

  delay(2000);
  Incoming();
  if (inc == 0x0A){

  if (Serial.available() >= 24) {
    for(i=0;i<24;i++){
    rx[i]=Serial.read();
    }
  }
  switch (rx[7]) {
    case 0x01:
      clearLCD();
      firstline();
      LCD.print(" You have left    master mode");
//      secondline();
//      LCD.println("  master mode");
      delay(3000);
      break;
    case 0x03:
      clearLCD();
      firstline();
      LCD.print("You were not in master mode");
//      secondline();
//      LCD.println("  master mode");
      delay(3000);
      break;
    default:
      clearLCD();
      firstline();
      LCD.print("Problem leaving master mode");
//      secondline();
//      LCD.println(" master mode");
      delay(3000);
      clearLCD();
      firstline();
      LCD.print("Please try again");
//      secondline();
//      LCD.println("   scan again");
      delay(3000);
      break;
  }
```

```
  }
}


void Welcome(){
  clearLCD();
  firstline();
  LCD.print("Welcome, please select an option");
//  secondline();
//  LCD.println("Select an option");
}

void Slow(){
  clearLCD();
  firstline();
  LCD.print("Your finger scan  was too slow");
//  secondline();
//  LCD.println("  was too slow");
  delay(3000);
  clearLCD();
  firstline();
  LCD.print("Please try again");
//  secondline();
//  LCD.println("   scan again");
  delay(3000);
  }

void NotMaster(){
    clearLCD();
    firstline();
    LCD.print("Need the masters");
    secondline();
    LCD.println("FP for DB entry");
    delay(3000);
    clearLCD();
    firstline();
    LCD.print("   Please try");
    secondline();
    LCD.println("     again");
    delay(3000);
}

void TwoFingers(){
    clearLCD();
    firstline();
    LCD.print(" The templates  do not match");
```

```
//    secondline();
//    LCD.println("   dont match");
    delay(3000);
    clearLCD();
    firstline();
    LCD.print("Please try again");
//    secondline();
//    LCD.println("   scan again");
    delay(3000);
    }

void Unknown(){
  clearLCD();
  firstline();
  LCD.print("Unknown failure,please try again");
//  secondline();
//  LCD.println("Please try again");
  delay(3000);
}

void Failure(){
    clearLCD();
    firstline();
    LCD.print("Your finger scanhas failed");
//    secondline();
//    LCD.println("   has failed");
    delay(3000);
    clearLCD();
    firstline();
    LCD.print("Please try again");
//    secondline();
//    LCD.println("   scan again");
    delay(3000);
}

void Enroll_ID1(){
  EnterMaster();
  if (M == 0x0A) {
      int rx[24];
      int f;
      g = EEPROM.read(0);
      int checksum = ID[g] + 0x80;
      int idnumber = 0x30 + ID[g];
      clearLCD();
      firstline();
      LCD.print("  Scanning the  first template..");
```

```
    delay(1750);
    clearLCD();
    LCD.print("in 3...");
//    secondline();
//    LCD.println("template in 3...");
    delay(1000);
    LCD.print("2...");
    delay(1000);
    LCD.print("1...");
    delay(500);
    Serial.flush();
    int enroll_id1[] = {0x7E, 0x00, 0x00, 0x00, 0x33, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x1A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x4D, 0x30, 0x30,
0x30, idnumber, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x30, 0x30, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, checksum};

    for (i = 0;i < 55;i++){
      Serial.print(enroll_id1[i], BYTE);
    }

    delay(5250);
    Incoming();
    if (inc == 0x0A){

    if (Serial.available() >= 24){
    for (i = 0;i < 24;i++){
      rx[i] = Serial.read();
    }
   }


    switch (rx[7]) {
      case 0x01:
        Quality();
        if (quality_check >= 65){
          Enroll_ID2();
        }
        else{
          clearLCD();
          firstline();
          LCD.print("Poor fingerprint    quality");
          delay(2500);
          clearLCD();
          firstline();
          LCD.print("Please try again");
          delay(2000);
```

```
          LeaveMaster();
          Welcome();
         }
          break;
       case 0x07:
         Slow();
         LeaveMaster();
         Welcome();
         break;
       case 0x03: //case shouldnt happen with the M thing, see
2_5_2011_MASTER_HYPYERTERMINAL
         NotMaster();
         Welcome();
         break;
       case 0x02:
         Failure();
         LeaveMaster();
         Welcome();
         break;
       case 0x0D:
         Failure();
         LeaveMaster();
         Welcome();
         break;
       default:
         Unknown();
         for (i = 0;i < 24;i++){
           LCD.print(rx[i]);
         }
         LeaveMaster();
         Welcome();
         break;
     }
     }
   }
else{
     clearLCD();
     firstline();
     LCD.print("  The master's  fingerprint is..");
     delay(2000);
     clearLCD();
     firstline();
//     secondline();
     LCD.print("needed to enrolla new user");
     delay(2000);
     Welcome();
```

```
    }

}

void Enroll_ID2(){
  Serial.flush();
  int rx_2[24];
  int f;
  clearLCD();
  firstline();
  LCD.print(" Scanning the   second template");
  delay(1750);
  clearLCD();
  firstline();
  LCD.print("in 3...");
  delay(1000);
  LCD.print("2...");
  delay(1000);
  LCD.print("1...");
  delay(500);
  int enroll_id2[] = {0x7E, 0x00, 0x00, 0x00, 0x33, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x34};

  for (i = 0;i < 25;i++){
    Serial.print(enroll_id2[i], BYTE);
  }
  delay(5250);
  Incoming();
  if (inc == 0x0A){

  if (Serial.available() >= 24){
    for (i = 0;i < 25;i++){
      rx_2[i] = Serial.read();
    }
  }

  switch (rx_2[7]) {
    case 0x01:
      Quality();
      if (quality_check >= 65){
        clearLCD();
        firstline();
        LCD.print(" The print was  saved with the..");
        delay(2000);
        clearLCD();
        firstline();
```

```
    LCD.print("ID: 3030303");
    LCD.print(ID[g]);
    delay(3000);
    g = g + 0x01;
    EEPROM.write(0,g);
    LeaveMaster();
   }
   else{
    LeaveMaster();
    clearLCD();
    firstline();
    LCD.print("Poor fingerprint    quality");
    delay(2500);
    Delete();
   }
   Welcome();
   break;
 case 0x07:
   Slow();
   LeaveMaster();
   Welcome();
   break;
 case 0x03: //should only get this case in Part 1
   NotMaster();
   Welcome();
   break;
 case 0x02:
   Failure();
   LeaveMaster();
   Welcome();
   break;
 case 0x0D:
   Failure();
   LeaveMaster();
   Welcome();
   break;
 case 0x0E:
   TwoFingers();
   LeaveMaster();
   Welcome();
   break;
 default:
   Unknown();
   for (i = 0;i < 24;i++){
    LCD.print(rx_2[i]);
   }
```

```
    LeaveMaster();
    Welcome();
    break;
  }
  }
}

void Delete(){
  EnterMaster();
  clearLCD();
  firstline();
  LCD.print("Trying to delete the print now");
  delay(2000);
  int checksum = ID[g] + 0xC0; //Need to devise a better scheme for this but for now its ok.
  int idnumber = 0x30 + ID[g];
  if (M == 0x0A){
    Serial.flush();
    int rx[24];
    int f;
  //obviously need to put the ID in here and make it longer and also change how many to print
    int delete_id[] = {0x7E, 0x00, 0x00, 0x00, 0x22, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x2C, 0x30, 0x30,
0x30, idnumber, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, checksum};

    for (i = 0;i < 39;i++){
      Serial.print(delete_id[i], BYTE);
    }
    delay(2000);
    Incoming();
    if (inc == 0x0A){

    if (Serial.available() >= 24){
      for (i = 0;i < 25;i++){
        rx[i] = Serial.read();
      }
    }
    switch (rx[7]){
      case 0x01:
        clearLCD();
        firstline();
        LCD.print("  Deleted your    fingerprint");
        delay(2000);
        clearLCD();
        firstline();
        LCD.print("Please try again");
        delay(2000);
```

```
        LeaveMaster();
        Welcome();
        break;
      case 0x03:
        NotMaster();
        break;
      case 0x02:
        Failure();
        LeaveMaster();
        Welcome();
        break;
      default:
        Unknown();
        LeaveMaster();
        Welcome();
        break;
    }
    }
  }
else{
 Delete();
  }

}

void Quality(){
  int f;
  Serial.flush();
  int rx[24];
  clearLCD();
  firstline();
  LCD.print("  Checking the   print quality");

  int quality[] = {0x7E, 0x00, 0x00, 0x00, 0x68, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x68};

  for (i = 0; i < 25; i++) {
    Serial.print(quality[i], BYTE);
    }
  delay(2500); ///Assuming it needs the same delay in case of a time out
  Incoming();
  if (inc == 0x0A){

  if (Serial.available() >= 24) {
    for(i=0;i<24;i++){
      rx[i] = Serial.read();
```

```
    }
  }
  switch (rx[7]) {
    case 0x01:
      quality_check = rx[11];
      break;
    default:
      Unknown();
  }
  }
  delay(2000);
}
```

Appendix 2: Arduino MOSFET Testing Code

```
int s1Pin = 4;
int s2Pin = 5;
int s3Pin = 6;
int s4Pin = 7;

void setup() {
  pinMode(s1Pin, OUTPUT);
  pinMode(s2Pin, OUTPUT);
  pinMode(s3Pin, OUTPUT);
  pinMode(s4Pin, OUTPUT);
}

void loop() {
  int i;

  digitalWrite(s1Pin, HIGH);
  delay(500);
  digitalWrite(s1Pin, LOW);
  delay(500);
  digitalWrite(s2Pin, HIGH);
  delay(500);
  digitalWrite(s2Pin, LOW);
  delay(500);
  digitalWrite(s3Pin, HIGH);
  delay(500);
  digitalWrite(s3Pin, LOW);
  delay(500);
  digitalWrite(s4Pin, HIGH);
  delay(500);
  digitalWrite(s4Pin, LOW);
  delay(1000);

  for (i = 0; i < 10; i ++) {
    digitalWrite(s1Pin, HIGH);
    delay(500);
    digitalWrite(s1Pin, LOW);
    delay(500);
  }

  for (i = 0; i < 100; i ++) {
    digitalWrite(s2Pin, HIGH);
    delay(50);
    digitalWrite(s2Pin, LOW);
    delay(50);
```

```
  }
    for (i = 0; i < 100; i ++) {
    digitalWrite(s3Pin, HIGH);
    delay(50);
    digitalWrite(s3Pin, LOW);
    delay(50);
  }
    for (i = 0; i < 100; i ++) {
    digitalWrite(s4Pin, HIGH);
    delay(50);
    digitalWrite(s4Pin, LOW);
    delay(50);
  }
  delay(1000);
   digitalWrite(s1Pin, HIGH);
   digitalWrite(s2Pin, HIGH);
   digitalWrite(s3Pin, HIGH);
   digitalWrite(s4Pin, HIGH);

   delay(5000);
   digitalWrite(s1Pin, LOW);
   digitalWrite(s2Pin, LOW);
   digitalWrite(s3Pin, LOW);
   digitalWrite(s4Pin, LOW);

   delay(1000);
  digitalWrite(s1Pin, HIGH);
  digitalWrite(s3Pin, HIGH);
  delay(1000);
     digitalWrite(s1Pin, LOW);
   digitalWrite(s3Pin, LOW);
   delay(1000);

      digitalWrite(s2Pin, HIGH);
   digitalWrite(s4Pin, HIGH);
   delay(2000);
      digitalWrite(s2Pin, LOW);
   digitalWrite(s4Pin, LOW);

}
```

Appendix 3: Work Log

499 Weekly Log

Week 1:

1 Hour
Started to write Arduino Code for getting it to work with the FIM and started to design circuit for RS232 to TTL logic converter. I tried to mimic the code from when I met with Prof. Hedrick last November where we were able to get the version information, essentially tweaking it so that it is in the proper language for the Arduino. I'm a bit confused about serial print, if I want it to be 0x7E or 0x7E, BYTE. Either way I have both codes saved.

1 Hour
Constructed possible circuit of serial communication between FIM and the Arduino. This was the one that Professors Cotter and Hedrick designed. Tested out some hex printing with th e Arduino to better figure out how to send the commands. Figured out I can save space by using arrays to print long lists of serial commands compared to individual print lines. This saves some memory which may become useful for combining multiple sections of codes.

1 Hour
Designed and tested some use of toggle switches to wake up the FIM from the Arduino. I had this idea because the Arduino will need to know when to contact the FIM. A quick flip of toggle switch executes the whole set of commands so flipping it on then off will say I want to enroll, or i want to identify, This is the way the Arduino will know you are there and want to us the system. Could potentially put in delete commands there as well. or manually send those with a separate program.

3 Hours:
Constructed the RS232 to TTL circuit using the MAX232CPE chip with help of Prof. Hedrick. Tried to test the Arduino code with it but I was not able to enroll a fingerprint. Will continue to investigate this. Started using RX and TX but they are annoying since you can't use them with USB, so switched to "softwareserial" and made my own pins. The tx and rx lights don't go on when the USB is not being used, but I verified transmission occurred with an LED. Will check the switching circuit and code for possible errors. I am wondering if I need small delays in between each part of the message.

1 Hour

Reading through the book of serial commands that can be used. Many of them are for older models and appear to not work with this newer one. Will make sure that I have the new one in. Trying to use their example to calculate checksums and figure out how to register a user. The example given is with an ID and Password, but it is possible for the FIM to return a user ID and not have a password. This method saves two templates for the user. I will pursue this to hopefully get it to work.

SFC Comment: The amount of time on the project should be 12+ hours per week. The fingerprint reader could still be trouble and it was a good idea to get Prof. Hedrick's assistance with this at this point. Also, some simple locking mechanism should be added to show that the system works. Hopefully, the parts from China will help in the implementation.

Week 2

2 Hours
Continued to try to code the Arduino, but still I am not having success with it. Keep rechecking the code and it seems fine. Some ideas: RS232 is not correct, check the internet for example code or people who have advice about using this FIM.

3 Hours
Tried looking on the internet for some example code of how to use the Nitgen or any productive advice from people who have used this. I found that lots of people have tried to do similar projects in the past but the boards were all dead ends that people seem to have forgotten about. I checked both the Arduino site, sparkfun site, and tried Google too. I actually created a post on the Arduino website seeing if anyone had any advice on my code or had made progress since most of the posts hadn't been discussed since 2009. Ended up emailing a few of the people on the board and got a response so now I have some example codes that will hopefully jumpstart me (just happened this morning 1/19/2011 so I haven't looked at it yet). Will look over this code in the next days and hopefully make some great progress.

2 Hours
Got the MOSFET device up and running. At first it was difficult figuring out where the power and things go and how to secure the wires. But, after a few tries and closely looking at the pictures from their website the device works. I could only test it with one LED because I didn't have more. Will come back later in the week to fully test it once I have more equipment

1 Hour
Looked through old emails and logs and found the manufacturers of some good locks that are relatively inexpensive and that should work well with the MOSFET device. That being said the lock is here:
http://www.rockler.com/product.cfm?page=20132&source=googleps&utm_source=GoogleBase

&utm_medium=organic&utm_campaign=Google and it is a 12V DC electric strike. I need to power it so another wall-wart from sparkfun that is 12V: http://www.sparkfun.com/products/9442. Then to get 2 wires out of this to connect power to the actual lock we'll need this little guy which goes from the female DC jack to 2 wires: http://www.sparkfun.com/products/10288 Now confused if I need male or female adapter.

3 hours
Thinking that the Arduino code is correct I'm trying to verify my RS232 to TTL circuit is working correctly by the following example http://www.elecircuit.com/miscellaneous/TTL-Level---RS232-Level-Converter-by-MAX232-193.html but only some characters come through correctly, but none come back in capitals. Hi is not expressed as well. Measured the voltage levels and they seem to fluctuate. Main problem is the low voltage, it went from -6 to about 0 V and anything less than 3 is not good. Even the positive voltage fluctuated but was more consistently around 5 V. Worried I'm using wrong pin out diagram, though most for the MAX232 seem to be the same. Still think Arduino code for FIM could be correct will try to arrange a meeting with Prof. Hedrick

1 Hour
Examined the pin out diagrams for the max232 and think I might be using the wrong capacitor values. I have a meeting scheduled to go over the circuit but we had to postpone due to the inclement weather.

1 Hour
Tried working with MOSFET, testing all 4 switches. Seems to work quite well granted it was loaded with just LEDs, I picked up a bunch from Gene Davidson to debug, etc. But all 4 switches operated as expected, not so hard to set up and use. Here I had to alter the code to incorporate all four switches in, and used the switch to a resistor and LED.

SFC Comment: (3% Awarded) Good effort this week and good initiative to post to forums for help. The software you found should be very useful. The path for this week is clear:

- Hardware: refine Arduino - Nitgen hardware interface
- Study software to understand completely

Week 3

2 Hours:
Reading and researching the example code I was given. Looking up things like unsigned long, serial.flush, serial.available, global variables, #define function, and #include among many other things. The main idea is that I see how he prints the stuff, and my method would definitely work. Need to get the RS232 to TTL going now.

1 hour

Trying to make my serial.read part a lot better, its lacking. Need to have a good way to verify no errors and that the connections, scans, and identifications were properly performed. Need a debugging method.

1 hour
Reading about rs232 to TTL circuits including schematics and caps. Someone said 22uF caps could work, but will take longer to charge, though still no luck. Found you can use ceramic caps, will see Gene.

1 hour
Tried to do the above circuit with new caps from Gene (1uf ceramic caps) but it still not work. This is getting quite frustrating and I really need to see Prof. Hedrick for his advice and expertise on this subject matter.

1 hour
Researched how to compare errors for help on checking if acknowledgment packets are correct and that things are working well. Here is an http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1212704875 arrray compare example that can use memcmp(array1, array2, size) which should be easier and lots less code. Try to implement this in code Sunday

2 Hours
Writing code for request connection. Pretty sure this is good, even have it connected with the LCD, and
starting to use global variables to save code space. Hopefully this doesn't cause any problems in the end.
Tried to write some identification code but I am getting more confused about users and how to do this using the commands. Will have to think about this for future coding. I have a serial receive command working. That is if I send something, such as 7Eabcefghijklmnopqrstuvwxyz it prints of the first 24 characters properly

1 Hour
RS 232/TTl works. Thanks to sparkfun's diagram and my own program: http://www.sparkfun.com/tutorials/104 The arduino code didn't work and I was discouraged. But I had some idea that I didn't trust it.Connecting rx and tx seemed to work with this example, I got what I put in. Wrote my own code to say Hello Nate and it works! Finally. Good progress today

1 Hour
Trying out by request connection code out in Hyperterminal just to see what it prints. Got lots of funky symbols but if I didn't sending byets (00 is null 01 which is a little squaqre) but rather integers in decimal form I verified through HyperTerminal that the code is working. At least sent the stuff. Because flush and available aren't used on the software serial library I have to revert to unplugging tx/rx uploading, unplugging USB, plugging in TX and Rx and plugging in external power, but hey it works!

1 Hour
Riding the wave of good things I hooked up the FIM to my Arduino and once again it came through and I was happy when the LCD screen said “Connected to FIM". Then i was too excited and wrote the code to enroll a FP but that didn’t work. I will have to check the code and put more thought into it. Because I had the connection I wanted to jump on this Glad some great progress was made today

2 hours
Working on the request connection. I like my idea of using arrays. Most of this time was spent on reading serial data. We need to know the connection was made. Experimented with the PC and had some difficulty but after understanding more about serial.available() and how the data (up to 128 bytes) waits in the serial buffer I am pretty sure I understand and can write some code to verify the connection is made. Next time: put incoming serial data in to an array. Try to print the array and see if it what’s you typed. Then find a way to compare element by element of matrixes. I can put in what should be received as the acknowledgment and compare those. Alternative learn more about error codes and just check that specific portion of the array. if there is no error its good.

1 Hour
Figured out a way I could debug and ensure the connection had been made, use the LCD screen to print out what it sent me. Found out that I had in fact the correct connection and need to ALWAYS remember it prints in decimal. Don’t be discouraged when you see big numbers coming out that don’t match. Thus, a definite successful connection has been made received back 7E 00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 all in hex.

2 Hours

At this point I am trying really hard to get this thing to take a scan. I checked the previously written code and decided to go with CMD_REGISTER_FP. At first nothing seems to happen but using the above method for debugging I find an error code! Looking it up results in "Not in Master Mode". Naturally the next thing i did was write the code to enter mastermode. I did this and then debugged just entering master mode and I had successfully entered master mode after a few tries. Then I did the code for the registering the fingerprint...after entering matermode of course. I got it to at least take the scan. I did receive through my code that the scan didn’t work but for now that’s not even a problem since I wasn’t even paying attention to the scanner and probably didn’t make it in time. All that matters was I got it. Obvious next work, try to get a successful print into the system!

2 Hours
Trying to debug this fingerprint thing. I noticed that it seems as if two 0x7Es were sent, or that I am printing one which causes my check to be wrong and also messes up the LCD since that takes 3 characters, thus I am missing 3 of them which could tell me the error code. There is something resembling there result 0x0D which is "result cancelled" I notice that the example

code (uses a user id and password) ends up sending something, receiving something, sending something new, and another receive. I wonder if I am not sending that second set so its not saving and maybe then cancelling. Compacted code a bit, every function has a wait for serial data so I made a function to save a bit more space and make things clearer. Tried many different ways including my own and the example code but I keep getting that 7E. I will print the code off and really examine it as well as continue to try different methods including delays.

3 Hours
Spent some time trying to compare the code between my request for a connection and my enroll functions looking through for a difference, but they appeared identical. I didn't see any print sign for this 7E. I abandoned this and then tried some examples from the books but kept getting really weird results back like -2476 and things like that which aren't really at all what is expected. I tried tweaking lots of things within the code. Still nothing, I know the example is for a DB of 10 but I don't see how that plays into what they are sending. I have been checking through all my papers for information about size of DB and will continue to look. I also tried the first example (says its defunct) and while it did not scan either I didn't get so many weird things back though it appeared like the master mode thing was playing a role as I got 00, 00, 00, 2F which is the command for master mode where I would expect the command for the enroll_FP_STEP2 command. Need to figure this enrolling stuff out, then identify and more about the user database and how many we will have with only 1 lock. Maybe we could go down the route of multiple users for one locker?

SFC Comment: (3% Awarded): Excellent progress for this week. Let's go down the simplest route first and have multiple users for 1 locker. If there is enough time at the end, we can try to expand the test cases. Let's try and get the overall system working first.

Week 4

1 Hour
Worked with the enroll command that I had written. Sometimes I received funny data back and was wondering why this happened and if it was affected by the fact that there are 4 communications performed during the process (2 to the FIM and 2 back from it). I wrote a delete all command which needs master mode but it works well.

2 Hours
Worked a bit more with the enroll process but I now extended my code to include an identify function. I am still getting some weird numbers back from the device with enroll and I need to investigate this so I can solve that problem. I read in some places (example code and internet) that there is "endian" nature and that the data might essentially be backwords, but I don't think this applied to my code otherwise I wouldn't have been able to enter MM or get the connection. Keep this in mind though.

1 Hour:

I am almost sure the enroll_fpstep 1 and step 2 commands are obsolete in this one, but I am still confused on how to use data. Definitely use c md_Register_FP. I tried to send the data in a few different fashions (with step 1 and step 2 method) but none of them worked.

2 Hours:
The enroll function works really well but just for me since I am familiar with the system, like how to put the finger there and when. Now I need to make if statements for the errors such as timing out of the scans and getting errors back. Also worked on identification and got that to work. I received back a data block from the device and my FP ID was 0000. I tried with different fingers and didn't receive that data along with 0x02 instead of 01 indicating failure. Continue with this, like above by making IF statements for all scenarios received.

1 Hour:
Experimented with the identification code and think it works quite well. I don't think I need to actually use the fact that the users have an ID, just that they are in the DB. Might need this if you have multiple lockers though! I enrolled 4 of my fingers and it identified them all as in the DB and I also tried 2 other fingers and they were not recognized.

3 Hours
Back to the enroll function with the weird received data. Now I get the device to send back to 7Es which doesn't make sense with my checking method. I compensated for this temporarily by shifting my check bit but need to figure this out. The 2nd template always seemed to save to the DB…even if my finger wasn't there but I realized I was examining the data sent back from the first template and created a new variable for this data. Still have the double 7E, but if you do the enrolling properly it doesn't have this problem, WEIRD! Need to go through and test this, debugging why I am getting this.

3 Hours
Sick of this 7E stuff, I went into a test and debug mode. I tried lots of things and just printed out everything I got back to see what results causes what serial data. After doing countless tests, see the document I will try to attach, I think I understand this stuff a lot better. In the end to keep it short, I split it into 2 functions (like the ones that didn't work). If the first received data is correct, I call the second function which was written outside of the function running. This seemed to fix all problems with the 7E stuff and my checks for errors all worked well.

3.5 hours
Coded in identification error message, like timeouts, cancelled, failure, etc. I tested it in the same manner I did for enroll, and see the attached document for more details. I have to incorporate the same delay of 5.25 seconds after the call for the scan, otherwise the timeout is skipped because the data was sent to the Arduino until after the scan timed out (about 5 seconds of light). Now I also stripped down the wall-wart we got in, soldered it to the alligator clips, and hooked it up to the DMM. Red=positive and blue=negative. Soldered the locking to the breadboard connecting device (similar to what I did for the MLX) and connected it of the

wall-wart. The lock works well with a pushbutton switch, applying power you hear a click and while power is applied the lock can move. Otherwise it is locked.

4.5 Hours
Got a primitive or rough draft system working, that is I incorporated the lock in with the MOSFET device. Now when it identifies, it prints to the LCD and the lock actually opens! Now I am realizing more serious things about the code that need to be done, for instance, limited master mode access. Not everyone can or should be able to enter the system, so how about if my FP is scanned correctly they can enter, otherwise it fails. I want to enroll my FP as a master. Identification doesn't need master mode so it can be down any time as needed to get the lock open. I got it to successfully take me in as a master, and then I was locked out for a long time. In order to get into master mode now you need my print along with my ID. So what is my ID? I couldn't figure it out, and I thought it was 0x30 0x30 0x30 0x30. Of course it was printed on the LCD in decimal so I sent to it 48 48 48 48 and I got weird responses and everything. I knew the first ID assigned was typically 0000 for users but I didn't know if it was the same for masters. Then I realized you can get the ID from identify. I changed my code to spew out the ID number on the LCD and saw 48 48 48 48, just like I thought. Then I went into my excel sheet which I used previously to switch the decimal number to hex and realized it was 0x30's, changed my code, and immediately got into master mode. This was very frustrating but in the end it worked out ok and my old code still worked the same with me as the master. Next step is for me to go into MM, delete all FP and see if the master one stays. If it doesn't then go back in MM using the null (since no FP in DB) and try to enroll my FP with a specific ID. I feel like 0000 is a bit boring so I will try to enter a specific one, if not I will keep 0000. Then work on something with pushbuttons and controlling my code with them. One for enroll (need my FP though!), one for identify, and maybe more if I see the need for them (delete).

SFC Comment: (3% Awarded): Very good documentation of the testing performed. We may include this as an appendix to the report. The end to end system seems to be working now. So, start making backups of all the code. You could upload to the Google Docs folder once a week also as a checkpoint.

I think I should see a demo this week so I can see how it looks to a user.

Week 5

3 Hours
First I went in with my master mode ID of 0000 and deleted all FP. This then allows you to once again enter MM without a FP as there are none in the DB. Then I set my right index finger in as the master with ID 1313 and Password 1989. You can use either the ID or password to enter as before but I want to use the FP to control who can enter. This worked

Next I tried some stuff with pushbuttons. If its high enroll a user, if a different one was high identify. At first I had some trouble with what was returned from digitalRead along with how to set the toggle switches up but I got it working so it sits there waiting for input. It was a bit touchy

literally; if you just grazed the toggles it went into enrolling. Tried to add a nested loop with a delay and I think this will fix it. This will require the switch to be high for a few seconds before it enrolls, this would have to be a deliberate action not by chance. Things to do: Master this push button stuff, think of alternative to enter this system. This way it is autonomous. I think I should start use IDs soon then I could delete users. It will also help for implementing this with multiple lockers, since then you will need an ID to get to your locker.

2 Hours
The enroll with the toggle switches works well now. I changed my code up a lot to make more functions for printing which makes the enrolling, etc functions look a lot neater. Works ok, noticing some glitches in the prints though and I need to go through and find them all so I can make the system as smooth as possible. Separated my code, I don't need enrolling master FP on the regular code, as that should be uploaded to the Arduino separately. Tried to go through and delete all FP and enroll my master print again but the LCD decided not to work on the copied code so I hope I didn't mess the FPs up. I will fix this first thing next time and enroll my master FP again. Then do some more testing of my functions with the switches. Also keeping in my mind the ID number stuff and how I could implement that.

1 hour
The LCD seems dead, not sure what happened. I get a line if black squares on the 1st line. This means according to someone herehttp://www.avrfreaks.net/index.php?name=PNphpBB2&file=printview&t=25912&start=0 that it is getting 5V but not initializing correctly, I don't see the sparkfun message either. Will continue to investigate this as I need it. Used the 2nd set of connections on the MAX232 to set up a channel to have my prints displayed in hyperterminal because I need to know what I am doing.

1 Hour
Thanks to the 2nd channel on the 232 I was able to get my code up and running with hyperterminal so this is my short term solution. I enrolled a master and all that, I had some glitches and had to comment out a lot of code but it works. Backing up code was a great idea! will try to explore assigning IDs to enrolled users next since I think that might me best. Its also easier to debug using hyperterminal since everything prints, you don't need the delays. Maybe this diversion will work out. I will leave the identify stuff out for now along with the lock and pushbuttons and focus on getting this other code perfect. I think its best to have 2 sets of code, one for enrolling master etc and the other that requires you to enter master for enrolling but there is no way to become the master there.

3 Hours
Ended with master FP ID: 1313
Now I have 2 separate codes so entering a master FP is completely separate from identify and enrolling. Essentially one code is for masters and the other is for the regular system. of course this entails entering mastermode but if you fail to do so then you cant enter FP. Delete all is in the master code, because if you do this, you will have to enter a master FP again so I took it

from the regular code. Later I would like to implement the delete user code which uses a specific ID. I also was able to enter a user with a specific ID so this will be more applicable. I have the idea of for now storing 16 users in the DB and giving them IDs 0000, 0001, 0002,.., 000F. I think I could store these in an array and have a little counter so if a user is enter it ups the counter by 1.
Another thing is that I got rid of my If statements and replaced them with cases, this seems to just be easier for whatever reason and looks more efficient allowing me to debug problems easier. Seems much simpler his way.

2 hours
Cleaned up my circuit because it was messy and tried a different wire setup for the pushbuttons but that failed, they both need their own power. Changed lots of small snippets to make better printing things. Have one problem with my Enroll function. I can enroll but then after the statement "need to be in MM to enroll" is printed and I thought this should be skipped over due to the cases. This is supposed to be used for cases where MM couldn't be entered for whatever reason. I need to use my master code after every time i enroll prints because otherwise I have a repeated ID error. Will upload this code tonight for the master since I think that is permanent. I also think I know how to implement 4 different locks, since know IDs are assigned (up to 16 users). I suppose I could test this with the LEDs, and should I try to hook this up to a different lock like the door one?

2 Hours
Debugged the code so that the "Need to be in MM to enroll" is not displayed every time. See what I did for specific methods. In short Cases within cases didn't seem to mesh, so I changed the first case to be an IF ELSE and it works well now. I also added a similar statement using a variable inc so that if the incoming function finds the FIM unresponsive it will skip the remaining code. There is no point in sending and print more information if it isn't working properly. Overall I think I am ready for a demo for 1 lock with multiple users

Backing up code by saving it new every day and made a file for code in the shared file on google documents. Every time I end I delete all FP and reenroll the master FP with ID1313. Uploaded an updated version of "WhatIdid" for debugging the master mode problem.

SFC Comment: (3% Awarded): The demo worked well for me and it seems quite robust. We just need to polish the overall system now to make it look a bit more professional and make the UI prettier by using scrolling on the LCD screen.

Also, we should test the system in some user trials to determine how well it

---

Week 6

2 Hours
FIM needs to be sturdy in the design, seems to have a failure if it wobbles a bit, pinky and thumbs are bad. I had some troubles with enrolling a user but I think it was a loose wire and wobbling. I wrote up my application for human involvement in research and will hand this in tomorrow.

1 hour
Fixed my problem of every time I unpowered I lose which ID is next (of course I could have used my older function but I like having IDs). I use EEPROM and save a variable g which is used to access the gth element of a vector which houses the ID numbers. When the master program is uploaded and run, g is set by EEPROM address 0 to be hex 00. Then before calling g up in the normal program I read the 0 address of EEPROM and g becomes 0 the first time. After enrolling a user g is incremented and then re-wrote to EEPROM thus increasing the index so the next user as the next ID. Works nicely when the power is unplugged still can hold up to 16 users and when there is more you get unknown failure (ID already in use after all 16 IDs are filled up

2 hours
Worked on my website, soldered new LCD, hooked it up (works!) and also experimented a lot with the scrolling function. Turns out my second line function is useless, it automatically goes there by itself so I will try not to use it. Scrolling does both lines 1 character at a time. Clearing the LCD afterwards works out ok and the cursor returns to the first place if first line is called. Will implement this fully in my system tomorrow and make it friendlier.

4 Hours
Found out after some struggling of literally just staring at my code then my circuit that I need a pull up resistor between the ground otherwise my pin could be undefined and this circuit noise could cause it to run high. See http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1267209223 for the explanation and simple solution. Seems to be working well, wonder why I didn't have this problem before, oh well.

Experiment with quality:

IDENTIFICATION:

Full finger strong press: 63, 71
Full finger medium press:73, 70
Full finger light press:didnt realize i was there....50, 50
Full finger dab:72, 59
Half finger hard: FAIL, have to check these in enroll
half finger medium:
half finger light:
half finger dab

ENROLLL

full finger strong:65, 62
FF medium: 59, 65 (left index)
FF light:52, 50
FF dab: 69 75
Half finger hard (vertical): 57 and too slow, 64, 49
HF medium/light (vertical): too slow, failure, 63, 58
HF (side): 57, 63

Set it for 65 to enroll.

Master quality check still doesn't work well, but I know my print is ok.

if you don't leave MM then no FP is required to renter. FP quality of 65 is iffy sometimes with entry. More pain=more security. If the first template is poor quality it breaks the loop but if the 2nd one is poor it is technically saved. So you leave MM, enter it again, delete that print.
This cutoff seems to work well, so I began the testing. We will see how the results are and if the cutoff needs to be increased though I believe any higher and it will be too hard to enroll as it is already somewhat difficult. I have enrolled 4 users in my DB and will continue until the DB is full, then test the identification process 10 times for each user. So far the response is good and the system seems to work well.

1 Note. Infinite loop if the 2nd quality check is poor the FP must be deleted and for that the master FP is needed. If its not scanned correctly you are stuck.

A good hard day of work, constantly updating printing statements to look better. I think it would be a good idea to also ask the users what would make the system easier to use during the identification process. Changed the master code as well so its fit for the LCD again.

1 hour
continued to enroll new users. Some go in smoothly some do not. I think movement of the sensor greatly affects the success, it needs to be still for a good enrollment, this is key for the design of the final product. Also, due to variation with fingers it is possible their prints are good, for instance guitar players may have calluses so use the other index finger. I have a total of 8 users in the DB and will continue to get my 15.

2 Hours
Worked on my website: antipasto.union.edu/~clarkn I think it looks ok, just need to change the Final picture when I have the design enclosed.

1 Hour
Enrolling users in the DB. Having some difficulty but I believe a hard surface that is sturdy is a good idea in general and the scanner shouldn't be moving at all during the scans. Maybe the

cutoff is to high but again some users prints go in on the first try, some after 2-3 tries, and some take many more attempts. It seems they are doing it ok, but the quality is poor. In general, the system is working fine still, its just the print quality threshold is not being reached. Will keep trying to enroll users right now I have ID 0008 in so a total of 9 users. More than half are in and then I will test which is quick.

SFC Comment: (3% Awarded): Testing seems to be going well. Be sure to document how this goes and see what the user thinks of using the system. This might point to some future improvements to the UI. The enclosure you mentioned sounds promising so let's see can the entire system be housed in a nice package. It should all be very close to done next week.

The website looks pretty good to me. It does not have to be the prettiest in the world. It has most everything we need.

Week 7

2 Hours
Continue to enroll new users and am well on my way to a full database but now I am experiencing one problem. It is failing to save the FP and the message "scan failed please try again" was displayed. So, I need a way to test it and I think if I change the ID numbers as well as some of the EEPROM stuff I can make it so the users don't need to be deleted.
Ideas:
1. Make a new variable to count the IDs with...how about H. Comment out/delete all things to do with g. We want to preserve whatever g is that way the ID numbers stay in tact for the,

2. Change the cheksum and insum statements to reflect the new variable H

3. Change the ID_2 if it is a success to have H, dont save it to the EEPROM and make sure there is nothing with g
So I tried it with my left index finger and it worked. Will not try this user again. Seems to really through off the system. But using a tissue to clean the scanner really helps with the quality! Left index= 000B. I didn't want to enroll another of my fingers but it appears this users FP is not so compatible with the system, but I will continue enrolling and test soon.

2 Hours
Have all 16 users enrolled in my system, and have identified 3 of them and so far the identification has been smooth, and I have received good feedback along with nice suggestions. The enrollment is clearly much more difficult than the identification process. Will continue to test and also plan out the final design. I have started with some basic drawings, after I am done I will start to do some soldering and talk to gene on Monday about drilling in the boxes I bought.

1 Hour

Working through the testing, also noticed that there is a < on the FirstDraft webpage so I will have to fix that. So far the response has been very positive and the systems seems easy to use while the

2 Hours
Still collecting data, finally got my 1st access denied, though the user did state he thought he lifted his finger too early, either way this is part of the study so it's not perfect though it seems to be working extremely well. More than half way done with collecting the data. Also got a few other access denied mainly with one person, they stated that putting their finger down after the scanner light was on resulted in access denied but putting it down early was successful. Later I tried this method with my fingers and I didn't have this same problem.

4 hours
I finished the testing, cleared all the users FPs, made small changes to the code to make the system better, and uploaded the newly altered version of the code. The scanner's accuracy was 96.25 %. Started construction of the final project, soldered the max232 together as well as the pushbuttons, just need some breadboard I (makes it easier) to have the 9 pin out of the FIM, and then reconnect everything together. I will see Gene tomorrow about drilling holes.

2 Hours
Talked with Gene and have been laying out where I will make my cuts. He recommends going to the machine shop to seek their guidance cutting the pieces out and hopefully it won't be too hard for them. Wrote my abstract for Steinmetz as that needs to be submitted soon

1 Hour
Made some more marks on my box for where the power cord and USB cord to the Arduino will go, and went to the machine shop. Paul was very helpful and said he will do the cuts for me, hopefully by Friday or Monday. From there I will have to secure the major components, with double sided tape or glue and make the final connections.

SFC Comment: (3% Awarded): The unit looks very good and well laid out. It came together more quickly than I thought. Now just keep it safe between now and the the presentation/demo. Work on the presentation and poster.

---

4 hours

Worked on mounting everything with Gene. All of the components are nicely secured with screws/washer or double sided tape which seems to be quite strong and durable, a big thanks to him. Then I made all the final connections, including fiddling around trying to find the best lengths for the wires. The toggle switches were hard, but I used some of the multi-stranded wire and it helped a lot. I wont the lid to be able to be opened easily for re-uploading master code. Had some frustrations with connecting the lock as it was always open, then I realized I had the

positive and negative leads to power the MOSFET wrong, woops. The problem was easlily solved and the system worked as it did before but now has the professional appeal. Labeled the switches so that everything is clear. The box is relatively durable, can be turn upside down, etc. The wires should resit pull as by Gene's suggestion I tied a know on the inside so pulling on them will not unplug components.

The rest of the week was spent working on the presentation and poster.

Appendix 4: Arduino Master Code

```
#include <SoftwareSerial.h>
#include <EEPROM.h>

#define maxim 50
//#define powerLCD 3
byte txPin = 6;
byte powerLCD = 5;

SoftwareSerial LCD = SoftwareSerial(0, txPin);
// since the LCD does not send data back to the Arduino, we should only define the txPin
int d = 0;
int i;
int f;
int header_rx;
int M;
int g = 0x00;
int inc;
int quality_check = 0;
void setup()
{
  EEPROM.write(0, g);
//  pinMode(rxPin,INPUT);
  pinMode(txPin,OUTPUT);
  pinMode(powerLCD, OUTPUT);
  digitalWrite(5,HIGH);
  LCD.begin(9600);
  Serial.begin(9600);
  delay(250);
  Serial.flush();
  LCDon();
  delay(2000);
  ReqConn();
  Welcome();
  delay(2000);
  DeleteAll();
  delay(2000);
//  EnterMaster();
//  EnterMaster_noFP();
//  LCD.print("Enroll Pinky");
//  Enroll_ID1();
//  delay(10000);
//  delay(2000);
  Enroll_Master_FP1();
//  delay(3000);
//  LCD.print("Leave Master");
//  LeaveMaster();
```

```
//  delay(2000);
//  LCD.print("Enroll");
//  delay(2000);
//  Enroll_ID1();
//LCD.print("Trying to Delete All");
//DeleteAll();
//delay(3000);
//LCD.print("Tring to Delete All");
//DeleteAll();
}

void loop()
{
}

void ReqConn(){
  Serial.flush();
  int rx[24];
  int bytes[] = {0x7E, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01};

  for (i = 0; i < 25; i++) {
  Serial.print(bytes[i], BYTE);
  }

  delay(2000);
  Incoming();

  if (Serial.available() >= 24) {
    for(i=0;i<24;i++){
    rx[i]=Serial.read();
    }
  }

switch (rx[7]) {
  case 0x01:
    clearLCD();
    firstline();
    LCD.print("Connected to FIM");
    delay(2000);
    break;
  default:
    clearLCD();
    firstline();
    LCD.print(" The connection");
//    secondline();
    LCD.print("   has failed");
    delay(2500);
```

```
   clearLCD();
   firstline();
   LCD.print("Please try again");
   delay(2500);
   break;
}
}

//Clears the LCD display of all characters
void clearLCD(){
LCD.print(0xFE, BYTE);   //command flag
LCD.print(0x01, BYTE);    //position which starts the second line
}

////Turn on the LCD Display
void LCDon(){
LCD.print(0xFE, BYTE);   //command flag
LCD.print(0x0C, BYTE);    //position which starts the first line  change to 0x0C if you don't want a blinking
cursor
}

void firstline(){
LCD.print(0xFE, BYTE);   //command flag
LCD.print(0x80, BYTE);    //position which starts the first line
}

////The following function sets the cursor to the first position on the first line
void secondline(){
LCD.print(0xFE, BYTE);   //command flag
LCD.print(0xC0, BYTE);    //position which starts the second line
}

//This command will attemp to enter "master mode" based on the FP in the database designated as
masters, the masters ID is require as well.
void EnterMaster(){
  M = 0;
  clearLCD();
  firstline();
  LCD.print(" Verifying the");
//  secondline();
  LCD.print(" master FP now");
  delay(2000);
  Serial.flush();
  int rx[24];
  int f;
```

```
//  int master[] = {0x7E, 0x00, 0x00, 0x00, 0x2F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x39, 0x30, 0x30, 0x30, 0x30, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0};

  int master[] = {0x7E, 0x00, 0x00, 0x00, 0x2F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x39, 0x31, 0x33, 0x31, 0x33, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC8};

  for (i = 0; i < 39; i++) {
  Serial.print(master[i], BYTE);
  }

  delay(5250);
  Incoming();

  if (Serial.available() >= 24) {
    for(i=0;i<24;i++){
    rx[i]=Serial.read();
    }
  }
  switch (rx[7]) {
    case 0x01:
      clearLCD();
      firstline();
      LCD.print("In Master Mode");
      delay(2000);
      M = 100;
      break;
    default:
      clearLCD();
      firstline();
      LCD.print("Failed to enter");
//      secondline();
      LCD.print(" master mode");
      delay(2000);
      M = 13;
      break;
  }
}

void EnterMaster_noFP(){
  Serial.flush();
  int rx[24];

  int master2[] = {0x7E, 0x00, 0x00, 0x00, 0x2F, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x32};
  for (i = 0; i < 25; i++) {
  Serial.print(master2[i], BYTE);
```

```
  }

  delay(2000);
  Incoming();

  if (Serial.available() >= 24) {
    for(i=0;i<24;i++){
    rx[i]=Serial.read();
    }
  }
  switch (rx[7]) {
    case 0x01:
      clearLCD();
      firstline();
      LCD.print("In Master Mode");
      delay(2000);
      break;
    default:
      clearLCD();
      firstline();
      LCD.print("FailedtoenterMM");
      delay(3000);
      break;
}
}

void Incoming(){
    for(d=0;d<maxim;d++) {
    if (Serial.available() > 0) {
      header_rx = Serial.read();
      delay(50);
    if (header_rx == 0x7E){
      clearLCD();
      firstline();
//      LCD.print(" FIM responding");
//      secondline();
//      LCD.print("  Please  wait");
      break;
  }
  else{
    clearLCD();
    firstline();
    LCD.print("FIM unresponsive");
//    secondline();
    LCD.print("Please try again");
    delay(25);
    break;
    //NEED A WAY TO GET OUT Of THIS INCASE THE CONNECTION HAS FAILED, could be an infinite loop
```

```
  }
  }
  }

  if (header_rx == 0x7E) {
    for(d=0;d<maxim;d++) {
      if (Serial.available() >= 24) break;
      delay(75);
    }
  }
  }

void LeaveMaster(){
  Serial.flush();
  int rx[24];

  int master[] = {0x7E, 0x00, 0x00, 0x00, 0x26, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x26};

  for (i = 0; i < 25; i++) {
  Serial.print(master[i], BYTE);
  }

  delay(2000);
  Incoming();

  if (Serial.available() >= 24) {
    for(i=0;i<24;i++){
    rx[i]=Serial.read();
    }
  }

  switch (rx[7]) {
    case 0x01:
      clearLCD();
      firstline();
      LCD.print(" You have left");
//      secondline();
      LCD.print("  master mode");
      delay(2000);
      break;
    case 0x03:
      clearLCD();
      firstline();
      LCD.print("You were not in");
//      secondline();
      LCD.print("  master mode");
      delay(2000);
```

```
     break;
   default:
     clearLCD();
     firstline();
     LCD.print("Problem leaving");
//     secondline();
     LCD.print(" master mode");
     delay(3000);
     break;
  }
}

void Welcome(){
  clearLCD();
  firstline();
  LCD.print("    Welcome");
//  secondline();
  LCD.print("Select an option");
}

void Slow(){
  clearLCD();
  firstline();
  LCD.print("Your finger scan");
//  secondline();
  LCD.print("  was too slow");
  delay(4000);
  clearLCD();
  firstline();
  LCD.print(" Please try the");
//  secondline();
  LCD.print("   scan again");
  delay(3000);
  Welcome();
  }

void NotMaster(){
   clearLCD();
   firstline();
   LCD.print("Need the masters");
//    secondline();
   LCD.print("FP for DB entry");
   delay(4000);
   clearLCD();
   firstline();
   LCD.print("   Please try");
//    secondline();
   LCD.print("     again");
```

```
   delay(3000);
//    Welcome();
}

void TwoFingers(){
   clearLCD();
   firstline();
   LCD.print(" The templates");
//    secondline();
   LCD.print("   dont match");
   delay(4000);
   clearLCD();
   firstline();
   LCD.print(" Please try the");
//    secondline();
   LCD.print("   scan again");
   delay(3000);
//    Welcome();
   }

void Unknown(){
  clearLCD();
  firstline();
  LCD.print("Unknown Failure");
//  secondline();
  LCD.print("Please try again");
  delay(3000);
//  Welcome();
}

void Failure(){
   clearLCD();
   firstline();
   LCD.print("Your finger scan");
//    secondline();
   LCD.print("   has failed");
   delay(4000);
   clearLCD();
   firstline();
   LCD.print(" Please try the");
//    secondline();
   LCD.print("   scan again");
   delay(3000);
//    Welcome();
}

void Enroll_Master_FP1(){
  LCD.print("Enrolling Master");
```

```
  EnterMaster_noFP(); //What about a global variable so I can have cases here, if you are not in master mode, because I am not there, dont even bother doing the rest, break out
  delay(100);
  Serial.flush();
  int rx[24];
  int f;

  int enroll_master1[] = {0x7E, 0x00, 0x00, 0x00, 0x33, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x4E, 0x31, 0x33, 0x31, 0x33, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x31, 0x39, 0x38, 0x39, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xA3};

  for (i = 0;i < 55;i++){
    Serial.print(enroll_master1[i], BYTE);
  }


  delay(5250);
  Incoming();
  if (Serial.available() >= 24){
    for (i = 0;i < 25;i++){
      rx[i] = Serial.read();
    }
  }

switch (rx[7]) {
  case 0x01:
  Quality();
  delay(1500);
  clearLCD();
  firstline();
    LCD.print("Sending 2nd");
    delay(3000);
    Enroll_Master_FP2();
    break;
  case 0x07:
    Slow();
    LeaveMaster();
    break;
  case 0x03:
    clearLCD();
    firstline();
    LCD.print("Need MM to enter");
//    secondline();
    LCD.print("  a master FP");
    delay(3000);
  case 0x02:
    Failure();
```

```
   LeaveMaster();
   break;
  case 0x0D:
   Failure();
   LeaveMaster();
   break;
  default:
   Unknown();
   for (i = 0;i < 24;i++){
    LCD.print(rx[i]);
   }
   LeaveMaster();
}
}

void Enroll_Master_FP2(){
 Serial.flush();
 int rx_2[24];
 int f;

 int enroll_master2[] = {0x7E, 0x00, 0x00, 0x00, 0x33, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x35};

 for (i = 0;i < 25;i++){
  Serial.print(enroll_master2[i], BYTE);
 }
 delay(5250);
 Incoming();
 if (Serial.available() >= 24){
  for (i = 0;i < 25;i++){
   rx_2[i] = Serial.read();
  }
 }
switch (rx_2[7]) {
 case 0x01:
 Quality();
 delay(1400);
 clearLCD();
 firstline();
  LCD.print("FP saved as a");
//   secondline();
  LCD.print("master ID 1313");
  delay(3000);
  break;
 case 0x07:
  Slow();
  LeaveMaster();
  break;
```

```
  case 0x03:
    clearLCD();
    firstline();
    LCD.print("Need MM to enter");
//    secondline();
    LCD.print("  a master FP");
    delay(3000);
  case 0x02:
    Failure();
    LeaveMaster();
    break;
  case 0x0D:
    Failure();
    LeaveMaster();
    break;
  case 0x0E:
    TwoFingers();
    LeaveMaster();
    break;
  default:
    Unknown();
    for (i = 0;i < 24;i++){
      LCD.print(rx_2[i]);
    }
    LeaveMaster();
}
}

void DeleteAll(){
  LCD.print("Deleting all FP");
  EnterMaster();
  switch (M) {
    case 100:
    {
      Serial.flush();
      int f;
      int rx[24];

      int delete_send[] = {0x7E, 0x00, 0x00, 0x00, 0x23, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x23};

      for (i = 0; i < 25; i++) {
        Serial.print(delete_send[i], BYTE);
      }

      delay(2000);

      Incoming();
```

```
    if (Serial.available() >= 24) {
     for(i=0;i<24;i++){
       rx[i]=Serial.read();
     }
    }
    switch (rx[7]) {
     case 0x01:
       clearLCD();
       firstline();
       LCD.print(" Deleted All FP");
       delay(3000);
       LeaveMaster();
       break;
     default:
       for (i = 0;i < 24;i++){
         LCD.print(rx[i]);
       }
       LCD.print("Didn't Delete");
       LeaveMaster();
    }
    break;
  }
  case 13:
  {
    clearLCD();
    firstline();
    LCD.print("Need to be in");
//     secondline();
    LCD.print("MM to delete");
    break;
  }
 }
}
void Quality(){
  int f;
  Serial.flush();
  int rx[24];
  clearLCD();
  firstline();
  LCD.print("  Checking the   print quality");

  int quality[] = {0x7E, 0x00, 0x00, 0x00, 0x68, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x68};

  for (i = 0; i < 25; i++) {
    Serial.print(quality[i], BYTE);
    }
```

```
  delay(4000); ///Assuming it needs the same delay in case of a time out
  Incoming();
  if (inc == 0x0A){

  if (Serial.available() >= 24) {
   for(i=0;i<24;i++){
     rx[i] = Serial.read();
   }
  }
  switch (rx[7]) {
   case 0x01:
     clearLCD();
     firstline();
     LCD.print("Quality of ");
     LCD.print(rx[11]);
     delay(2000);
     break;
   default:
     Unknown();
  }
  }
  else{
   clearLCD();
   firstline();
   LCD.print("No data recived");
  }

  delay(2000);
}
```